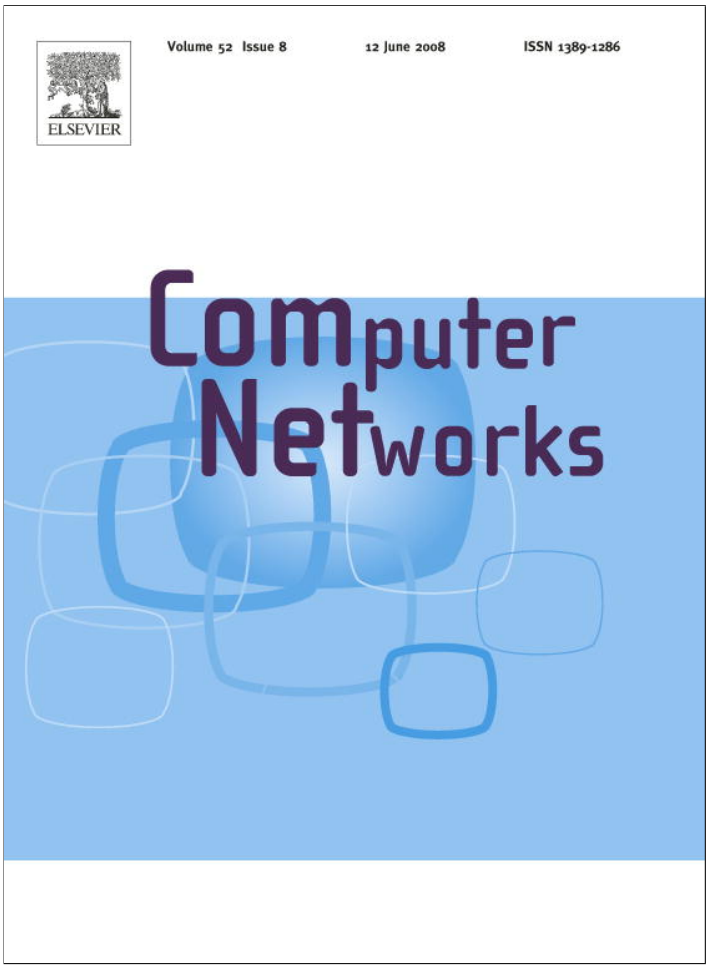


Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 52 (2008) 1545–1566
Computer Networks
www.elsevier.com/locate/comnet

Detector SherLOCK: Enhancing TRW with Bloom filters under memory and performance constraints

Seung Yeob Nam [a,*], Hyu-Dae Kim [b], Hyong S. Kim [c]

[a] Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 712-749, Republic of Korea
[b] KAIST Institute for Information Technology Convergence, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Republic of Korea
[c] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States

Received 23 February 2007; received in revised form 11 January 2008; accepted 17 January 2008
Available online 1 February 2008

Responsible Editor: L. Salgarelli

Abstract

Computer worms and bots are significant threats to large networks because they can spread very rapidly and are used for DDoS. The first phase of worms and bots begins by scanning vulnerable hosts. Missing on-going scanning activity can significantly deteriorate network performance. We propose a new scanning detection scheme, SherLOCK, based on the connection attempt success ratio. The proposed scheme can detect scanners with guaranteed false positive and false negative probabilities and with a limited memory size. Detection of scanners at high-speed links requires a high-speed memory and such memory devices are expensive and limited in size. We reduce the memory requirement by applying the Bloom filter. We show how slow scanners can be detected with a guaranteed performance for a given offered traffic load and memory size. This study can help to design the system that satisfies the target performance requirement. The detection performance is guaranteed under the assumption that malicious scanners and benign hosts have distinct behaviors in terms of the connection success ratio. We extend the proposed detector with a sampling mechanism to detect more intelligent scanners with guaranteed performance. These include scanners that use a list of pre-acquired IP addresses. We evaluate the performance of the proposed scheme through experiment using well-known traffic traces.

Keywords: Scanner; Slow scanner; Scanner detection; Connection attempt success ratio; Bloom filter; Memory conflict

1. Introduction

Internet attacks such as distributed denial-of-service (DDoS) attacks and worm attacks are increasing in severity. According to CERT [1], the number of reported network attack incidents has grown exponentially, and it continues to increase. Computer worms and bots are significant threats to large networks as they can spread very rapidly and are used for DDoS [2–5]. The first phase of worms and bots begins by scanning vulnerable hosts. In this paper, we address the detection of such scanners

* Corresponding author. Tel.: +82 53 810 2551; fax: +82 53 810 4742.
E-mail addresses: synam@ynu.ac.kr (S.Y. Nam), papin@comis.kaist.ac.kr (H.-D. Kim), kim@ece.cmu.edu (H.S. Kim).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.01.008

in order to prevent further attacks and to minimize the damage.

There are many important issues that need to be considered for a practical scanning detection system. First, memory requirements should be considered. Since inter-packet arrival time can be very short due to a high link rate, SRAM is usually required for fast packet processing [6]. Second, low false positive and false negative probabilities need to be guaranteed in order to suppress explosive growth of worm infection and to minimize interruptions in normal traffic. Third, the detection algorithm should be simple so that it can cope with bursts of packets arriving at a high speed. Fourth, scanners need to be detected as early as possible so that the damage can be minimized. Fifth, detection of slow scanners is also important since slow scanners can easily bypass most window-based detection schemes. There have been many approaches to the scanner detection problem [6–18], but there are few schemes that address all the above issues collectively. The closest to our objectives was Weaver et al.'s scheme [6], which we compare with our proposed system, SherLOCK, in Section 6.

Since it is shown that normal hosts and scanners exhibit different behavior in terms of connection attempt success ratio in [7], we use the statistical characterization of connection attempt success ratio to detect scanners. The resulting detection scheme guarantees false positive and false negative probabilities with a small memory requirement. We apply Bloom filters to reduce the memory size required for connection state management. We find that 16 MB SRAM is adequate to handle a large number of flows in core routers [19].

The contribution of this paper is summarized as follows:

- Although the threshold random walk (TRW) [7] and the optimized TRW [8] are the state-of-the-art techniques in the scanning detection area, the memory constraint is not considered in these works. The optimized TRW by Schechter et al. [8] can detect scanning worms faster than the original TRW [7] especially when a normal host gets infected, but managing and searching among 64 destination addresses for each local host is challenging especially in a high-speed environment, e.g. at least 1 Gbps, due to the limited memory access time and the limited memory size. SRAM is usually required to process packets arriving from high-speed links, but the size of SRAM is still very limited. In this memory-limited environment, the TRW scheme [7] and the optimized TRW [8] may not guarantee the false positive and false negative probabilities due to memory conflicts. Weaver et al. [6] propose a modified version of the TRW scheme considering these memory issues, but this modified scheme does not guarantee the false negative probability [6]. We propose SherLOCK that considers these memory conflict issues and guarantees the false positive and false negative probabilities with a limited memory requirement.
- Although there are schemes which can detect slow or stealthy scanners [16,17], SherLOCK is the first one to detect slow scanners with guaranteed performance under the condition of a limited memory size. We derive a relation which shows the minimum detectable scanning rate for the given traffic load and the memory size. It is shown that slower scanners can be detected as the memory size increases in a quantitative fashion.
- Many of existing scanning detection schemes [6–8] assume malicious hosts are doing random scanning without any knowledge of valid IP addresses. Our extended scheme based on sampling can detect more intelligent scanners. These intelligent scanners know a limited number of valid IP addresses in advance and can successfully bypass short-history-based detection schemes.

The rest of the paper is organized as follows. In Section 2, we discuss related works. In Section 3, we describe the detection mechanism of SherLOCK and the structure of the tables that are used to store connection status information in detail. In Section 4, we first investigate the memory conflict in the hash table to detect slow scanners for the given offered load and the memory size. We then analytically derive the detection threshold and the memory size which can guarantee false positive and false negative probabilities under the memory conflict in Bloom filters. In Section 5, we extend the basic version of SherLOCK by incorporating a sampling technique to detect more intelligent scanners. In Section 6, we evaluate the performance of the proposed scheme through experiment and compare the performance of SherLOCK with that of the simplified version of TRW developed by Weaver et al. [6]. Finally, conclusions are given in Section 7.

2. Related works

We can classify existing schemes into four categories depending on the metrics used to detect scanners. The metrics are connection attempts [16], connection attempt rate [9–12], connection attempt failure rate [11–14], and connection attempt failure ratio [6–8]. The connection attempt-based schemes such as Bro [16] can detect slow scanners, but they usually have high false positive rates. In case of either connection attempt rate-based scheme or connection failure rate-based scheme, the scanners can evade detection by lowering the scanning rate. A low detection threshold may incur many false alarms, and thus, determining the threshold is a very difficult problem.

Spice [17] detects scanners, including stealthy scanners, based on the anomalous behavior. However, it may not be suitable for fast detection on high-speed links since it requires a complex computation that does not scale with the link speed. Furthermore, selection of the threshold for the anomalous event reporting is quite difficult.

TRW scheme [7] and optimized TRW scheme [8] use a connection failure ratio-related metric, a likelihood ratio, to detect the scanning activity rapidly with a small number of connection attempts, usually less than 20, while guaranteeing the performance. However, neither of them considers memory constraint. Weaver et al. [6] uses a simplified version of TRW that can be implemented with SRAM. However, it does not guarantee the false negative probability due to collisions in the connection cache [6].

There are also other approaches focusing on network attacks with a reduced memory size [11–13,15]. Although the memory sizes are reduced significantly through refined sampling and memory management schemes, they usually use the metrics of either connection attempt rate or connection failure rate. Thus, they may not detect slow scanners.

Estan et al. propose a family of efficient bitmap algorithms for counting active flows [15]. Triggered bitmap may be useful to count the number of connection attempts for each source address with a moderate error. However, our scheme makes a decision with a rather small number of connection attempts and responses, less than 30. Since the number is usually small, the triggered bitmap is likely to be reduced to the direct bitmap. The direct bitmap can be considered as a special case of Bloom filters [20,21] with only one hash function. In our scheme, the counting accuracy is very important to guarantee the false positive and false negative probabilities, and the accuracy can be usually improved with more than one hash functions under the same memory size. Thus, we use Bloom filters to count connection attempts and responses.

3. Scanner detection scheme

We now describe the operation of the proposed scanner detection scheme in detail. Table 1 summarizes the major parameters and variables.

3.1. Detection rule

The scanner is detected based on the connection attempt success ratio of the source host. The connection attempt success ratio of a source s, $q(s)$, is defined as follows:

$$q(s) = \frac{RESPONSE(s)}{ATTEMPT(s)}, \tag{1}$$

where the $ATTEMPT(s)$ is the number of distinct IP addresses that a source s attempts to connect to and the $RESPONSE(s)$ is the number of distinct IP addresses that responded to the source s. We define two timers, $TIMER1(s)$ and $TIMER2(s)$. The scanner detection decision is made after observing n connection attempts for each source address. For the nth connection attempt, our scheme waits until $TIMER1(s)$ expires. $TIMER1(s)$ is used to allow sufficient time to count the response to the last attempt. According to [22], the median round-trip time (rtt) is measured to be lower than 450 ms by the IPMON system of Sprint even including transcontinental connections. We set the threshold for $TIMER1(s)$ to 500 ms. $TIMER2(s)$ is used to

Table 1
Major parameters and variables

n	Number of attempts required for detection decision
θ_1	Maximum connection attempt success ratio of scanners
θ_2	Minimum success ratio of benign hosts
δ_1	False negative requirement
δ_2	False positive requirement
η	Detection threshold in terms of connection success ratio
Y_1	Number of responses to a scanner monitored until the number of attempts reaches n
Y_2	Number of responses to a benign source monitored until the number of attempts reaches n
m	Bit vector size of a Bloom filter
k	Number of hash functions used for Bloom filters
p_c	Collision probability of a Bloom filter

measure the idle period, in terms of scanning activity, of each host. When the $ATTEMPT(s)$ reaches the pre-selected threshold n and the $TIMER1(s)$ expires, then we detect the source as a scanner if the following condition is satisfied:

$$q(s) \leqslant \eta, \tag{2}$$

where η is the detection threshold. Variable parameters n and η are determined according to the required false positive and false negative probabilities and these are discussed in Sections 4 and 5.

3.2. Implementation

We describe the proposed scheme in more detail including major issues related to implementation. We first describe the data structure for SherLOCK and then the connection status update procedure. We also explain the proposed eviction policy required to resolve contentions in the memory.

