A Scalable Set-Union Counting Approach to Pushing Back DDoS Attacks

Yu-Kwong Kwok, Min Cai, and Kai Hwang
University of Southern California, Los Angeles, CA 90089, USA *

November 5, 2004

Abstract

It is a pressing task to fight off DDoS (Distributed Denial-of-Service) attacks effectively on the Internet. However, despite that there has been a plethora of efforts suggested in the literature to combat DDoS attacks, we are yet to see a really practicable efficient solution. First of all, such a solution has to be scalable because we need to inspect and process a gigantic amount of traffic flows in order to accurately detect an attack in its incipient stage. Thus, maintaining per flow state or simple per-packet based book-keeping schemes will not work in practice. Secondly, such a solution has to be simple to implement such that the subsequent action—pushing the attack back—can be launched in a swift manner.

In this paper, we propose a novel technique for detecting incipient DDoS attacks and pushing the attack flows back to the perimeter ingress routers of an autonomous system. Our technique is based on probabilistic counting of the cardinality of the union of two crucial packet sets: ingress packet set and the destination packet set. In a high speed core network, these packet sets are huge, even if we just keep track of them over a short period of time. Our technique is of very low time- and space-complexity. Indeed, the amount of space needed to keep track of accurate traffic matrix entries is $O(\log \log n)$ (where n is the number of packets to be counted) so that the pushback mechanism is highly efficient because the amount of information to be communicated among routers is very small.

Our NS-2 simulation results indicate that our proposed new technique is highly effective in combating DDoS attacks with an accuracy rate at around 90%. This can be achieved with a small amount of counting storage, say 2 KBytes, at each router. Furthermore, the amount of legitimate packets dropped is less than 10% when the pushback mechanism is in action. In summary, the new technique is both time- and space-efficient, demand low-cost to implement and low-overhead to operate, results in high accuracy and low error rate, and is highly scalable in defending against DDoS attacks over an increasing number of routers.

Keywords: network security, distributed denial-of-service (DDoS) attacks, Internet, pushback, probabilistic counting, rate limiting algorithms, scalable algorithms.

1 Introduction

Distributed denial-of-service (DDoS) attacks [26] on the Internet are problems that urgently demand effective solutions because such attacks' adverse effects are detrimental while the costs of launching such attacks

*USC GridSec Technical Report TR-2004-21. This research was supported by an NSF ITR Research Grant under contract number ACI-0325409. Corresponding author: Kai Hwang, Email: kaihwang@usc.edu, Tel: 213-740-4470, Fax: 213-740-4418.

are minimal. Indeed, by its nature, a DDoS attack is difficult to deal with in an effective manner as indicated by the fact that in most cases people could only tackle the aftermath of the attack, instead of *detecting* the attack before it becomes a full-blown one and *terminating* it decisively during the incipient stage [6, 14, 16].

We strongly believe that the only effective solution to combat DDoS attacks is to be able to *identify* an incipient attack and then swiftly *pushback* to the boundary of the administrative domain [18]. The critical question in such an approach is how we can accurately and timely estimate what fraction of the attack traffic comes from each ingress router. Obviously, the estimation of the distribution of attack traffic can be done by employing the network traffic measurement. Once we have the traffic matrix among ingress routers or ingress links, we can easily figure out which ingress routers contribute most attack traffic. However, existing traffic measurement approaches are typically based on online flow measurement combined with offline traffic analysis [15, 19]. The major drawback with the flow measurement approach is its space-complexity when the scale of concurrent flows increases [19]. Although the scalability can be improved by only tracking those large flows as proposed by C. Estan et al. [12], it still needs large amount of processing time to compute traffic matrices from flow measurements. Thus, it is difficult to simply apply existing traffic measurement approaches to do online estimation of the attack traffic distribution among ingress routers or links. In summary, as detailed in Section 2, the design of an effective integrated DDoS detection and pushback mechanism remains a baffling and challenging problem.

In this paper, we meet this grand research challenge by proposing a new approach in quickly detecting an incipient DDoS attack and pushing it back to the ingress routers of the administrative domain involved. To estimate an accurate real-time distribution of DDoS attack traffic among routers, we present a scalable set-union counting approach to compute traffic matrices with very low space-complexity. We integrate our online DDoS traffic estimation with an efficient and practical pushback scheme to limit the rate of attack flows on the ingress routers that contribute the most in the attack traffic flows. Consequently, the false positives of the pushback scheme can be reduced while the false negatives are kept at a low level.

The organization of this paper is as follows. In the next section, we present our proposed scalable traffic tracking technique that is readily amenable to practical implementation due to its very low time- and space-complexity. Section 4 describes the online detection and pushback algorithm which is based on the traffic tracking technique. Section 5 contains the experimental results using NS-2 [23] and our interpretations. We provide some concluding remarks in the last section.

2 Related Work

The DDoS problem has attracted intense attention from the research community. Various kinds of measures have been proposed to tackle the problem including sophisticated router packet filtering schemes [9, 24, 22], flow tracking mechanisms [10, 33], traceback techniques [1, 2, 5, 7, 25, 27, 28, 29], pushback algorithms [18, 34], etc. In particular, a myriad of algorithms have been proposed for the traceback problem. However, in our study (as detailed in Section 3), we argue that tracing back the detailed path information is unnecessary for effective detection and pushback of DDoS attack flows. Specifically, we believe that for pushing back DDoS attacks away from an AS, we need to identify the edge routers only but not also the core routers. In the following, we summarize some previous research efforts that are more closely related to our approach. For an excellent survey of DDoS defense solutions, the reader is referred to [17, 21, 30].

Yau et al. [34] suggested a scheme based on max-min fairness control at an upstream router connected to a server that is potentially under attack. The scheme works by having the server notifies the upstream router about the desired rate control levels for certain flows destined at the server. Control-theoretic analysis shows that the proposed scheme is effective. However, the major deficiency is that the scheme lacks the mechanism for identifying the "major" routers that should participate in the rate control process. Ioannidis and Bellovin [18] are also pioneers in proposing the concept of pushback—to rate limit the potentially problematic flows in some upstream routers. They described in detail an implementation of the pushback idea. However, the congestion signature used in their scheme is costly to verify in that it requires high time-complexity actions

(with respect to the processing budget in a router) such as sorting.

Sung and Xu [31] recently proposed an integrated scheme for responding to DDoS attacks. Their scheme is built on top of a traceback scheme which can provide information about which links the potential attack flows have traversed. Such link-traversal information is to be carried by preferential markings on the packets. With such information, the downstream router then tries to drop most of the packets that come from the paths reported by the underlying traceback scheme. Obviously, a major drawback of this approach is that the line of defense could be too close to the victim such that the attack flows may have already depleted the network resources to a large extent. Secondly, using a path-based differentiation method could possibly generate a large number of false positives. Wang and Shin [32] also recently suggested a scheme that is called "transport-aware" router-based protection, which works by leveraging on the differentiated services provisioning mechanisms in routers to segregate legitimate flows from potentially malicious flows. Different weights are then assigned to the flows so as to rate limit the malicious flows. Again, the major deficiency is that the overheads required in each router could be very high. Indeed, a large amount of processing is required to segregate the flows and compute their weights.

3 The Proposed Scalable Traffic Tracking Technique

In this section, we describe our proposed traffic tracking technique, which is based on a new approach to scalable set-union counting built on top of algorithms recently introduced by Durand and Flajolet [11].

In our formulation, we model the core network (i.e., the network of routers) as a directed graph $G = (V, E)$, where V is the set of routers R_i ($i = 1, \ldots, |V|$) and E is the set of directional links l_{ij} ($i, j = 1, \ldots, |V|$). Furthermore, we define the following key notation:

- $\mathcal{S}_i$: The set of packets that originate from a source router R_i (i.e., the packets are injected into the network via this router);
- $\mathcal{D}_i$: The set of packets that terminate at a destination router R_i (i.e., the packets leave the core network via this router);
- $\mathcal{L}_{ij}$: The set of packets that traverse a link l_{ij}.

With the above notation, to compute the traffic matrix $A = \{a_{ij}\}$, we need to determine:

$$a_{ij} = |\mathcal{S}_i \cap \mathcal{D}_j| \tag{1}$$

The other notation used throughout the paper is listed in Table 1.

A straightforward method to compute the intersection is to keep the two sets of packets and then perform a comparison. Indeed, in many previous research efforts, efficient set representation schemes, such as the Bloom Filter [3, 4], are used [10, 28]. However, in a core network with tens of Gbps link speeds, the number of packets in such sets would be astronomical, even over a very short period of time. Thus, it is simply impractical to maintain such sets in order to perform the computation of set intersections necessary for determining the traffic matrix.

Observe that the intersection computation can be transformed into a union operation [20]:

$$a_{ij} = |\mathcal{S}_i \cap \mathcal{D}_j| = |\mathcal{S}_i| + |\mathcal{D}_j| - |\mathcal{S}_i \cup \mathcal{D}_j| \tag{2}$$

Thus, we need to keep counters at each router R_i in order to determine $\mathcal{S}_i$ and $\mathcal{D}_i$. However, we also need to compute $|\mathcal{S}_i \cup \mathcal{D}_j|$ efficiently. To do this, we can make use of the counting algorithms recently designed by Durand and Flajolet [11], which in turn is based on an early piece of work by Flajolet and

Table 1: Definitions of notation.

Symbol	Definition
$N = \|V\|$	The number of routers in the AS
R_i	Router i
l_{ij}	Link connecting routers R_i and R_j
$\mathcal{S}_i$	Set of packets originated from R_i
$\mathcal{D}_i$	Set of packets terminated at R_i
$\mathcal{L}_{ij}$	Set of packets traverse the link l_{ij}
$A = \{a_{ij}\}$	Traffic matrix of the whole AS
$H()$	Hash function (we use SHA-1 in our study)
P	A packet
ID(P)	The unique ID of a packet P (composed of the IP address of the ingress router, the destination IP address, and a time-stamp)
B	A binary bit string, the hashed output of ID(P)
m	Number of buckets maintained by each router
$\rho(B)$	Position of the first bit 1 of the binary string B, counting from left to right
$\hat{n}$	The estimate of a set's cardinality by the Stochastic Averaging Algorithm [11]
α_m	The correction factor [11] for the estimate $\hat{n}$
τ	Measurement period
T_1	Threshold for checking the volume of traffic arrived at a router
T_2	Threshold for checking whether an ingress router is an *Attack-Transit Router* (ATR)
Γ	The real set of ATRs
$\hat{\Gamma}$	The estimated set of ATRs
θ_{positive}	The false positive rate of identifying ATRs
θ_{negative}	The false negative rate of identifying ATRs
X_i	The attack traffic volume from ingress router R_i to the last hop router
Y_i	The legitimate traffic volume from ingress router R_i to the last hop router
ξ	The ratio of legitimate traffic volume to attack traffic volume
ε_l	The relative error of dropped legitimate packets
ε_a	The relative error of leaked attack packets
δ_i	The uniform packet dropping rate at ingress router R_i

Martin [13]. We formalize our statistic tracking based on their ideas in Algorithm 1 below. For the details about the analysis and the counting theory, the reader is referred to [11, 13].