3.2.1. Bloom filters

Since the decision is based on the statistic of connection success ratio, the proposed scheme manages the connection status of each source address.[1] We use two Bloom filters [20,21] to manage the set of IP addresses that a given source host attempts to connect to and the set of IP addresses that responds to the given source address, respectively, with a reduced memory size. We store the set of the distinct destination addresses for a source s in an m-bit Bloom filter. We define a vector $\boldsymbol{V}$ of m bits to contain the destination address information. Each element of the vector is initially set to 0. k independent hash functions, $h_1, h_2, \ldots, h_k$, each of which has a range of $\{1, \ldots, m\}$, are used to map each destination address d into the vector $\boldsymbol{V}$.

If a packet from the source s to the destination d is observed, then the bit vector corresponding to the source address s is determined by hashing the source address s. The destination address d is then registered in the bit vector as follows. The bits at positions $h_1(d), h_2(d), \ldots, h_k(d)$ in $\boldsymbol{V}$ are set to 1. Destination d of the first packet from the source s gets registered using the Bloom filter and the subsequent packet with the same (s, d) does not get registered. The bits at positions $h_1(d), h_2(d), \ldots, h_k(d)$ represent the existence of the prior packets of (s, d). However, occasionally $h_1(d), h_2(d), \ldots, h_k(d)$ bits can be set by other destinations as the Bloom filter fills up. We define this blocking in the Bloom filter as a collision. One of the advantages of Bloom filters is that it is possible to control the probability of collision by adjusting the parameters k and m.

If the number of destinations associated with the source address s is increased, then the number of bits m should be increased to keep the collision probability at the same level. SherLOCK classifies the source s as a scanner if the number of distinct destinations reaches a pre-determined threshold n. Thus, the number of destination IP addresses associated with one source address does not exceed n and the bitmap size m can be kept constant. We analyze the effect of collisions in Bloom filters on the false positive and false negative probabilities in Sections 4 and 5.

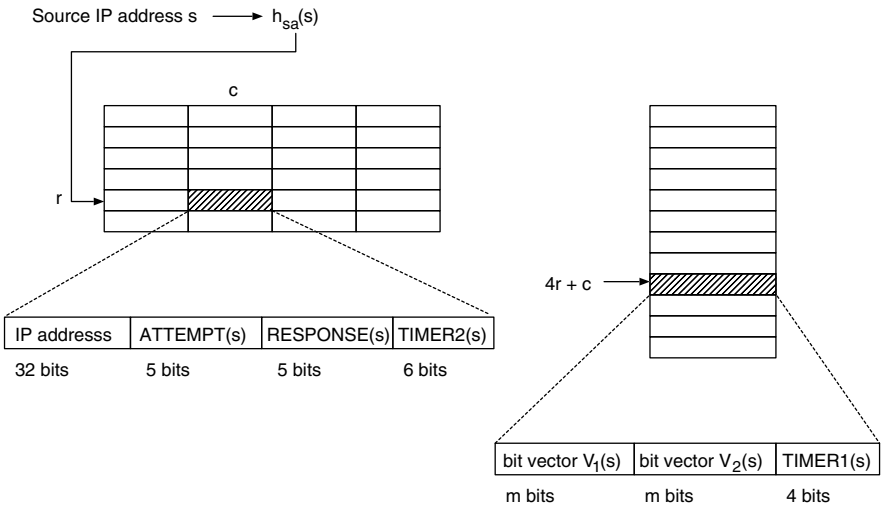
3.2.2. Data structure

We manage the connection status, especially the connection attempt status and the response status, of each source address in the tables called double connection status table. Fig. 1 shows the detailed structure of the double connection status table, which consists of two tables: primary and secondary connection status tables. The primary and secondary tables have exactly the same number of entries and the entries of the two tables are mapped as follows. If an IP address occupies an entry at the row r and column c of the primary table, then the $(4r + c)$th entry of the secondary table is also assigned to that IP address. Thus, for each source IP address, the corresponding entry in the secondary table is determined depending on the entry position in the primary table.

The primary table is a hash table that is indexed by the hash value of the source IP address. If we let h_{sa} denote this hash function, then a source address s can occupy an entry at the row of $h_{sa}(s)$. Occasionally two or more distinct source addresses may have the same hash value. In order to cope with this collision, we put four entries for each hash value[2] and we put IP address (source address) field in each entry to discriminate different source addresses which have the same hash value. In addition to the IP address field, the fields for $ATTEMPT(s)$, $RESPONSE(s)$, and $TIMER2(s)$ are maintained in the selected entry of the primary

[1] Although we consider only horizontal scanning over distinct IP addresses, our scheme can be readily extended to detect vertical scanning over the distinct port numbers.

[2] This structure can be implemented in software or in a 4-way associative cache in hardware for high-speed processing.



(a) Primary connection status table (b) Secondary connection status table

Fig. 1. The structure of the double connection status table.

table. Two bit vectors $V_1(s)$ and $V_2(s)$, and $TIMER1(s)$ are managed in the corresponding entry of the secondary table. For a source address s, a bit vector $\boldsymbol{V}_1(s)$ manages the set of distinct IP addresses that the host s attempts to connect to. Another bit vector $\boldsymbol{V}_2(s)$ manages the set of IP addresses which respond to s. If the first packet from s to d is observed, then the address d is registered in the bit vector $\boldsymbol{V}_1(s)$. If the response packet from d to s arrives afterwards, then the address d is recorded in the bit vector $\boldsymbol{V}_2(s)$.

$ATTEMPT(s)$ counts the number of attempts made by s and is equal to the number of destinations registered in $\boldsymbol{V}_1(s)$. $RESPONSE(s)$ counts the number of responses to s and is equal to the number of addresses registered in $\boldsymbol{V}_2(s)$.

In case of TCP connection request through TCP SYN packet, no response will be sent back to the source if the destination IP address is inactive. If the SYN packet is sent to an inactive port of an active host, then the host is likely to reply with TCP RST packet. If scanners usually send packets to invalid IP addresses, then we do not count TCP RST packets as a connection attempt failure. However, in case of vertical scanning, we interpret TCP RST packets as a connection attempt failure. In the existing schemes, only unanswered connection attempts are considered as the connection attempt failure. Our scheme can be extended so that TCP RST responses are counted as the connection attempt failure in case of vertical scanning detection.

Two timers, $TIMER1(s)$ and $TIMER2(s)$, are associated with the source s. If the $ATTEMPT(s)$ reaches n, then $TIMER1(s)$ is increased by 1 every 100 ms. This timer can be either synchronized or non-synchronized among different hosts. If we let t_1 denote the threshold for the $TIMER1(s)$, then the $TIMER1(s)$ value of t_1 means that at least $100 \times (t_1 - 1)$ ms has passed since the arrival time of the last nth attempt. Since we wait at least 500 ms after the arrival of the nth attempt as discussed earlier, the default value of t_1 is set to 6. If the value of $TIMER1(s)$ reaches t_1, then the decision on the scanning activity of s is made according to the rule described in the previous subsection. If the source address s is detected as a scanner, we report s as a scanner and clear every field corresponding to s. Otherwise, we delete the entry of s.

$TIMER2(s)$ measures the idle period, in terms of scanning activity, of each source address. The timer is reset to 0 whenever the value of $ATTEMPT(s)$ is increased, i.e. when a new attempt of s arrives. Otherwise, it is increased by 1 every minute. The increment period of 1 minute is determined with the following reasoning. Let N_s denote the maximum number of source addresses which can be accommodated by the connection status table and let us assume that $N_s = 10^6$. There are four entries for each address of the hash table. If we assume the hash function h_{sa} is perfectly random, then a source address in one entry of the hash table sees the next different source address which maps to the same hash table address after about $N_s/4$ more distinct source addresses according to the mean of the geometric distribution. Fraleigh et al. [22] report that the average number of active flows per minute is less

than 300,000 for all measured OC-48 links. The flow arrival rate is usually at most this rate and the arrival rate of distinct source addresses is usually lower than the flow arrival rate. If the distinct-source address arrival rate is as high as 250,000 per min, then the interarrival time of different source addresses to the same hash address is about $N_s/4/250{,}000 = 1$ min for $N_s = 10^6$. In order to discriminate the age of different source addresses mapping to the same hash address, the increment interval of $TIMER2(s)$ needs to be less than 1 min. Currently we select 1 min, but this increment interval can be adjusted according to the detailed value of N_s and the distinct-source arrival rate. Six bits are allocated to the $TIMER2(s)$ and we can count up to 63 min. $TIMER2(s)$ is used to evict less active, in terms of scanning activity, source addresses when the connection status table is overloaded. But, it has a rather low priority compared with other eviction condition as shown later. Since we need to utilize the memory space efficiently, we clear entries for the source IP addresses which have been inactive for 30 min. In other words, if the value of $TIMER2(s)$ increases up to 30 min, then the entry for s is cleared.

3.2.3. Double connection status table update procedure

The double connection status table is updated when a packet arrives. Let us assume that a packet from s to d arrives at the monitoring node. Then, the double connection status table is updated as follows:

Step 1: On the packet arrival, we check whether the packet is a response to an existing connection attempt. We check whether the address s is registered in both $V_1(d)$ and $V_2(d)$. If s exists in both $V_1(d)$ and $V_2(d)$, then the packet is a duplicate response to d. We neglect the packet and exit the routine. If s exists in $V_1(d)$, but not in $V_2(d)$, then the packet is a new response for the attempt from d. We increase $RESPONSE(d)$ by 1 and we register s in $V_2(d)$. If s does not exist in either $V_1(d)$ or $V_2(d)$, then the packet is not a response and we go to Step 2.

Step 2: We check whether the arriving packet is a new connection attempt. If it is true, we then register d in $V_1(s)$. We check whether the destination d is registered in $V_1(s)$. If d exists in $V_1(s)$, then the packet is not a new connection attempt and we exit the routine. If d does not exist in $V_1(s)$, the packet is a new attempt of s. Before registering d in $V_1(s)$, we check the value of $ATTEMPT(s)$. If the threshold, n, is reached, then we neglect the new attempt until the decision is made with the current data set. If the threshold is not reached, we register d in $V_1(s)$, increase the value of $ATTEMPT(s)$ by 1, and exit the routine.

Let us investigate the number of memory references required for an arriving packet. When a packet going from s to d is observed, at least two memory reads are required to read the connection status of d: one memory read for the line $h_{sd}(d)$ in the primary table $((32 + 5 + 5 + 6) \times 4 = 192$ bits)[3] and one memory read for the bit vectors and $TIMER1$ of d in the secondary table ($2\,m + 4$ bits). If it is not a response from s to d, two more memory reads are required to read the connection status of s and the number of bits is the same as the case for d. Two memory writes, one write to the primary table ($32 + 5 + 5 + 6 = 48$ bits) and another write to the secondary table ($2\,m + 4$ bits), are required for either s or d in case of an entry update. No memory write is required if there is no entry update. Thus, our scheme requires more memory references than Weaver et al.'s scheme [6], which requires two memory reads and two memory writes. One of the reasons for more memory references in our scheme is as follows. Weaver et al.'s scheme targets to detect either internal scanners, whose IP addresses belong to the subnet address range, or external scanners and manages address cache for either internal or external addresses, but not for both kinds of addresses simultaneously. In this case, it is enough to check the connection status of an internal (external) address in case of detecting internal (external) scanners. In order to detect scanners bidirectionally, our scheme checks the connection status of both internal and external addresses one by one. Thus, memory reads are required for both d and s in our scheme. Our scheme supports guaranteed false negative and false positive probabilities at the cost of an increased size of memory reference compared with Weaver et al.'s scheme [6]. Although more memory references are required per packet, we show

[3] The 32 bits for the IP address field can be reduced to 16 bits by using 16-bit tag instead of direct IP address as in [6]. Such reduction could lead to a collision when two different IP addresses map to the same tag value concurrently.