To obtain a stochastic estimate of the cardinality of a set, we need to make use of the following value ρ defined over a group of binary string $x \in \{0,1\}^\infty$ [13]: $\rho(x)$ is defined as the position, counting from left to right, of its first bit 1. For example, $\rho(1\cdots) = 1$, $\rho(001\cdots) = 3$ [13]. It can be shown [13] that the largest value of ρ for a set of randomly distributed binary bit strings generated by the hash function from a set of items (e.g., packets with unique IDs) is a very accurate estimate of $\log N$, i.e., the logarithm of the total number of unique items in the set. To obtain a better estimate, it is useful to distribute the random hashed output strings to m *buckets*. The more buckets used, the higher the accuracy. However, there is a trade-off in that the amount of time spent in computing the hashings and the space required to stored the strings would both be higher.

Algorithm 1 Statistics Tracking at Each Router

1: INPUT: A new packet P that arrives at the router (may be originated from, terminated at, or just passed by, the router);
2: OUTPUT: An updated m buckets of statistics;
3:
4: $B \leftarrow H(\text{ID}(P))$; {hashing the unique packet ID into a randomly distributed binary bit string}
5: $j \leftarrow B \bmod m$; {determine the bucket index for the new string}
6: Save B to the j-th bucket;
7: Update the value of ρ_j of the j-th bucket;

With Algorithm 1, based on the Stochastic Averaging Algorithm [11], each router can already compute the following estimate of the cardinality of a packet set (e.g., the set of packets originating from it) that the router is keeping track of:

$$\hat{n} = \alpha_m m 2^{\frac{1}{m}\sum_j^m \rho_j} \tag{3}$$

Here, α_m is a *correction factor* and is shown [11] to be:

$$(\gamma(-\frac{1}{m})\frac{1-2^{1/m}}{\log 2})^{-m} \tag{4}$$

where $\gamma(x)$ is given by:

$$\gamma(x) = \frac{1}{x}\int_0^\infty e^{-t}t^x dt \tag{5}$$

Specifically, mathematical analysis shown in [11] indicates that the standard error is approximately given by: $\frac{1.3}{\sqrt{m}}$. For example, with $m = 1024$, the theoretical standard error is about 4%.

To compute the cardinality of the union of two sets (e.g., $|\mathcal{S}_i \cup \mathcal{D}_j|$), we just need the two routers involved (i.e., routers R_i and R_j) to perform the Distributed Max-Merge Algorithm [11]. Specifically, to compute the estimate $\hat{N}$ using Equation 3, for each bucket, we take the larger value of the two from the corresponding buckets of the two routers. The resulting estimate $\hat{N}$ then stochastically approximates the value of $|\mathcal{S}_i \cup \mathcal{D}_j|$. We can then compute the value of $|\mathcal{S}_i \cap \mathcal{D}_j|$. Furthermore, we can also perform *traceback* by computing $|\mathcal{S}_i \cap \mathcal{L}_{xy} \cap \mathcal{D}_j|$ for each link l_{xy} connecting R_i and R_j. Nevertheless, to achieve DDoS detection and pushback, traceback is unnecessary.

Our proposed traffic matrix tracking technique is highly efficient, both in terms of space and time. First of all, it is clear that the time complexity is $O(n)$, where n is the number of packets in a set, because every packet only needs a constant amount of time for processing (e.g., hashing, saving into the bucket, etc.). Secondly, the tracking scheme is space efficient in that all the buckets together consume only $\log\log n$ bits of storage as shown by the analysis in [11, 13]. Thus, the bucket statistics can be efficiently exchanged among routers by piggy-backing the ρ values with the RIP packets.

Thus, the central question that we need to investigate is the *accuracy* of the stochastic approximation of the set-union cardinality, which is crucial in implementing an accurate DDoS detection and pushback scheme, as detailed below.

4 The Proposed DDoS Detection and Pushback Scheme

Equipped with the highly scalable traffic matrix tracking scheme described in Section 3, we can formalize our proposed online real-time DDoS detection and pushback scheme. In our proposed scheme, we make the following assumptions:

- The scheme is to be executed within a single autonomous system (AS) such that all the routers are cooperative and can trust each other.
- Each ingress router inserts its IP address in each incoming IP packet forwarded to it from outside of the AS. Furthermore, the router also inserts a time-stamp (or some unique sequence number) into the packet header. The only requirement of this time-stamp is that it will not wrap around within a time period τ (i.e., the measurement period, elaborated below). Thus, the tuple (ingress router IP address, time-stamp, destination IP address), can be used as a unique ID for each packet in the AS. Such an address and time-stamp marking can be done in a digitally signed and encrypted format.
- Thus, the pushback is done all the way up to the "perimeter" of the AS (i.e., to the ingress routers). Indeed, we assume, without loss of generality, that all the DDoS attacks are launched from outside of the AS, and thus, the flooding packets must pass through some ingress routers.
- Finally, we assume that the number of ingress routers involved in a single DDoS attack instance follows the "power-law", i.e., a majority (e.g., 80%) of the traffic passes through only a minority (e.g., 20%) of the ingress routers. We believe that, as indicated in [18], this is a reasonable assumption because it is very difficult for an attacker to uniformly distribute the attack traffic flows across all ingress routers of an AS.