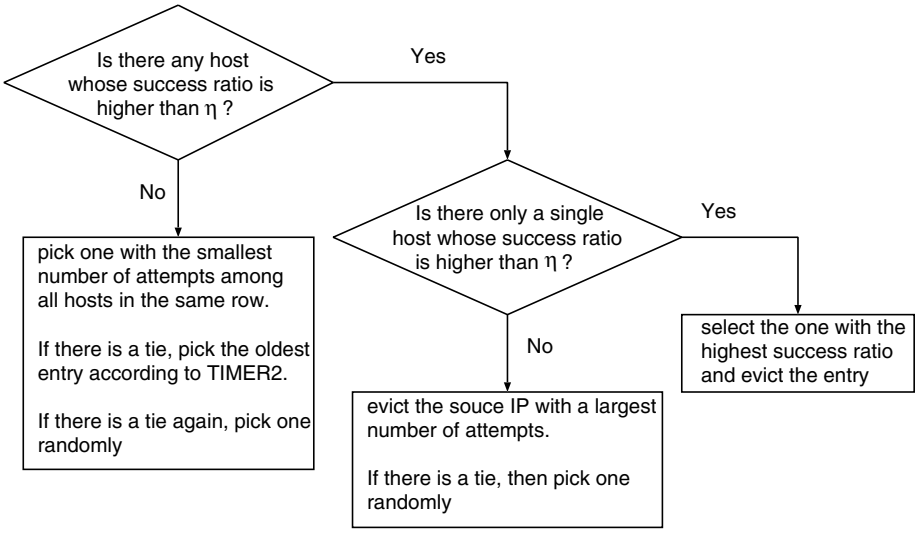


Fig. 2. Eviction policy.

that the processing overhead is moderate for the link rate of 1 Gbps in Section 6.

3.2.4. Hash table entry eviction policy

In order to resolve conflict in the hash table, especially when more than four source addresses are mapped to the same hash address, we need an eviction policy. We evict the source address with the highest connection success ratio first so that we can maintain the entries corresponding to scanners with a low connection success ratio longer in the hash table. We find that about 70% of normal hosts are making only a single connection attempt and many of them are making unsuccessful connection attempts from many NLANR traffic traces [23]. In order to discriminate these normal hosts with a low success ratio from malicious scanners with multiple connection attempts, we consider the number of connection attempts as another criterion for eviction. The number of connection attempts is important when there is no host with a high connection success ratio over the threshold η. The final eviction policy is summarized in Fig. 2.

4. Performance analysis considering contentions in the hash table and Bloom filters

In this section, we show that SherLOCK provides guaranteed performance in terms of false positive and false negative probabilities if scanners and benign hosts exhibit different behavior in terms of connection success ratio. In addition, we investigate memory requirement for Bloom filters and the hash table in order to guarantee the false positive and false negative probabilities.

Let E_1 (E_2) denote the event that a given host is a scanner (normal host). The requirements on the false positive and false negative probabilities can be expressed as

$$\text{– Low false negatives : } \Pr(\text{no detection}|E_1) < \delta_1, \tag{3}$$

$$\text{– Low false positives : } \Pr(\text{detection}|E_2) < \delta_2. \tag{4}$$

We attempt to satisfy the above performance requirement with a limited size of memory, e.g. SRAM, adequate for high-speed operation. We manage the memory size by controlling the bit vector size m and the number of rows (R) in the hash table. The number of columns in the hash table is fixed to 4.

In order to detect scanners, the scanner-related entries need to be retained in the hash table until the detection resolution. The retention of the scanner-related entries is affected by various factors and we focus on the following three major factors: aggregate request arrival rate of normal hosts (λ), the number of rows in the hash table (R), and the scanning rate (λ_1) of the scanner in attempts per second. A request packet represents the first packet between a given node pair, and none of subsequent packets between the given node pair is counted as the request packet. Thus, the request packet arrival is equivalent to the flow arrival. Since it is reported that exponential distribution fits flow inter-arrival time distributions well rather than Gamma and Weibull distribution [24], we assume that request

packets, i.e. the first packet of each flow, arrive at the monitoring node according to a Poisson process.

We first derive a condition which statistically guarantees the retention of the scanner-related entry in terms of λ, λ_1, and R. We show that slower scanners can be detected if we increase the memory size. Let us consider the case where the first request packet of a new scanner observes no space in the corresponding row of the hash table. Then, it is highly likely that at least two entries are occupied by the hosts with a high success ratio or by the hosts with a single connection attempt since about 70% of normal hosts make only a single connection attempt according to our investigation of many traces from NLANR [23]. The new entry for the scanner replaces one existing entry according to the eviction policy. If there are many hosts with a high success ratio in the same row, then the scanner-related entry is likely to be retained because the entries with a high success ratio will be evicted earlier than the scanner-related entry according to the eviction policy of Fig. 2. Let us consider the case where there is at least one host with a single connection attempt just after the appearance of the scanner entry. In this case, if the next request of the scanner arrives earlier than other distinct requests to the same row, then the entry of the scanner is retained according to the policy '*pick one with the smallest number of attempts among all hosts in the same row*' of Fig. 2. Thus, we obtain the following sufficient condition on the scanning rate λ_1 that prevents the eviction of the scanner entry.

Proposition 1. $\Pr(\text{remain}(\tilde{n}))$ *denotes the probability that the scanner entry remains in the hash table after* $\tilde{n}$ *requests. Let* ζ *and* φ *denote the ratio of the hosts with only a single connection attempt and the ratio of the hosts with a success ratio higher than* η *among the hosts with multiple connection attempts, respectively.* θ' *represents the probability that a scanner makes an unsuccessful connection attempt. Then, there exists* $\tilde{n}$ *which satisfies*

$$\Pr(\text{remain}(\tilde{n})) \geqslant 1-\varepsilon, \quad \varepsilon = (1/2)^{10} \tag{5}$$

under the following condition:

$$\lambda_1 \geqslant \lambda/R. \tag{6}$$

The smallest integer value of $\tilde{n}$, $\tilde{n}^*$, *which satisfies (5) is given as*

$$\tilde{n}^* = \left\lceil \frac{\log \varepsilon}{\log\left\{1 - 0.5\theta'(1-(1-\beta)^4 - 4\beta(1-\beta)^3)\right\}} \right\rceil,$$

where $\beta = \zeta + (1-\zeta)\varphi$ *and* $\lceil u \rceil$ *denotes the smallest integer that is larger than or equal to u. If* N_e *denotes the number of requests required to assure retention of the scanner entry in the hash table, then we have*

$$E[N_e] \leqslant 2/(\theta'(1-(1-\beta)^4 - 4\beta(1-\beta)^3)). \tag{7}$$

Proof. The proof is given in Appendix A. □

By Proposition 1, the entry corresponding to the scanner remains in the hash table with a probability of at least 99.9% (=$1-(1/2)^{10}$) after $\tilde{n}^*$ requests when $\lambda_1 = \lambda/R$. As an example, if $\zeta = 0.7$ as observed from many NLANR traces, $\theta' = 0.8$ and $\varphi = 0.5$ as a conservative estimate, then $\beta = 0.85$ and $\tilde{n}^* = 14$. However, scanner-related entries survive in the hash table with less than 3 request packets on average by (7). Let $g(\zeta)$ denote the upper-bound of $E[N_e]$ on the right-hand side of (7). Fig. 3 compares the number of requests required to guarantee the scanner-entry retention probability of $1-\varepsilon$, $\tilde{n}^*$, and the upper-bound of $E[N_e]$, $g(\zeta)$, over a range of ζ. When ζ decreases from 1 to 0.4, even though $\tilde{n}^*$ increases from 14 to 16, $g(\zeta)$ is maintained lower than 3. Thus, the scanner-related entry usually survives with a much smaller number of requests than $\tilde{n}^*$ without delaying the scanner detection process significantly and the performance of SherLOCK is not significantly affected by ζ. Regarding the detectable scanning rate, if λ is 1000 requests/s and R is 100,000, then λ_1 can be as low as 0.01 by (6). In this case, we can detect scanners whose mean request inter-arrival time is 100 s. From Proposition 1, we find that as the number of rows (R) increases, slower scanners can be detected.

The following relations are sufficient conditions for the false positive and false negative probability requirements (3) and (4):

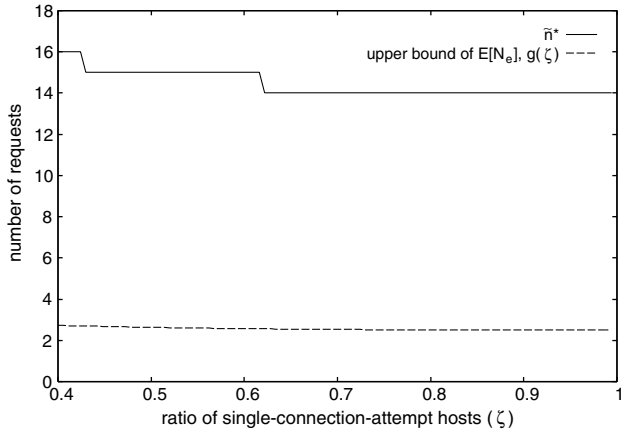


Fig. 3. Comparison of $\tilde{n}^*$ and the upper-bound of $E[N_e]$, $g(\zeta)$, for the various values of $\zeta(\theta' = 0.8, \varphi = 0.5)$.

$$\Pr(Y_1/n > \eta) < \delta_1', \quad \delta_1' = 1 - (1-\delta_1)/(1-\varepsilon), \tag{8}$$

$$\Pr(Y_2/n \leqslant \eta) < \delta_2, \tag{9}$$

where $\varepsilon = (1/2)^{10}$, and they are derived in the following way. We assume that the scanner makes at least $n + \tilde{n}$ connection attempts. $\tilde{n}$ is required to ensure the existence of the entry corresponding to the scanner with a high probability and n attempts are required to make a decision. The false positive probability of (4) can be expressed as

$$\begin{aligned}\Pr(\text{detection}|E_2) &= \Pr(\text{detection, remain}(\tilde{n})|E_2)\\ &= \Pr(\text{remain}(\tilde{n})|E_2)\\ &\quad \Pr(\text{detection}|\text{remain}(\tilde{n}), E_2)\\ &\leqslant \Pr(Y_2/n \leqslant \eta),\end{aligned} \tag{10}$$

where the last inequality is obtained because $\Pr(\text{remain}(\tilde{n})|E_2) \leqslant 1$. By (10), (9) becomes a sufficient condition for (4). The false negative probability of (3) can be expressed as

$$\begin{aligned}\Pr(\text{no detection}|E_1) &= 1 - \Pr(\text{detection}|E_1)\\ &= 1 - \Pr(\text{remain}(\tilde{n})|E_1)\\ &\quad \Pr(\text{detection}|\text{remain}(\tilde{n}), E_1)\\ &= 1 - \Pr(Y_1/n \leqslant \eta)\\ &\quad \Pr(\text{remain}(\tilde{n})|E_1)\end{aligned} \tag{11}$$

Since $\Pr(\text{remain}(\tilde{n})|E_1) \geqslant 1 - \varepsilon(\varepsilon = (1/2)^{10})$ by (5) and $\Pr(Y_1/n \leqslant \eta) = 1 - \Pr(Y_1/n > \eta)$, if the inequality (8) is satisfied, then (3) holds by (11). Thus, (8) is a sufficient condition for (3).

We now focus on how to minimize the bit vector size m in order to reduce the memory size. We also need to keep n as small as possible to quickly detect scanners with a small number of connection attempts. Hereafter, we find the set of parameters (n, η) that minimizes m while satisfying both the false positive and false negative conditions and keeping the value of n as small as possible. The minimum value of m is found as well.

4.1. Collision probabilities of Bloom filters

Let m be the number of bits allocated for the bit vector of the Bloom filter. Let k be the number of hash functions used for the Bloom filter. After inserting n' destination addresses into a bit vector of size m, the probability that a particular bit is still 0 is $(1 - 1/m)^{kn'}$. The collision in Bloom filters occurs when a new destination address observes 1 in every bit position indicated by k hash functions. The collision probability of the arriving destination addresses after the n'th registered address is then

$$(1 - (1 - 1/m)^{kn'})^k \approx (1 - e^{-kn'/m})^k. \tag{12}$$

The right-hand side of (12) is minimized when $k = \ln 2 \times m/n'$. It then becomes

$$(1/2)^k = (0.6185)^{m/n'}. \tag{13}$$

From both (12) and (13), we know that the collision probability decreases as m/n' increases. We also find that if n' is increased for a given k, then m also should be increased in order to keep the collision probability at the same level. Thus, the value of n' should be kept small to reduce the memory size.