With the above assumptions, the ID(P) of each packet P can be uniquely defined using the inserted ingress IP address, the unique time-stamp (or sequence number), and the destination IP address. This is done in order not to rely on the source IP address which can be trivially spoofed. The cooperation among routers within the AS in processing the packets is depicted in Figure 1. It should be noted that every router performs the statistics tracking actions (for clarity, only the right-most ingress router and the last hop router are annotated with the statistics tracking mechanisms), and, in particular, each ingress router inserts its IP address and a unique time-stamp into each packet entered the AS. Consequently, every router can make use of the unique composite ID for the hashing and counting actions.

Figure 2 illustrates the components involved in each router for DDoS detection and pushback. The Traffic Statistics Tracking Module is responsible for accumulating statistics using Algorithm 1. In Algorithm 2, two thresholds are used. Specifically, threshold T_1 is used by the Traffic Volume Monitoring Module for detecting whether the volume of traffic arriving at the router R_j is abnormally high over a certain fixed time period τ (e.g., five seconds). If so, it is reasonable for the router to suspect that a DDoS attack is in the incipient stage. Thus, in the Traffic Statistics Integration Module, the router R_j first attempts to determine where all these traffic flows come from, by checking the intersection between the sets S_i and D_j for all ingress routers R_i (with $i \neq j$) . If the cardinality of the intersection with R_i is higher than the threshold T_2, then the router R_j assumes that the ingress router R_i is involved in the forwarding of the attack flow. Such a router is called an Attack-Transit Router (ATR). Here, it should be noted that the flow volume going through each ingress router involved in the attack may not be too high. As such, the threshold T_2 should not be set to a very large number. Obviously, the exact values of these thresholds depend on the normal load levels of various legitimate traffic paths within the AS. The Rate-Limiting Pushback Module then notifies those suspected ATRs about the desired rate-control parameters for preferential packet dropping mechanisms in the ATRs (detailed in Section 5.4).

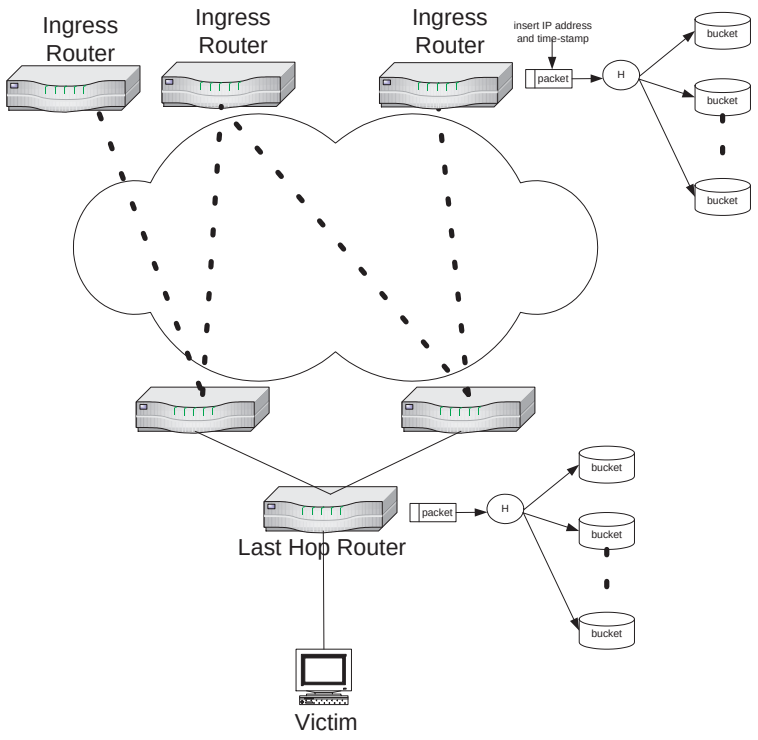


Figure 1: Control flow among the DDoS detection and pushback modules.

Algorithm 2 DDoS Detection and Pushback

for each router R_j in the AS **do**
  Execute Algorithm 1 forever;
  Monitor the value of $|D_j|$ over a fixed time period τ;
  if $|D_j| > T_1$ **then**
    for each ingress router R_i ($i \neq j$) in the AS **do**
      Get the m ρ values from router R_i;
      if $|S_i \cap D_j| > T_2$ **then**
        Perform pushback to router R_i: Notify R_i that it has to rate-limit (by preferential dropping) the incoming packets destined at R_j;
      end if
    end for
  end if
end for

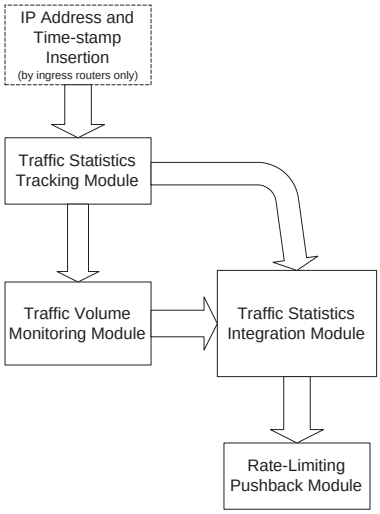


Figure 2: Structure of the DDoS detection and pushback mechanisms.

5 Performance Evaluation

In this section, we present the simulation results for various aspects of our set-union counting algorithm, including the relative error of traffic matrix estimates, the false negatives and false positives of identifying the ATRs, as well as the time and space costs of traffic matrix computations.

5.1 Simulator Implementation

We implemented our set-union counting algorithm in the NS-2 simulator [23], which is a widely recognized a packet level discrete event simulator. It uses C++ language for fast simulation speed and TCL language for easy configuration. To implement the traffic statistics tracking mechanism, we added two C++ classes into the simulator to compute the traffic matrices, i.e., `LogLogCounter` and `TrafficMonitor` classes. The `LogLogCounter` class is used to compute the loglog summary (i.e., the ρ values of the m buckets) of distinct packets traversed through each router. It is a subclass of `Connector` and is added to the head of each `SimplexLink` when it is initialized as shown in Figure 3. Thus, when a packet is forwarded to a particular link by the routing module on each node, the `recv` method of the `LogLogCouter` object associated with this link will be called. The `LogLogCounter` object first computes the uniformly distributed binary bit string B by applying the SHA-1 hash function on the unique ID of the packet. Then it updates the value of ρ in the corresponding bucket and passes the packet to the next `Connector` object associated with this link. The `TrafficMonitor` keeps track of all `LogLogCounter` objects and for each each time period τ, it will be triggered to compute the traffic matrix for this time period using the set-union counting algorithm.

5.2 Accuracy of Estimated Traffic Matrices

As mentioned before, the accuracy of the traffic statistics is a vital factor for the last hop router (in front of a victim) to correctly identify each and every ATR so as to push back the attack flow. To evaluate the accuracy of traffic matrix estimates, we first defined the *relative error* of a traffic matrix estimate as follows:

$$Er = \frac{\sum_{ij=1}^{K} (\hat{a}_{ij} - a_{i,j}) \Big/ a_{ij}}{K}, \quad \text{when} \quad a_{ij} \neq 0 \tag{6}$$

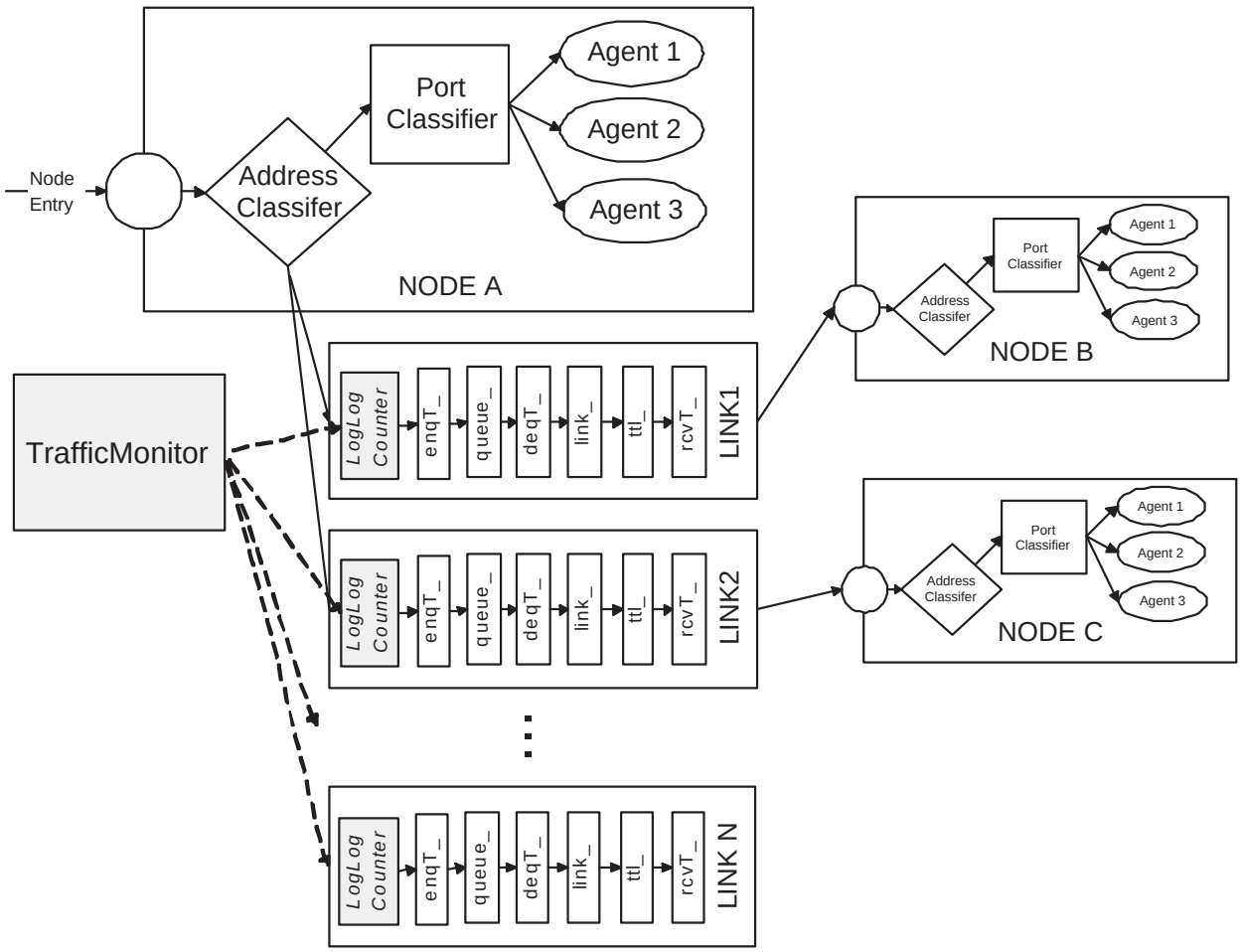


Figure 3: Our NS-2 implementation of the set-union counting algorithm.