4.2. False positive and false negative probabilities in the presence of collisions in Bloom filters

We now analyze the effect of collisions in Bloom filters on the false positive and false negative probabilities. The collision occurs when a packet corresponding to a new source/destination pair (s, d) arrives and finds that all the bits corresponding to the hash values of d are set to 1. If collisions occur in both $\boldsymbol{V}_1(s)$ and $\boldsymbol{V}_2(s)$, $ATTEMPT(s)$ does not increase since it is not considered as a new attempt. Even though a response packet from d to s is observed, $RESPONSE(s)$ does not increase because the address d already exists in the bit vector $\boldsymbol{V}_2(s)$. Thus, the source/destination pair (s, d) that experiences collisions in both $\boldsymbol{V}_1(s)$ and $\boldsymbol{V}_2(s)$ affects neither $ATTEMPT(s)$ nor $RESPONSE(s)$.

If a new source/destination pair (s, d) experiences a collision in $\boldsymbol{V}_1(s)$ but not in $\boldsymbol{V}_2(s)$, then $ATTEMPT(s)$ does not increase since it is not considered as a new attempt of s. But, if there is a response from d to s, then the $RESPONSE(s)$ increases by 1. Since the new attempt is not counted and only the response is counted in this case, these source/destination pairs tend to increase the success ratio compared to the case without collision. If s is a benign source, then the collision only in $\boldsymbol{V}_1(s)$ is likely to decrease the false positive ratio. If s is a scanner, then this kind of collision likely increases the false negative ratio. Thus, we investigate the false negative probability of a malicious source in more detail when such collisions occur.

The first address to be registered in the Bloom filter does not experience collision. As the number of the addresses registered in the Bloom filter increases, new destination addresses of s tend to experience

higher collision probability as shown in (12). After $n-1$ addresses are registered, the collision probability is highest as $(1-e^{-k(n-1)/m})^k$. p_c denotes the highest collision probability,

$$p_c = (1-e^{-k(n-1)/m})^k. \quad (14)$$

As a conservative approximation to obtain an upper bound of the false negative probability, we assume that every new connection attempt of a given source s experiences the same collision probability of p_c except the first attempt which experiences no collision. We now state the following property of binomial distribution.

Lemma 1. *Let $X(n,r)$ denote a binomial random variable with n trials and the success probability of r. Then, for r_1, r_2, r_3, and r_4 $(0 \leqslant r_1 \leqslant r_2 < \eta < r_3 \leqslant r_4 \leqslant 1)$, we have*

$$\Pr(X(n,r_1)/n > \eta) \leqslant \Pr(X(n,r_2)/n > \eta), \quad (15)$$
$$\Pr(X(n,r_4)/n \leqslant \eta) \leqslant \Pr(X(n,r_3)/n \leqslant \eta). \quad (16)$$

Proof. Since n is a fixed number, we have $\Pr(X(n,r)/n > \eta) = \Pr(X(n,r) > n\eta)$ and it can be expressed as

$$\Pr(X(n,r) > n\eta) = \sum_{j=\lfloor n\eta \rfloor + 1}^{n} \binom{n}{j} r^j (1-r)^{n-j}.$$

Since n and η are usually fixed in the above equation, we consider the right hand side term as a function of r and let $f(r)$ denote the right hand side term of the above equation. If we differentiate $f(r)$ with respect to r, then we have

$$f'(r) = \sum_{j=\lfloor n\eta \rfloor + 1}^{n} \binom{n}{j} r^{j-1}(1-r)^{n-j-1}(j-nr). \quad (17)$$

Let us consider r less than η, i.e.$r < \eta$. Since $j > n\eta$, we have $j > n\eta > nr$. Thus, $f'(r) > 0$ for $r < \eta$ by (17) and we have $f(r_1) \leqslant f(r_2)$ since $r_1 \leqslant r_2 < \eta$. Thus, (15) is proved. (16) can also be proved in a similar way. □

Due to the collision in Bloom filters, it is likely that more than n attempts are required to fill the Bloom filter designed to accommodate n IP addresses. Let A denote the total number of connection attempts of a source address s until the nth connection attempt is counted by the Bloom filter. $Y_1(j)$ and $Y_2(j)$ denote the conditional random variables $Y_1|A=j$ and $Y_2|A=j$, respectively. Let $Y_1'(n)$ denote $Y_1(n)$ when the scanner has the highest success ratio of θ_1. Let $Y_2'(n)$ denote $Y_2(n)$ when the benign source has the lowest success ratio of θ_2. Based on the data analysis in [7], we assume that $\theta_1 < \theta_2$. We assume that the response of each attempt is independent and identically distributed (i.i.d.) for the same source address. Then, we obtain the following relationship:

Proposition 2. *If a set of (n,η) satisfy (18) and (19), then the sufficient conditions for the false positive and false negative probability requirements, (8) and (9), are satisfied.*

$$\Pr(Y_1'(n) > n\eta) + 1 - (1-p_c)^{n-1} < \delta_1', \quad (18)$$
$$\Pr(Y_2'(n) \leqslant n\eta) < \delta_2, \quad (19)$$

where $Y_1'(n) \sim \text{Binomial}(n,\theta_1)$, $Y_2'(n) \sim \text{Binomial}(n,\theta_2)$, and $\delta_1' = 1-(1-\delta_1)/(1-\varepsilon)$ from (8).

Proof. The proof is given in Appendix B. □

We now solve (18) and (19) simultaneously to find the values of n and η which guarantee the false positive and false negative probabilities. In order to solve the inequality (18) explicitly in terms of n, we consider the following set of inequalities:

$$\Pr(Y_1'(n) > n\eta) < x, \quad (20)$$
$$1-(1-p_c)^{n-1} \leqslant \delta_1' - x, \quad (21)$$

where $0 < x < \delta_1'$. If we find the set of (n,η) that satisfies both (20) and (21) simultaneously, then those values of (n,η) also satisfy (18).

Thus, if we show that there exist n and η that satisfy (19)–(21) simultaneously, then it proves that SherLOCK guarantees the false positive and false negative probabilities even in the presence of collisions in Bloom filters and the contention in the hash table. It is not trivial to obtain the range of n which satisfies (19) and (20) in a closed form with the exact binomial distribution. We first consider a simple form of upper bound for binomial distribution. For a binomial random variable $X \sim \text{Binomial}(n,p)$, the following inequalities are derived using Chernoff bounds [25, Corollary 3.1.2]:

$$\Pr(X - np \geqslant \xi) \leqslant \exp(-2\xi^2/n),$$
$$\Pr(X - np \leqslant -\xi) \leqslant \exp(-2\xi^2/n).$$

Applying the above inequalities to $Y_1'(n)$ and $Y_2'(n)$, we obtain

$$\Pr(Y_1'(n) > n\eta) \leqslant \exp\{-2n(\eta-\theta_1)^2\}, \quad (22)$$
$$\Pr(Y_2'(n) \leqslant n\eta) \leqslant \exp\{-2n(\theta_2-\eta)^2\}. \quad (23)$$

If we consider (20) and (22) simultaneously, then n that satisfies

$$n > \frac{-\ln x}{2(\eta-\theta_1)^2} \tag{24}$$

also satisfies (20). If we consider (19) and (23) simultaneously, then n that satisfies

$$n > \frac{-\ln \delta_2}{2(\theta_2-\eta)^2} \tag{25}$$

also satisfies (19). Let $h_1'(\eta)$ and $h_2'(\eta)$ denote the lower bounds on the right hand side of (24) and (25), respectively. (21) leads to

$$n \leqslant \frac{\ln(1-\delta_1'+x)}{\ln(1-p_c)} + 1. \tag{26}$$

Let $h_3'(m/(n-1))$ denote the upper bound in (26), where p_c is a function of $m/(n-1)$ given by (14). The set of (n,η) that satisfies (24)–(26) simultaneously is expressed as a shaded area in Fig. 4. Let η^* and $\hat{n}$ denote the values of η and n that is on the intersection of $h_1'(\eta)$ and $h_2'(\eta)$, respectively. From Fig. 4, we find that the minimum value of n is obtained when $\eta = \eta^*$. The values of η^* and $\hat{n}$ are then

$$\eta^* = \frac{\theta_2 + \sqrt{\ln\delta_2/\ln x}\,\theta_1}{1+\sqrt{\ln\delta_2/\ln x}},$$

$$\hat{n} = -0.5\ln\delta_2/(\theta_2-\eta^*)^2.$$

The minimum value of n is determined as $n^* = \lfloor \hat{n} \rfloor + 1$, where $\lfloor u \rfloor$ denotes the largest integer that is smaller than or equal to u.

The required size of the bit vector m and the number of the hash functions k can be determined by (26). We assume that $m/(n-1)$ is maintained by adaptively changing the value of m according to n. In order to assure that $(\eta^*, \lfloor \hat{n} \rfloor + 1)$ exists in the shaded area of Fig. 4, $n = \lfloor \hat{n} \rfloor + 1$ has to satisfy (26). We substitute $\lfloor \hat{n} \rfloor + 1$ for n and solve (26) in terms of p_c. We then have

$$p_c \leqslant 1 - (1-\delta_1'+x)^{1/\lfloor \hat{n} \rfloor}. \tag{27}$$

Since the minimum value of p_c is $0.6185^{m/(n-1)}$ from (12)–(14), the range of $m/(n-1)$ is given by (27) as

$$\frac{m}{n-1} \geqslant \frac{\ln\{1-(1-\delta_1'+x)^{1/\lfloor \hat{n} \rfloor}\}}{\ln 0.6185}.$$

m is minimized when n has the minimum value of $n^* = \lfloor \hat{n} \rfloor + 1$ and the minimum value of m is determined as

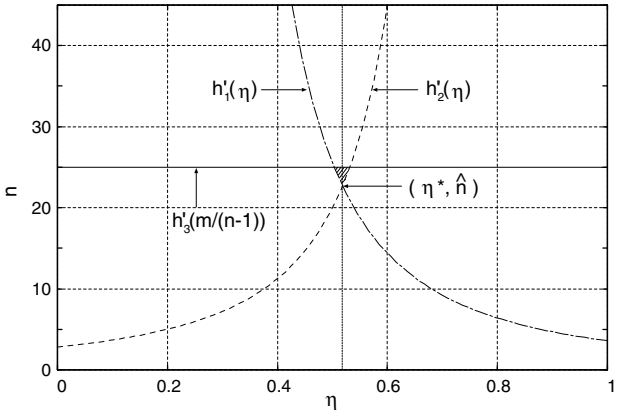


Fig. 4. Solution area for (24)–(26).

$$m^* = \left\lceil \frac{\ln\{1-(1-\delta_1'+x)^{1/\lfloor \hat{n} \rfloor}\}}{\ln 0.6185} \right\rceil \lfloor \hat{n} \rfloor. \tag{28}$$

The corresponding k is determined by $k = \ln 2 \times m^*/\lfloor \hat{n} \rfloor$ from (13) and (28) when $n' = n-1$ and $n = \lfloor \hat{n} \rfloor + 1$. We find a set of parameters (n,η,m,k) for any given constraints, δ_1 and δ_2, and we thus prove that our scheme guarantees the false positive and false negative probabilities even in the presence of collisions in Bloom filters and the contention in the hash table.

4.3. Memory requirement

We now show the required memory size for the target performance in terms of the false positive and false negative probabilities. We find the set of parameters (n,η,m,k) which minimizes the required memory for given constraints, δ_1 and δ_2.

Earlier we showed the parameter selection process for an arbitrary value of x $(0 < x < \delta_1')$ in (20) and (21). In order to find the minimum value of m, we perform the same parameter selection process for various values of x in the interval of $(0,\delta_1')$. We find the minimum bit vector size m as follows:

- For each x in $(0,\delta_1')$, we find the set of parameters (n, η, m, k) according to the given selection process. The selected parameters are the ones that minimize n and m for the given x.
- In the next step, we select the set (n,η,m,k) that minimizes m from the list of the set (n,η,m,k) obtained for various values of x in $(0,\delta_1')$.

Fig. 5 shows the values of m for various values of x $(0 < x < \delta_1')$ when $\theta_1 = 0.2$, $\theta_2 = 0.8$, and $\delta_1 = \delta_2 = 0.05$. The value of x changes from 0.0005 to

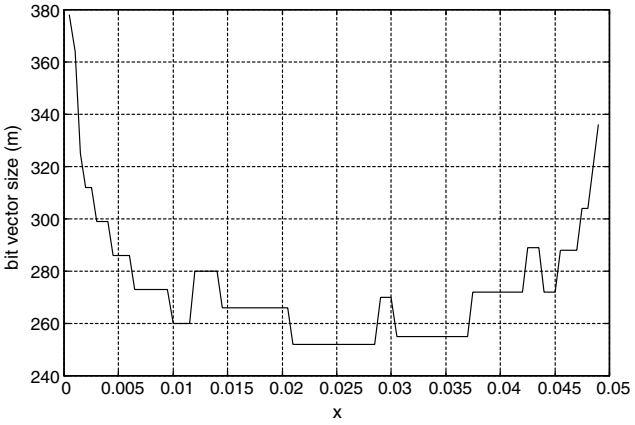


Fig. 5. Bit vector size (m) for various values of x.