where $\hat{a}_{ij}$ and a_{ij} are the estimated and real number of packets from router i to router j respectively, K is the number of non-zero elements in the matrix and N is the number of routers. Obviously, when Er decreases, the traffic matrix estimate becomes more accurate and close to the real traffic matrix.

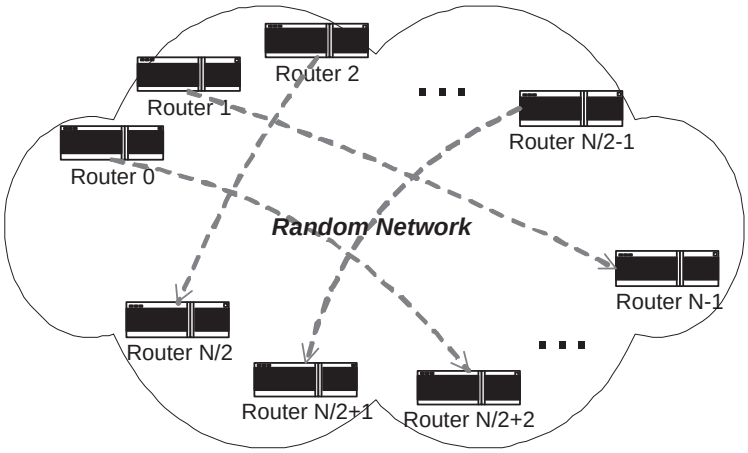


Figure 4: The simulation setup for measuring the accuracy of estimated traffic matrices.

Our first simulation configuration consists of different network sizes up to 200 routers. In this set of simulations, we generated a random topology for each run using the GT_ITM [35] topology generator. We also randomly partitioned the N routers into two equal-sized group. Routers 1 to $N/2$ are ingress routers and Router $N/2$ to N are egress routers. Each ingress router i generates traffic volume a_{ij} to a random egress router j as shown in Figure 4. The traffic volumes a_{ij} follow a normal distribution with mean μ and standard deviation σ.

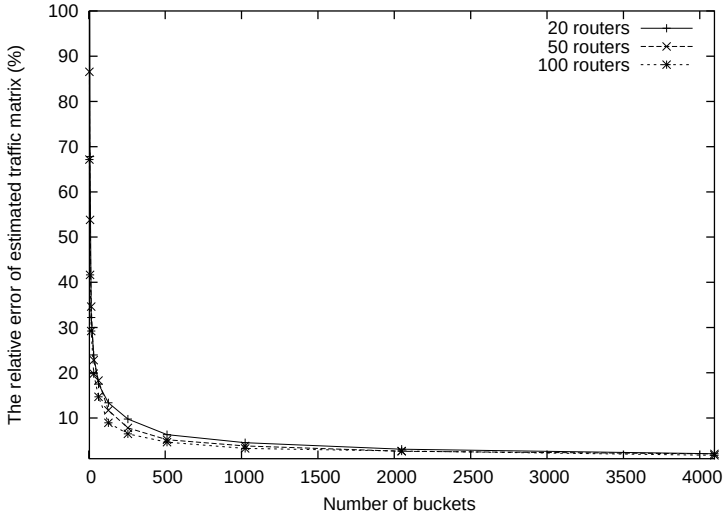


Figure 5: The relative error of the estimated traffic matrix decreases significantly when the number of buckets increases for a network with 50 routers.

As Figure 5 shows, the relative error of the estimated traffic matrix decreases significantly when the number of buckets increases from 4 to 4096. This simulation is performed in a network of 50 routers, and the mean and standard deviation of traffic rates are 100K pps (packets per second) and 20K pps respectively. When there are only 4 buckets, the error ratio of the measured traffic matrix is 87%. It decreases rapidly to 7% and 2% when the number of buckets increases to 512 and 4096, respectively.

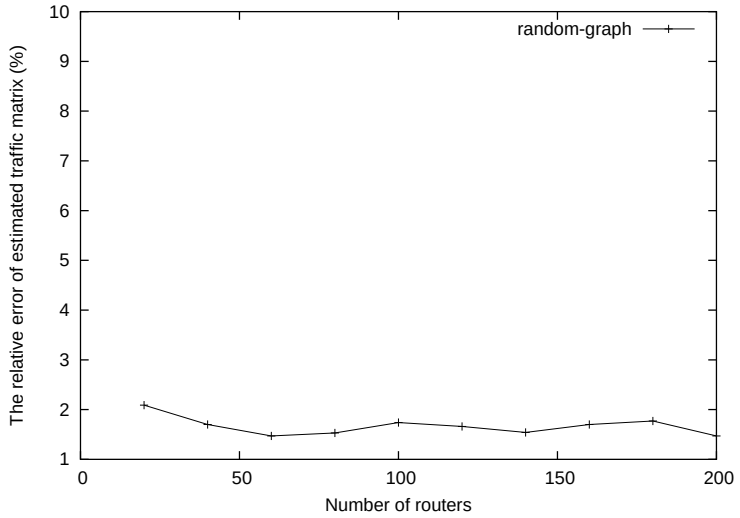


Figure 6: The relative error of the estimated traffic matrix almost the same when the network increases from 20 routers to 200 routers.