0.0490 ($\approx\delta_1'$) with an increment of 0.0005. Since m takes discrete values from a very limited set of integers as shown in Fig. 5, we find the minimum value of m through an exhaustive search. In Fig. 5, the minimum value of m (252) is obtained at $x = 0.025$. Although 252 is the minimum value of m when the Chernoff bound is used, Chernoff bound is a rather loose bound. If we use the binomial distribution for $Y_1'(n)$ in (20) and $Y_2'(n)$ in (19) in order to find the minimum n that satisfies (20) and (19) simultaneously, then we find a smaller n. The optimal value of η cannot be solved explicitly using the binomial distribution. Thus, we use η obtained from Chernoff bound to compute the near-optimal n with binomial distributions assuming that the optimal η for the binomial distribution is not much different from the value obtained from Chernoff bound.

Table 2 shows the set of parameters (n, η, m, k) that minimize m under the given conditions of $(\theta_1, \theta_2, \delta_1, \delta_2)$ when Chernoff bound is used. Table 3 shows the result when the binomial distribution is used. The parameters in Table 3 guarantee the required false positive and false negative probabilities. Comparing the results in Tables 2 and 3, we find that n and m obtained from the Chernoff bound

Table 2
Parameters that minimize m when Chernoff bound is used

θ_1	θ_2	δ_1	δ_2	η	n	m	k
0.2	0.8	0.05	0.05	0.515	19	252	10
0.2	0.8	0.01	0.01	0.513	28	486	13
0.2	0.8	0.005	0.005	0.514	33	640	14
0.1	0.9	0.05	0.05	0.529	11	120	9
0.1	0.9	0.01	0.01	0.516	16	255	12
0.1	0.9	0.005	0.005	0.515	18	323	14

Table 3
Parameters that minimize m when Binomial distribution is directly used

θ_1	θ_2	δ_1	δ_2	η	n	m	k
0.2	0.8	0.05	0.05	0.515	7	78	10
0.2	0.8	0.01	0.01	0.513	13	228	14
0.2	0.8	0.005	0.005	0.514	18	306	13
0.1	0.9	0.05	0.05	0.529	3	20	7
0.1	0.9	0.01	0.01	0.516	5	80	14
0.1	0.9	0.005	0.005	0.515	7	108	13

are always upper bound of the results obtained from binomial distribution. n is kept smaller than 20 for every case when binomial distribution is directly used for $Y_1'(n)$ and $Y_2'(n)$ as shown in Table 3.

Fig. 1 shows that $52 + 2m$ $(=32 + 5 + 5 + 6 + 2m + 4)$ bits are required for each source address in the double connection status table: 48 bits in the primary table and $(2m + 4)$ bits in the secondary table. When the number of rows of the hash table is R, $(52 + 2m) \times 4 \times R = (208 + 8m)R$-bit memory is required to implement our scheme.

Let us consider the parameters obtained from binomial distribution. If R is 2×10^4, then every scenario in Table 3 can be implemented with 7 MB memory. Even when R is 10^5, if m is less than or equal to 134 bits, then those systems can be implemented with 16 MB SRAM. Table 3 shows that more strict performance can be provided in terms of the false positive and false negative probabilities as the memory size increases.

5. Sampling-based extension

In this section, we introduce a sampling technique to address a short-history-based scanner detection. If a scanner detection scheme makes decisions based on a small number of connection attempts, then it is possible to evade the detection system with only a small number of known IP addresses.

Up to this point, a scanner is assumed to select target IP addresses randomly without knowing any valid IP addresses in advance. More advanced worms and bots use a list of valid IP addresses to accelerate the spreading or evade detection based on the hit-list scanning technique [4,5]. In this section, we extend SherLOCK with a sampling technique and show that our sampling-based SherLOCK can detect scanners that know a limited number of valid IP addresses while guaranteeing false positive and false negative probabilities.

We explain the shortcoming of the basic version of SherLOCK with an example. Knowing the value of n, the scanner s with n valid IP addresses can operate without being detected by SherLOCK. Let us assume that the detection threshold η is 0.5 and n is fixed to 10. Let $d_1,\ldots,d_6$ denote the valid IP addresses that the scanner s already knows. Let $a_1,a_2,\ldots$ denote the random addresses to be scanned. If the scanner scans with six known addresses (d_1–d_6) and four random addresses from $a_1,a_2,\ldots$, then the success ratio would be at least 0.6 every time and the scanner can not be detected with the detection threshold of $\eta = 0.5$. This kind of problem also has been noted in [8] and can be common to other short-history-based schemes. We combine a sampling technique with SherLOCK to detect such scanners. We randomly sample source/destination pair with a probability of 0.1 and consider the connection attempts of only those sampled source/destination pairs. On average, less than 1 destination among d_1–d_6 would be processed. Thus, the effect of pre-acquired valid IP addresses on the connection success ratio can be controlled by the sampling probability. We use the sampling technique suggested in [12] to sample source/destination pairs. We use a uniform random hash function h_m to map (s,d) pairs to $[0,1)$. If the random number allocated to (s,d) is less than the sampling probability p_s, then the (s,d) pair is sampled and the connection status is monitored.[4]

In the performance analysis, we assume that the number of valid IP addresses known to the scanner is limited to l. We also assume that the scanner attempts to scan the known IP addresses as early as possible for a high connection success ratio and the scanner continues to scan afterwards. We note that Proposition 1 is still valid in this case because both λ and λ_1 are scaled by the same factor of the sampling probability. $\tilde{n}$ counts the sampled requests in this case.

Proposition 3. *For x_1 and x_2 $(x_1 > 0, x_2 > 0, x_1 + x_2 < \delta_1')$, if a 5-tuple of (n,η,m,k,p_s) satisfies the following relations,*

$$1-(1-p_c)^n \leqslant x_1, \tag{29}$$

$$\sum_{n\leqslant i<\alpha l}\binom{i-1}{n-1}p_s^n(1-p_s)^{i-n} \leqslant x_2, \tag{30}$$

$$\Pr(\widehat{Y}_1(n,\alpha l) > n\eta) < \delta_1' - x_1 - x_2, \tag{31}$$

$$\Pr(Y_2'(n) \leqslant n\eta) < \delta_2, \tag{32}$$

then the sufficient conditions for the false positive and false negative probability requirements, (8) and (9), are satisfied. Note that $\delta_1' = 1-(1-\delta_1)/(1-\varepsilon)$, $\varepsilon = (1/2)^{10}$, and p_s is the source/destination pair sampling probability. α is a real number larger than $(1-\theta_1)/(\theta_2-\theta_1)$ and $\widehat{Y}_1(n,\alpha l) \sim \text{Binomial}(n,\theta_1+(1-\theta_1)/\alpha)$.

Proof. The proof is given in Appendix C. □

(29), (31), (32) have the same form as (21), (20), and (19), respectively, although the success probability increases by $(1-\theta_1)/\alpha$ in case of Y_1 and the power of $1-p_c$ increases from $n-1$ to n. Thus, the solution set of (n,η,m,k) for (29), (31), (32) can be obtained in a similar way as the procedure described in Section 4. The value of α should be larger than $(1-\theta_1)/(\theta_2-\theta_1)$ in order to maintain the increased success probability of the scanner, $(\theta_1+(1-\theta_1)/\alpha)$, to be smaller than the minimum success probability of benign hosts, θ_2. If $\alpha > (1-\theta_1)/(\theta_2-\theta_1)$, then the false positive and false negative probability requirements are guaranteed. We now study the characteristics and the effect of α.

We induce the property of α from (30). Let A_s denote the number of attempts required to sample n attempts with no collision in Bloom filter, i.e. $p_c = 0$. (30) then becomes

$$\Pr(A_s < \alpha l) \leqslant x_2. \tag{33}$$

From (30), the expectation of A_s is given by $E[A_s] = n/p_s$. From the Markov inequality [26], we obtain

$$\Pr(A_s < \alpha l) \geqslant 1 - E[A_s]/(\alpha l). \tag{34}$$

Combining (33) and (34) yields $E[A_s] \geqslant \alpha\, l(1-x_2) \approx \alpha l$, since x_2 is usually negligibly small. Considering the collision in Bloom filters, $E[A] = n/(p_s(1-p_c)) \geqslant E[A_s]$ and

$$E[A] \geqslant \alpha l. \tag{35}$$

Combining $E[A_s] = n/p_s$ and $E[A_s] \geqslant \alpha l$, we get

$$p_s \leqslant n/\alpha l. \tag{36}$$

When α increases, the effective success ratio of the scanner $(\theta_1+(1-\theta_1)/\alpha)$ decreases. α thus decreases the effect of pre-acquired knowledge on valid IP addresses. As the difference between the success ratios of scanners and benign hosts increases, extended SherLOCK operates faster with a smaller number

[4] Since the sampling is based on (s,d) pair and the attempt and the response are counted at most one time for each (s,d) pair, duplicate attempts to the same valid IP address can not affect the connection success ratio of s.

of connection attempts (n) and a smaller memory size. However, the sampling may delay the detection decision. (35) shows that a large value of α tend to increase data collection time. As α increases, the memory size decreases, but the data collection time increases due to the decrease of the sampling probability in (36). As α increases, (29), (31), (32) are reduced to (21), (20), and (19), respectively, although the power of $1 - p_c$ differs by 1. Thus, the bit vector size m obtained from (21), (20), (19) becomes a lower bound of that obtained from (29), (31), and (32).

The sampling probability p_s can be computed from (30) when n is determined. If we focus only on the minimization of the bit vector size (m), then the optimal case is achieved when x_2 is negligibly small. If x_2 is very small, then p_s should be small according to (30). The small p_s leads to a long data collection time. In order to avoid too small p_s, we add the constraint that $x_2 = 0.1x_1$ to (29)–(31).

Table 4 shows the set of parameters (n, η, m, k, p_s) obtained from (29)–(32) when Chernoff bound is used to approximate Binomial distributions. Table 5 shows the set of parameters (n, η, m, k, p_s) obtained from (29)–(32) when Binomial distribution is used without approximation. Tables 4 and 5 show that the resulting parameters except p_s are not changed even though the number of known addresses l is changed. Only p_s is dependent on l as shown in (30). We find that the minimum value of m is significantly overestimated in case of Chernoff bound as opposed to Binomial distribution. As α increases from 10 to 20, n/p_s ($\approx E[A]$) increases, but the memory size does not decrease in Table 5. Thus, 10 is better choice for α than 20 in this case. Comparing the results of Table 5 with those of Table 3, we find that (n, m, k) are identical especially when $\delta_1 = \delta_2 = 0.05$. This means the minimum memory size is achieved at $\alpha = 10$ since the value of m in Table 3 is the lower bound of m obtained under the sampling.

Table 4
Parameters that minimize m when sampling technique and Chernoff bound are used ($\theta_1 = 0.2$, $\theta_2 = 0.8$)

δ_1 (=δ_2)	α	l	η	n	m	k	p_s
0.05	10	10	0.551	25	360	11	0.138838
		1000	0.551	25	360	11	0.001295
	20	10	0.539	22	294	10	0.057465
		1000	0.539	22	294	10	0.000557
0.005	10	10	0.550	43	882	15	0.263642
		1000	0.550	43	882	15	0.002342
	20	10	0.530	37	756	15	0.100002
		1000	0.530	37	756	15	0.000950

Table 5
Parameters that minimize m when sampling technique and Binomial distribution are used ($\theta_1 = 0.2$, $\theta_2 = 0.8$)

δ_1 (=δ_2)	α	l	η	n	m	k	p_s
0.05	10	10	0.551	7	78	10	0.016726
		1000	0.551	7	78	10	0.000162
	20	10	0.539	7	78	10	0.008226
		1000	0.539	7	78	10	0.000081
0.005	10	10	0.550	18	323	14	0.072460
		1000	0.550	18	323	14	0.000680
	20	10	0.530	18	323	14	0.034827
		1000	0.530	18	323	14	0.000340

If a scanner finishes probing just after scanning the known l addresses, the success ratio of the scanner would remain near 1.0. This type of scanners with a high success ratio cannot be detected based on the attempt success ratio. The extended SherLOCK can detect scanners with a large value of l only when the real success ratio of the scanner drops sufficiently lower than θ_2 by scanning random IP addresses.