We scaled the network from 20 routers to 200 routers and measured the relative errors for different network sizes. Figure 6 shows that the relative errors are almost the same when the network scales up to 200 routers. Therefore, our traffic statistics tracking scheme based on set-union counting approach is quite robust for different sizes of AS networks.

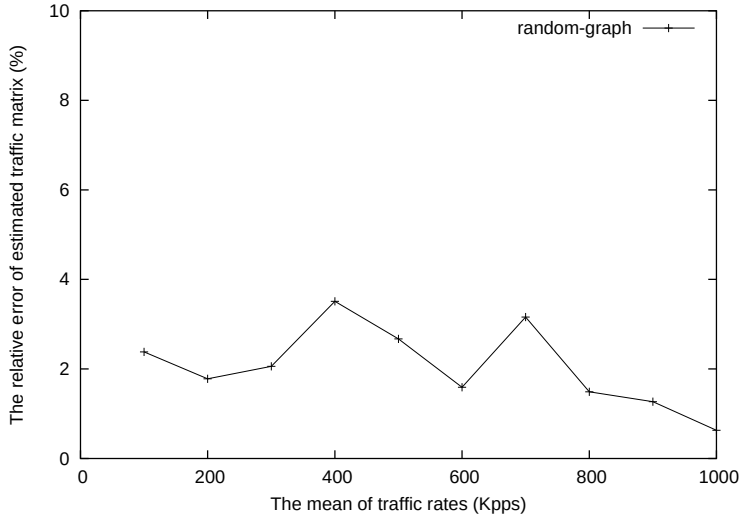


Figure 7: The relative error of the estimated traffic matrix decreases slightly when the average traffic rate increases.

We vary the traffic volume sent by each router in a given interval and measure the relative error for different sizes of traffic. We increased the average traffic rate on each router from 100 Kpps to 1000 Kpps and the relative error decreases slightly when the average traffic rate increases as shown in Figure 7. This is because we use 4096 buckets in this experiment and when traffic rate is only 100 Kpps, each bucket will only be hashed with about 25 packets for 1 second interval. Therefore the relative error is slight high in this case since there are too few samples for each bucket. However, when the traffic rate increases up to 1000 Kpps, the samples per buckets are increased up to 250 and the relative error of the estimation is decreased

slight.

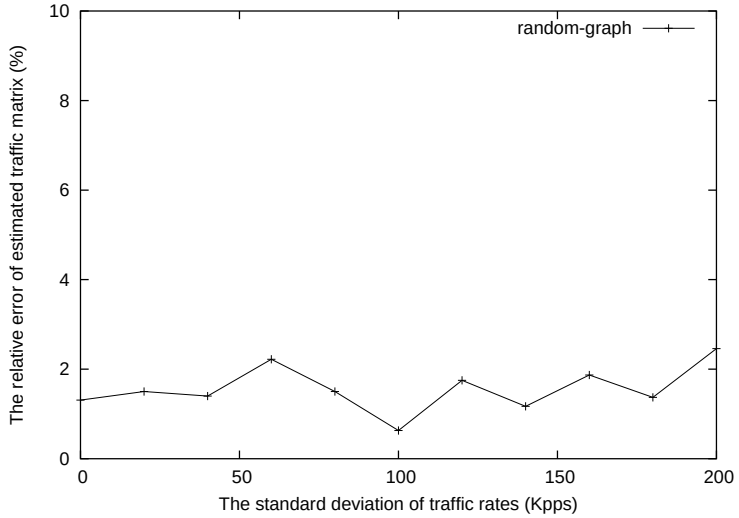


Figure 8: The relative error of the estimated traffic matrix almost the same when the standard deviation of traffic rates increases.

We also increased the differences of traffic rates among routers by increasing the standard deviation of traffic rates. In this simulation, we fixed the average traffic rate to be 1000 Kpps and varied the standard deviation from 0 to 200 Kpps. Figure 8 shows that the relative errors of estmated traffic matrices are almost the same even when the standard deviation of traffic rates increases from 0 to 200 Kpps.

5.3 Accuracy of Estimated Attack-Transit Routers

As we mentioned in Section 4, we need to identify the set of ingress routers that contribute most attack traffic. Such routers are called *Attack-Transit Routers (ATR)*. For a given victim v and the percentile p of the total attacking traffic volume, we identify the set of ATRs, denoted by $\Gamma(p)$, as the first k ingress routers in descending order of their traffic volumes to v and the sum of their traffic volumes to v is p percentage of the total traffic volume from all ingress routers in the AS. We define the *false positive* rate, θ_{positive}, of our estimated set of ATRs Γ as follows:

$$\theta_{\text{positive}} = \frac{|\Gamma - \hat{\Gamma}|}{|\hat{\Gamma}|} \tag{7}$$

and, similarly, the *false negative* rate, θ_{negative}, as following:

$$\theta_{\text{negative}} = \frac{|\hat{\Gamma} - \Gamma|}{|\hat{\Gamma}|} \tag{8}$$