6. Numerical results

In this section, we evaluate the performance of SherLOCK by experiment and compare SherLOCK with a simplified version of TRW [6] in terms of the false negative probability. The simplified version of TRW by Weaver et al. [6] is the only one that considers memory constraints among all variants of TRW [6–8]. Two Linux machines with a 2.8 GHz dual-core Intel Xeon CPU are used for the experiment. One machine generates input traffic according to the sampled packet traces and sends all the packets to the other machine that emulates a monitoring node. We use packet traces taken from NLANR archive [23] as the base traffic. We also inject packet traces of 1000 normal hosts with a success ratio of 0.8 (θ_2) and 1000 malicious hosts with a success ratio of 0.2 (θ_1). We model a successful attempt with a single pair of bidirectional packets. The response time of the response packet is distributed uniformly in the interval of (0, 450) ms. The interval between successive attempts is modeled with an exponential distribution.

The base traces from NLANR are described in Table 6. All the traces are sampled from the links with a link rate of 1 Gbps. Instantaneous traffic rate is especially high for trace 3, but there was no packet loss due to our algorithm complexity.

We limited the number of rows of the hash table, R, to 20,000 in order to test the scenario where the

Table 6
Description of NLANR traces

Trace	Duration (s)	No. packets	No. distinct sources	No. distinct flows
1	21,600	23,418,634	81,096	192,129
2	10,904	22,599,344	69,262	199,071
3	900	8,967,650	368,698	971,864

hash table is overloaded. The total number of entries in the hash table is 80,000. The double connection status table is overloaded with over 80,000 distinct sources of traces 1 and 3. MD5 is used for hash functions. Table 7 shows the experiment results for several scenarios. For the scanner without any pre-acquired IP addresses, i.e. $l = 0$, we use the parameters given in Table 3. For the scanner with 10 pre-acquired IP addresses, i.e. $l = 10$, the sampling-based extended SherLOCK is applied with the parameters corresponding to $\alpha = 10$ in Table 5.

In traces 1–3, the arrival rates of distinct flows (or requests) between distinct source and destination (s–d) pairs are up to 4000 flows/s for the first 1 s. As the observation interval increases, the arrival rate of the new flows decreases due to the increase of the recurrent IP addresses. The long-term arrival rate of distinct flows is less than 100 flows/sec for traces 1 and 2. From Proposition 1, when $\lambda = 100$ and $R = 20{,}000$ the scanner-related entry remains in the hash table with a probability of 99.9% if $\lambda_1 \geqslant 0.005$, that is, if the average inter-arrival time of distinct requests from the scanner is less than 200 s. The mean inter-arrival time of distinct requests of the scanners ($1/\lambda_1$) is set to 100 s for traces 1 and 2.

In case of trace 3, the arrival rate of request packets (λ) is 1080 requests/s and the mean inter-arrival time ($1/\lambda_1$) of distinct requests from each scanner is set to 20 s. According to Proposition 1, if the number of the rows of the hash table R is at least 21,600, then the existence of scanner-related entries is guaranteed with the probability of at least 99.9%. The lower bound of (5) in Proposition 1 is not a tight bound. We find that scanner-related entries are retained with the probability of 99.9% with a smaller value of R, especially for $R = 20{,}000$. Thus, the false positive and false negative probabilities are guaranteed for trace 3 as shown in Table 7. If sampling is not used, each scanner is sending $(n + \tilde{n}^*)$ packets in each scenario.

Table 7 shows that the double connection status table is heavily loaded in scenarios 1–6 through the number of evictions. Both the false positive and false negative probabilities are guaranteed in those scenarios even though the scanning rate λ_1 is as low as 0.01. The number of evictions can be larger than the number of distinct source IP addresses as one source address can be evicted multiple times if that host is making connections to many other hosts over a long time period. In scenarios 5 and 6, even when the instantaneous traffic rate goes up to near 1 Gbps, there is no packet loss due to the overhead of our algorithm and both of the false positive and false negative probabilities are guaranteed. When we compute the false positives, we include benign hosts in NLANR traces as well as the injected normal hosts. Since most of the IP addresses in the real trace do not make many connection attempts to distinct hosts, they are not usually detected and are counted as normal hosts.

In scenarios 7–12 with scanners using 10 pre-acquired IP addresses, we use the sampling-based SherLOCK described in Section 5. The environment is similar to that of scenarios $1 \sim 6$, but each

Table 7
Experiment results for the proposed scheme ($\theta_1 = 0.2$, $\theta_2 = 0.8$)

Scenario	Known addresses (l)	Trace	δ_1/δ_2	False positives	False negatives	No. evictions
1	0	1	0.05/0.05	7.80e−4	0.001	98,200
2	0	1	0.005/0.005	6.09e−5	0.003	91,404
3	0	2	0.05/0.05	9.68e−4	0.001	58,898
4	0	2	0.005/0.005	5.69e−5	0	55,255
5	0	3	0.05/0.05	2.41e−4	0	367,388
6	0	3	0.005/0.005	1.08e−5	0.003	367,969
7	10	1	0.05/0.05	9.74e−4	0.001	466
8	10	1	0.005/0.005	4.87e−5	0.002	3,233
9	10	2	0.05/0.05	1.32e−3	0.001	435
10	10	2	0.005/0.005	1.28e−4	0.003	2,690
11	10	3	0.05/0.05	2.73e−4	0	1,072
12	10	3	0.005/0.005	1.35e−5	0.001	12,899

scanner sends more, $(n+\tilde{n}^*)/p_s$, packets to fill Bloom filters under the sampling. We observe that both the false positive and false negative probabilities are guaranteed in these scenarios. Since sampling reduces the load to the memory, the number of evictions is decreased from the non-sampling case. The number of evictions of scenarios 8 and 10 are higher than those of scenarios 7 and 9. The reason is that the sampling probability is higher for 8 and 10 than for 7 and 9 in Table 5.

We now compare SherLOCK with the simplified version of TRW [6] in terms of the false negative probability. We use the default values used in [6] for the parameters of the simplified TRW. The number of entries in the connection cache is fixed to 1 million. Since the address cache also has 1 million entries, 5 MB of memory is required to implement this simplified TRW (1 MB for the connection cache, and 4 MB for the address cache). We use trace 3 to compare both schemes in an environment where the memory is heavily loaded. Especially in scenario 5 of Table 7, the bit-vector size m for SherLOCK is 78 bits according to Table 3 and requires 2.1 MB of memory. The false positive probability is well guaranteed by both schemes.

Fig. 6 compares the performance of SherLOCK and the simplified TRW in terms of the false negative probability for scenario 5. The simplified TRW is evaluated for various values of the detection threshold T between 5 and 10, which were used in [6]. While SherLOCK satisfies the false negative requirement of 0.05, the simplified TRW does not guarantee the false negative probability. In the simplified TRW, each entry of the connection cache is allowed to be shared by multiple connections. If 50% of the connection cache is occupied by existing connections, then a new attempt of a scanner may not be counted by a probability of up to 50% since the corresponding entry might have been used by another connection. Occasionally some packets belonging to other connections may increase the response ratio of scanners due to the entry merging. In trace 3, the number of flows is very large near the total number of entries (1 million) in the connection cache. These two effects can increase the response ratio of scanners leading to violation of the false negative requirement. As the detection threshold T increases, less number of scanners would be detected. Thus, the false negative probability increases with respect to the detection threshold T in case of the simplified TRW.

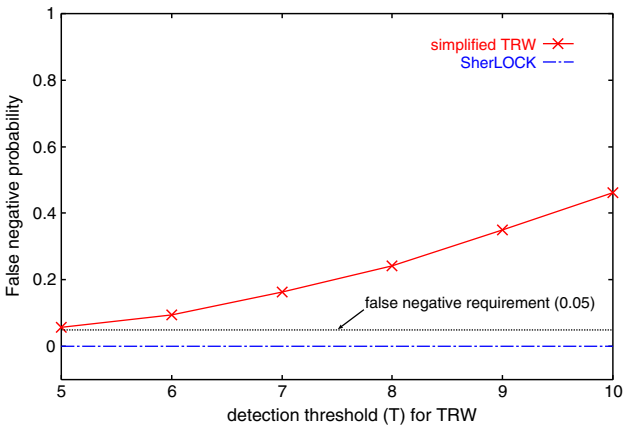


Fig. 6. Comparison of SherLOCK and the simplified version of TRW in terms of the false negative probability.

In contrast to the simplified version of TRW, if the three parameters λ, λ_1, and R, satisfy the condition (6) of Proposition 1, then the false positive and false negative probabilities of SherLOCK are guaranteed based on Proposition 1.

7. Conclusions

Time-window-based scanning detection schemes have two major drawbacks. First, slow scanners can easily evade those systems. Second, it is very difficult to find an optimal threshold since a low threshold can increase false positives and a high threshold allows slow scanners to go undetected. We propose SherLOCK, a scanner detector based on the connection attempt success ratio. A few schemes have been proposed based on the connection attempt success ratio, but there is no scheme that can guarantee the false positive and false negative probabilities in a high-speed link with a limited high-speed memory. SherLOCK can detect scanners with guaranteed false positive and false negative probabilities using a limited-size memory. We analyze two types of memory conflict in the proposed scheme: the contention in the hash table and the contention in the Bloom filter used to manage the connection status of each source address. SherLOCK can detect slow scanners for the given traffic load and the memory size. We present a simple way to compute the various parameters in SherLOCK to meet the target performance. Through experiment, we demonstrate that SherLOCK can detect very slow scanners with only 16 MB memory while guaranteeing the false positive and false negative probabilities. We expect that much slower scanners can be detected by SherLOCK if the memory constraint can be relaxed. The sam-

pling-based extended SherLOCK overcomes the shortcomings of short-history-based detection schemes and detects more intelligent scanners that use pre-acquired valid IP addresses while guaranteeing false positive and false negative probabilities.

Acknowledgements

This work was supported in part by US Army Research Office (DAAD19-02-1-0389), by the Korea Research Foundation Grant funded by Korea Government (KRF-2004-214-D00377), and by the MIC (Ministry of Information and Communication), Korea, under the ITRC support program by the IITA (IITA-2006-(C1090-0603-0002)).

Appendix A

Proof of Proposition 1

If the following three conditions are satisfied, then the scanner entry is retained in the memory by the eviction policy of Fig. 2:

- *E_con1:* When the first request of the scanner arrives at the hash table, at least two entries are occupied by the hosts which achieve a high success ratio or make only a single connection attempt.
- *E_con2:* The first request of the scanner is unsuccessful.
- *E_con3:* The second request of the scanner arrives earlier than the other distinct requests to the same row.

Thus, we can obtain the following lower bound for the probability that the scanner-related entry is retained in the memory.

$$\Pr(\text{non-eviction of the scanner entry}) \geqslant \Pr(E\_con1)\Pr(E\_con2)\Pr(E\_con3), \tag{A1}$$

where we assume that the events *E_con1*, *E_con2* and *E_con3* are independent. If we let β denote the probability that a specific entry in a row is occupied by a host which has a success ratio higher than η or by a host which makes only a single connection attempt, then $\beta = \zeta + (1-\zeta)\varphi$. $\Pr(E\_con1)$ is expressed in terms of β as

$$\begin{aligned}\Pr(E\_con1) &= 1 - \binom{4}{0}\beta^0(1-\beta)^4 \\ &\quad - \binom{4}{1}\beta^1(1-\beta)^3 \\ &= 1 - (1-\beta)^4 - 4\beta(1-\beta)^3. \end{aligned}\tag{A2}$$

From the definition of θ', we have

$$\Pr(E\_con2) = \theta'. \tag{A3}$$

Let X_1 and X_2 denote the inter-arrival time of distinct requests of the given scanner and the inter-arrival time of other requests arriving at the same row, respectively. Then, we have $X_1 \sim \exp(\lambda_1)$. When the aggregate request arrival process for the whole hash table is a Poisson process with a parameter λ, the arrival process to a particular row becomes a Poisson process with the parameter λ/R by Theorem 4 of [27, p. 74]. Then, we have

$$X_1 \sim \exp(\lambda_1), \quad X_2 \sim \exp(\lambda/R),$$

and $\Pr(E\_con3)$ is expressed as

$$\Pr(E\_con3) = \Pr(X_1 \leqslant X_2). \tag{A4}$$

Since both X_1 and X_2 have exponential distributions, we have

$$\Pr(X_1 \leqslant X_2) = \frac{\lambda_1}{\lambda_1 + \lambda/R}. \tag{A5}$$

From (A1)–(A5), we obtain an upper bound of the eviction probability as

$$\begin{aligned}\Pr(1 \text{ time eviction}) &\leqslant 1 - \Pr(E\_con1)\Pr(E\_con2)\Pr(E\_con3) \\ &= 1 - \frac{\lambda_1}{\lambda_1 + \lambda/R}\theta'(1 - (1-\beta)^4 \\ &\quad - 4\beta(1-\beta)^3). \end{aligned}\tag{A6}$$

We assume that an eviction of the entry of a specific source address is independent of other evictions of the same address. Then, the probability $\Pr(\text{remain}(\tilde{n}))$ that the scanner entry remains in the hash table after $\tilde{n}$ requests is bounded as

$$\Pr(\text{remain}(\tilde{n})) \geqslant 1 - \left(1 - \frac{\lambda_1}{\lambda_1 + \lambda/R}\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3)\right)^{\tilde{n}}.$$

If $\lambda_1 \geqslant \lambda/R$, then $\lambda_1/(\lambda_1 + \lambda/R) \geqslant 1/2$ and we have

$$\Pr(\text{remain}(\tilde{n})) \geqslant 1 - \left(1 - 0.5\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3)\right)^{\tilde{n}}. \quad \text{(A7)}$$

If $\tilde{n}$ satisfies the following relation:

$$1 - (1 - 0.5\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3))^{\tilde{n}} \geqslant 1 - \varepsilon,$$

then those $\tilde{n}$'s also satisfy (5) by Eq. (A7). The smallest integer value of $\tilde{n}, \tilde{n}^*$, can be calculated by solving the above inequality and the value is obtained as

$$\tilde{n}^* = \left\lceil \frac{\log \varepsilon}{\log\left\{1 - 0.5\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3)\right\}} \right\rceil.$$

Let p_e and N_e denote the probability that an entry induced by a request of a given scanner is not retained and the number of request packets from the scanner required to assure retention of the scanner entry in the hash table, respectively. Then, from the expectation of the geometric distribution, we have

$$E[N_e] = 1/(1 - p_e). \quad \text{(A8)}$$

Since p_e is equal to Pr(1 time eviction) of (A6), from (A6) and (A8) we have

$$E[N_e] \leqslant \frac{1 + (\lambda/\lambda_1 R)}{\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3)}.$$

When $\lambda_1 \geqslant \lambda/R$, we have $E[N_e] \leqslant 2/(\theta'(1 - (1-\beta)^4 - 4\beta(1-\beta)^3))$.

Appendix B

Proof of Proposition 2

A is the total number of connection attempts of a source address s until the nth connection attempt is counted by the Bloom filter. Since the packet arriving at an empty Bloom filter does not experience collision, the distribution of A is given as

$$\Pr(A = n + i) = \binom{n+i-2}{n-2} (1 - p_c)^{n-1} p_c^i, \quad \text{(B1)}$$

where $i = 0, 1, 2, \ldots$. Let us investigate the distribution of $Y_1(j)$, i.e. $Y_1|A = j$. In real case, the value of *RESPONSE* may not reach j due to collisions in V_2. But, we assume that there is no collision in V_2 and the value of *RESPONSE* can reach up to j as a conservative approximation in terms of false negative probability. This assumption is conservative because a large value of *RESPONSE* implies high false negatives for scanners. Let $Z_1(i)$ be a random variable that represents the result of the ith connection attempt of a malicious source. $Z_1(i)$ is 1 if the ith attempt is a success, and 0 if the attempt is a failure. Then, $Y_1(j)$ can be expresses in terms of $Z_1(i)$ as

$$Y_1(j) = \sum_{i=1}^{j} Z_1(i), \quad j \geqslant n.$$

Since we assume that the random variables $Z_1(i)$'s $(i = 1, 2, \ldots)$ are independent and identically distributed (i.i.d.), $Y_1(j)$ has a binomial distribution.

In order to discriminate the hosts with the success ratio of θ_1 from the hosts with the success ratio of θ_2, we need to put the detection threshold η between them. Thus, we assume $\theta_1 < \eta < \theta_2$. The false negative probability of a scanner s_1 under the condition that the scanner-related entry is not evicted can be expressed as

$$\begin{aligned}\Pr(Y_1/n > \eta) &= \sum_{j=n}^{\infty} \Pr(Y_1/n > \eta | A = j) \Pr(A = j) \\ &\leqslant \Pr(Y_1/n > \eta | A = n) \Pr(A = n) \\ &\quad + 1 - \Pr(A = n). \end{aligned} \quad \text{(B2)}$$

By the definition of $Y_1(n)$, we have $\Pr(Y_1/n > \eta | A = n) = \Pr(Y_1(n)/n > \eta)$. Since $\Pr(A = n) = (1 - p_c)^{n-1}$ by (B1) and (B2) can be changed into

$$\begin{aligned}\Pr(Y_1/n > \eta) &\leqslant \Pr(Y_1(n)/n > \eta)(1 - p_c)^{n-1} \\ &\quad + 1 - (1 - p_c)^{n-1}.\end{aligned}$$

Since we can obtain $\Pr(Y_1'(n) > n\eta) \geqslant \Pr(Y_1(n) > n\eta)$ from Lemma 1 and $(1 - p_c)^{n-1} \leqslant 1$ on the right hand side of the above inequality, we have

$$\Pr(Y_1/n > \eta) \leqslant \Pr(Y_1'(n) > n\eta) + 1 - (1 - p_c)^{n-1}. \quad \text{(B3)}$$

Thus, if we find the set of (n, η) which satisfies (18), then those set of (n, η) also satisfies the sufficient condition for the false negative probability (8) by (B3).

Y_2 is the number of connection successes of a benign host observed until n attempts are counted by the Bloom filter. Then, we have

$$\Pr\left(\frac{Y_2}{n} \leqslant \eta\right) = \sum_{j=n}^{\infty} \Pr\left(\frac{Y_2}{n} \leqslant \eta \middle| A = j\right) \Pr(A = j). \quad \text{(B4)}$$

In this case, we do not assume that there is no collision in V_2 since this assumption leads to decrease of the above false positive probability optimistically. Let R denote the maximum number the *RESPONSE* value can reach in this case. We consider only the attempts which can contribute to the *RESPONSE* value without collision in the Bloom filter V_2 in Y_2. Then, we have

$$\Pr\left(\frac{Y_2}{n} \leqslant \eta \middle| A = j\right) = \sum_{i=n}^{j} \Pr\left(\frac{Y_2}{n} \leqslant \eta \middle| R = i, A = j\right) \Pr(R = i | A = j). \tag{B5}$$

Since we assumed that the outcomes of the attempts of a benign host are i.i.d., we have

$$\Pr(Y_2 \leqslant n\eta | R = i, A = j) = \Pr(\widetilde{Y}_2(i) \leqslant n\eta), \tag{B6}$$

where $\widetilde{Y}_2(i)$ denotes $Y_2|R = i,\ A = j$ and $\widetilde{Y}_2(i) \sim \text{Binomial}(i, p_2)$, where p_2 is the success probability of the benign source. We can easily show that $\Pr(\widetilde{Y}_2(i) \leqslant n\eta)$ is a non-increasing function with respect to i when $i \geqslant n$ and $p_2 \geqslant \theta_2 > \eta$, that is,

$$\Pr(\widetilde{Y}_2(n) \leqslant n\eta) \geqslant \Pr(\widetilde{Y}_2(i) \leqslant n\eta), \quad i \geqslant n, \tag{B7}$$

Then, from (B4)–(B7), we can obtain the following bound:

$$\Pr\left(\frac{Y_2}{n} \leqslant \eta\right) \leqslant \Pr\left(\widetilde{Y}_2(n) \leqslant n\eta\right). \tag{B8}$$

When $A = n$, there is no collision in the Bloom filter and R is also equal to n. Thus, we have $\Pr(\widetilde{Y}_2(n) \leqslant n\eta) = \Pr(Y_2(n) \leqslant n\eta)$ from (B5), where $Y_2(n)$ denotes $Y_2|A = n$, and (B8) can be changed into

$$\Pr\left(\frac{Y_2}{n} \leqslant \eta\right) \leqslant \Pr(Y_2(n) \leqslant n\eta). \tag{B9}$$

Since $\Pr(Y_2'(n) \leqslant n\eta) \geqslant \Pr(Y_2(n) \leqslant n\eta)$ by Lemma 1, from (B9) we have

$$\Pr(Y_2/n \leqslant \eta) \leqslant \Pr(Y_2'(n) \leqslant n\eta). \tag{B10}$$

By (B10), the sufficient condition for the false positive probability (9) is satisfied when (19) holds.

Appendix C

Proof of Proposition 3

As the scanner probes more known IP addresses before the sampling is over, the monitored success probability is likely to increase. Thus, the connection success ratio can be maximized when the scanner probes as many known IP addresses as possible before the sampling is over. The false negative probability is also maximized when the scanner accesses as many valid IP addresses as possible. If we can guarantee the false negative probability in this worst case, the false negative probability should be guaranteed in other cases, i.e. when the scanner probes less number of known IP addresses, too. Thus, we assume that scanners attempt to scan known IP addresses as early as possible for a high connection success ratio.

Let A denote the number of connection attempts made until the *ATTEMPT* value of a given source address reaches n. Let B denote the number of attempts sampled until the *ATTEMPT* value of the source address reaches n.

As a conservative approximation we assume that every sampled attempt or (s, d) pair experience the collision probability of p_c given by (14) in Bloom filters. An attempt can be registered into a Bloom filter if it is sampled and does not experience collision in Bloom filters. Thus, the distribution of A is given by

$$\Pr(A = i) = \binom{i-1}{n-1} (p_s(1-p_c))^n (1 - p_s(1-p_c))^{i-n}, \tag{C1}$$

where $i = n,\ n+1,\ \ldots$. By conditioning on A, we can obtain the following relation regarding the false negative probability:

$$\begin{aligned}\Pr\left(\frac{Y_1}{n} > \eta\right) &= \sum_{i=n}^{\infty} \left(\frac{Y_1}{n} > \eta \middle| A = i\right) \Pr(A = i) \\ &\leqslant \Pr(A < \alpha l) + \sum_{i \geqslant \alpha l} \Pr\left(\frac{Y_1}{n} > \eta \middle| A = i\right) \Pr(A = i).\end{aligned} \tag{C2}$$

By conditioning on B, we can obtain

$$\Pr\left(\frac{Y_1}{n} > \eta \middle| A = i\right) = \sum_{j=n}^{i} \Pr\left(\frac{Y_1}{n} > \eta \middle| B = j, A = i\right) \Pr(B = j | A = i). \tag{C3}$$

In order to obtain an upper bound of the false negative probability, if we assume that the responses of the sampled attempts does not experience collisions in V_2 of the corresponding source address as a conservative approximation, then the *RESPONSE* value can reach up to j when $B = j$. Let us consider the case of $i > l$ since $i \geqslant \alpha l$ in (C2) and $\alpha > 1$ by the assumption. Since we assumed that the scanner

attempts to scan l known IP addresses first, those attempts to l known IP addresses are included in the total number of attempts i. If the sampling probability is fixed to p_s, then whether a specific attempt among i attempts is sampled or not is independent of sampling of other attempts by our sampling scheme. Then, the probability that a sampled attempt is from the list of known IP addresses of a scanner is l/i. Before evaluating (C3), let us investigate the distribution of $Y_1|B=n,\ A=i$. Let C denote the number of attempts to known addresses among B sampled attempts. Then, the distribution of $Y_1|B=n,\ A=i$ can be expressed as

$$\begin{aligned}\Pr(Y_1=z|B=n,A=i)&=\sum_{k=0}^{n}\Pr(Y_1=z|C\\&=k,B=n,A=i)\Pr(C\\&=k|B=n,A=i).\end{aligned}\tag{C4}$$

Although $\Pr(C=k|B=n,\ A=i)$ is given as a hypergeometric distribution of $\binom{l}{k}\binom{i-l}{n-k}\Big/\binom{l}{k}$, if i ($\geqslant \alpha l$) is much larger than n, then it can be approximated by the binomial distribution [28] as

$$\Pr(C=k|B=n,A=i)\approx\binom{n}{k}\left(\frac{l}{i}\right)^{k}\left(1-\frac{l}{i}\right)^{n-k}.\tag{C5}$$

If we assume that the attempt to any known addresses is always successful, then we have

$$\Pr(Y_1=z|C=k,B=n,A=i)=\begin{cases}\binom{n-k}{z-k}p_1^{z-k}(1-p_1)^{n-z}, & \text{for } z\geqslant k,\\ 0, & \text{for } z<k,\end{cases}\tag{C6}$$

where p_1 is the attempt success probability of a scanner for random IP addresses and $p_1\leqslant\theta_1$. Combining (C4)–(C6) yields

$$\Pr(Y_1=z|B=n,A=i)=\binom{n}{z}(\hat{p}_1(i))^{z}(1-\hat{p}_1(i))^{n-z},\tag{C7}$$

where $\hat{p}_1(i)=p_1+(1-p_1)l/i$. Let $Y_1^*(n,i)$ denote $Y_1|B=n,\ A=i$, then $Y_1^*(n,i)\sim\text{Binomial}(n,\hat{p}_1(i))$ by (C7).

In order to evaluate (C3), we now calculate $\Pr(B=n|A=i)$. Since $\Pr(B=n|A=i)=\Pr(B=n,\ A=i)/\Pr(A=i)$ and $\Pr(A=i)$ is given by (C1), we need to evaluate $\Pr(B=n,\ A=i)$. Since B and A are reflecting sampled attempts and total attempts, respectively, until n distinct attempts are registered in the Bloom filter V_1, the event of $B=n$ and $A=i$ means all of the n sampled attempts are registered in V_1 without collision and all other $i-n$ attempts are not sampled. In addition, the final attempt is always sampled and registered without collision by the definition of B and A. Thus, $\Pr(B=n,\ A=i)$ can be evaluated as

$$\Pr(B=n,A=i)=\binom{i-1}{n-1}(p_s(1-p_c))^{n}(1-p_s)^{i-n}.\tag{C8}$$

From (C1), (C8), and the definition of conditional probability, we have

$$\Pr(B=n|A=i)=\left(\frac{1-p_s}{1-p_s(1-p_c)}\right)^{i-n}.\tag{C9}$$

From (C3) and (C9), we can obtain

$$\Pr\left(\frac{Y_1}{n}>\eta\middle|A=i\right)\leqslant\Pr\left(\frac{Y_1}{n}>\eta\middle|B=n,A=i\right)+1-\left(\frac{1-p_s}{1-p_s(1-p_c)}\right)^{i-n}.\tag{C10}$$

We need to select α larger than $(1-\theta_1)/(\theta_2-\theta_1)$ so that the increased success probability of the scanner $\hat{p}_1(i)$ $(=p_1+(1-p_1)l/i)$ cannot be larger than the minimum success probability of a normal host θ_2. Since $Y_1^*(n,i)\sim\text{Binomial}(n,\hat{p}_1(i))$ and $\hat{p}_1(i)$ decreases as i increases, by Lemma 1 we can obtain

$$\Pr\left(\frac{Y_1}{n}>\eta\middle|B=n,A=i\right)\geqslant\Pr\left(\frac{Y_1}{n}>\eta\middle|B=n,A=k\right)\quad\text{for } k\geqslant i.\tag{C11}$$

From (C2), (C10) and (C11), the following relation can be derived

$$\Pr\left(\frac{Y_1}{n}>\eta\right)\leqslant 1-(1-p_c)^{n}+\sum_{i=n}^{\alpha l-1}\binom{i-1}{n-1}p_s^{n}(1-p_s)^{i-n}+\Pr\left(\frac{Y_1}{n}>\eta\middle|B=n,A=\alpha l\right).\tag{C12}$$

We assumed that α is selected such that αl is an integer. In the above relation, $Y_1|_{B=n,A=\alpha l} = Y_1^*(n,\alpha l) \sim \text{Binomial}(n, p_1 + (1-p_1)/\alpha)$. If $\widehat{Y}_1(n,\alpha l)$ denote $Y_1^*(n,\alpha l)$ especially when p_1 has the maximum value of θ_1, then $\theta_1 + (1-\theta_1)/\alpha \geqslant p_1 + (1-p_1)/\alpha$ and by Lemma 1, we have

$$\Pr(\widehat{Y}_1(n,\alpha l) > n\eta) \geqslant \Pr(Y_1^*(n,\alpha l) > n\eta). \tag{C13}$$

Combining (C12) and (C13) yields

$$\Pr\left(\frac{Y_1}{n} > \eta\right) \leqslant 1 - (1-p_c)^n + \sum_{i=n}^{\alpha l-1} \binom{i-1}{n-1} p_s^n (1-p_s)^{i-n} + \Pr(\widehat{Y}_1(n,\alpha l) > n\eta). \tag{C14}$$

From (8) and (C14), we can know that if (29)–(31) are satisfied, then the sufficient condition for the false negative probability requirement (8) is satisfied.

The false positive probability is not affected by the known IP addresses of scanners since it is determined by the behavior of benign hosts. We can show the following relation is valid in the same way as that used for Y_2 in Appendix B although we need to condition one more time about the sampled attempts B between A and R:

$$\Pr\left(\frac{Y_2}{n} \leqslant \eta\right) \leqslant \Pr(Y_2'(n) \leqslant n\eta). \tag{C15}$$

Thus, if (32) is satisfied, then the sufficient condition for the false positive probability requirement (9) is satisfied by (C15).

References

[1] CERT Coordination Center, <http://www.cert.org/stats/cert_stats.html>.

[2] T. Holz, A short visit to the bot zoo, IEEE Security and Privacy 3 (3) (2005) 76–79.

[3] D. Moore et al., Inside the Slammer worm, IEEE Security and Privacy 1 (2003) 33–39.

[4] S. Staniford, V. Paxson, N. Weaver, How to own the Internet in your spare time, in: Proceedings of the 11th USENIX Security Symposium, USENIX, August 2002.

[5] J. Wu, S. Vangala, L. Gao, K. Kwiat, An effective architecture and algorithm for detecting worms with various scan techniques, in: Proceedings of the Network and Distributed System Security Symposium, 2004.

[6] N. Weaver, S. Staniford, V. Paxson, Very fast containment of scanning worms, in: Proceedings of the 13th Usenix Security Conference, 2004.

[7] J. Jung, V. Paxson, A.W. Berger, H. Balakrishnan, Fast portscan detection using sequential hypothesis testing, in: Proceedings of the IEEE Symposium on Security and Privacy, May 2004.

[8] S.E. Schechter, J. Jung, A.W. Berger, Fast detection of scanning worm infections, in: Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), September 2004.

[9] L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, D. Wolber, A network security monitor, in: Proceedings of the IEEE Symposium on Research in Security and Privacy, 1990, pp. 296–304.

[10] M. Roesch, Snort: lightweight intrusion detection for networks, in: Proceedings of the 13th Conference on Systems Administration (LISA-99), November 7–12, 1999, USENIX Association, pp. 229–238.

[11] Q. Zhao, A. Kumar, J. Xu, Joint data streaming and sampling techniques for detection of super sources and destinations, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC), October 2005.

[12] S. Venkataraman, D. Song, P.B. Gibbons, A. Blum, New streaming algorithms for fast detection of superspreaders, in: Proceedings of the Network and Distributed Systems Security Symposium, February 2005.

[13] R.R. Kompella, S. Singh, G. Varghese, On scalable attack detection in the network, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC), October 2004.

[14] S. Robertson, E.V. Siegel, M. Miller, S.J. Stolfo, Surveillance detection in high bandwidth environments, in: Proceedings of the 2003 DARPA DISCEX III Conference, 22–24 April, 2003, IEEE Press, Washington, DC, 2003, pp. 130–139.

[15] C. Estan, G. Varghese, M. Fisk, Bitmap algorithms for counting active flows on high speed links, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC), October 2003.

[16] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer Networks 31 (23–24) (1999) 2435–2463.

[17] S. Staniford, J. Hoagland, J. McAlerney, Practical automated detection of stealthy portscans, Journal of Computer Security 10 (1) (2002) 105–136.

[18] S. Staniford-Chen et al., GrIDS – a graph-based intrusion detection system for large networks, in: Proceedings of the 19th National Information Systems Security Conference, vol. 1, October 1996, pp. 361–370.

[19] W. Fang, L. Peterson, Inter-AS traffic patterns and their implications, in: Proceedings of the IEEE GLOBECOM, December 1999.

[20] A. Broder, M. Mitzenmacher, Network applications of bloom filters: a survey, Internet Mathematics 1 (4) (2003) 485–509.

[21] L. Fan, P. Cao, J. Almeida, A.Z. Broder, Summary cache: a scalable wide-area web cache sharing protocol, Technical Report 1361, Computer Sciences Department, Univ. of Wisconsin-Madison, February 1998.

[22] C. Fraleigh et al., Packet-level traffic measurements from the Sprint IP backbone, IEEE Network 17 (13) (2003) 6–16.

[23] NLANR, National laboratory for applied network research, 2003, <http://pma.nlanr.net/Traces/>.

[24] E. Daskalova, M. Ilvesmaki, R. Kantola, Analysis of flow inter-arrival time distributions, in: Proceedings of the

IASTED Conference on Internet and Multimedia Systems and Applications, February 2005.
[25] S.M. Ross, Probability Models for Computer Science, Harcourt/Academic Press, 2002.
[26] A. Papoulis, Probability, Random Variables, and Stochastic Processes, third ed., McGraw-Hill, New York, 1991.
[27] R.W. Wolff, Stochastic Modeling and the Theory of Queues, Prentice Hall, 1989.
[28] E.R. Dougherty, Probability and Statistics for the Engineering, Computing and Physical Sciences, Prentice-Hall, 1990.

Seung Yeob Nam received the B.S., M.S., and Ph.D., degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea, in 1997, 1999, and 2004, respectively. From 2004 to July 2006, he was a Postdoctoral research fellow at CyLab in Carnegie Mellon University, supported by both CyLab and the Postdoctoral Fellowship Program of the Korea Science & Engineering Foundation (KOSEF). Between August 2006 and February 2007, he was a Postdoctoral researcher in the Dept. of EECS, KAIST. In March 2007, he joined the Department of Information & Communication Engineering, Yeungnam University, Gyeongsan, Korea, as a faculty member. His research interests include network monitoring, traffic engineering, network security, high-speed switching systems, wireless networks, etc. He received the Best Paper Award from the APCC 2000 Conference and Bronze prize from 2004 Samsung Humantech paper contest.

Hyu-Dae Kim received his B.S., MS., and Ph.D., degrees from the school of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), in 1999, 2001, and 2008, respectively. From 2001 to 2004, he was with IPOne Inc. as a research engineer. In 2008, he joined the KAIST Institute for Information Technology Convergence (KIITC), as a senior research engineer. His research interests include energy efficient protocols and cross layer design in 4G wireless communication systems and sensor networks.

Hyong S. Kim received the B.Eng. (Hons) degree in electrical engineering from McGill University, Montreal, Canada, in 1984, and the M.A.Sc. and Ph.D., degrees in electrical engineering from the University of Toronto, Toronto, Canada, in 1987 and 1990, respectively. Since 1990, he has been with Carnegie Mellon University, Pittsburgh, PA, where he is currently the Drew D. Perkins Chaired Professor of Electrical and Computer Engineering. His primary research areas are advanced switching architectures, fault-tolerant, reliable, and secure network architectures, and optical networks. His pioneering work on switch architectures has influenced many switching system designs in telecom industry. His Tera ATM switch architecture developed at CMU has been licensed for commercialization. He worked in Northern Telecom in 1992 as a research consultant addressing issues in high-speed network architectures. In 1995, he founded Scalable Networks, a Gigabit-Ethernet switching startup. Scalable Networks was later acquired by FORE Systems in 1996. He was at FORE Systems working on the company's technology roadmap until 1998. In 2000, he founded AcceLight Networks, an optical switching startup, and was CEO of AcceLight Networks until 2002. He is an author of over 80 published papers and holds over 80 patents in networking technologies. He was an editor for IEEE/ACM TRANSACTIONS ON NETWORKING from 1995 to 2000.