To evaluate the accuracy of the identification of the set of ATRs, we set up a simulation scenario in which one victim is attached to Router R_0 and all other routers are sending attack traffic as well as legitimate traffic to the victim via Router R_0, as shown in Figure 9. The attack traffic volume from router R_i to the victim is X_i and the legitimate traffic volume is Y_i, where $0 < i < N$. In this simulation, we assume that X_i follows exponential distribution with mean μ_x and Y_i follows normal distribution with mean μ_y and standard deviation σ_y. We define the ratio of legitimate traffic to attack traffic ξ as follows:

$$\xi = \sigma_x / \sigma_y \tag{9}$$

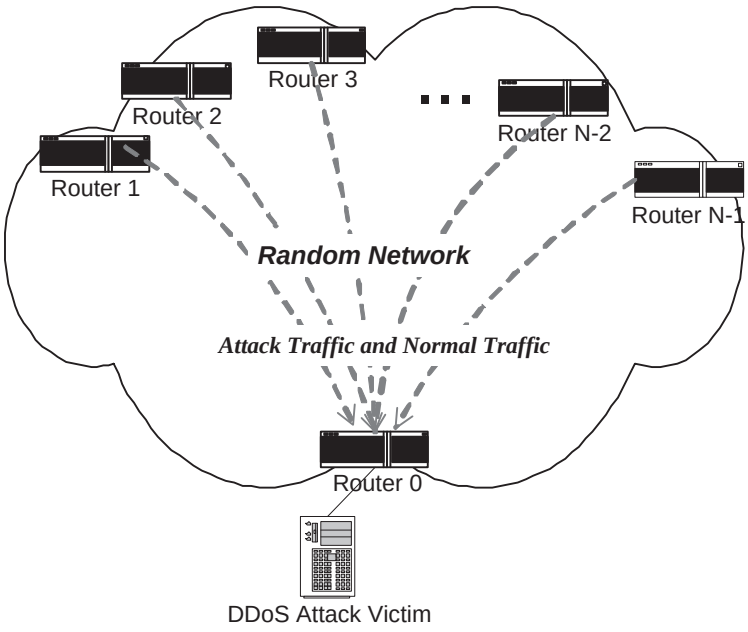


Figure 9: The simulation setup for measuring the accuracy of identifying ATRs.

When a DDoS is raging on, the total amount of attack traffic could be very high. Thus, we varied ξ from 0 to 1. Figure 10 shows the false positive rates of attacking-significant routers Γ for a network of 50 routers with percentile p equal to 70%, 80% and 90%. As can be seen, the false positive rate of identifying ATRs increases as ξ increases because as more and more legitimate traffic flows are in transit in the network, it is more likely to mis-interpret a legitimate flow as an attack flow. Thus, the probability of mis-interpreting a router as an ATR is also higher. However, this phenomenon should be considered very carefully in that mis-interpreting a router as an ATR does not mean that that router does *not* contribute any attack flow. Rather, such a router could just be a "minor" contributor and, as such, asking it to perform preferential packet dropping is also useful for eliminating some attack packets. This is evident in the results shown in Section 5.4 below.

5.4 Error of The Simple Rate-Limiting Scheme

After we identify the set of ATRs, we need to install packet filters on those routers to limit the rate of the attack traffic. Since the attackers can spoof their source IP addresses, it is still an open question to identify on routers whether an incoming packet is an attack packet or a legitimate packet. We believe that further research will work out some practical approaches for this problem and we can easily integrate them with our work. However, in this paper we assume a very simple rate-limiting scheme to evaluate the errors of the legitimate packets we dropped and the attack packets leaked through the rate-limiting flow control.

Suppose we drop packets uniformly on an ingress router R_i with dropping rate δ_i, where $i \in \Gamma$, we can compute the relative error of dropped legitimate packets ε_l as follows:

$$\varepsilon_l = \frac{\sum_{i \in \Gamma} Y_i \times \delta_i}{\sum_{i \in \Gamma} Y_i} \tag{10}$$

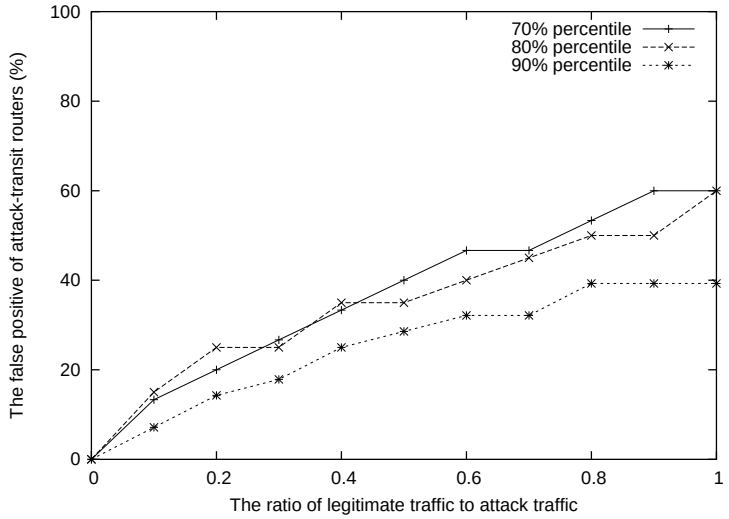


Figure 10: The false positive rate of identifying ATRs versus the ratio of legitimate traffic to attack traffic.

and the relative error of leaked attack packets ε_a is given by:

$$\varepsilon_a = \frac{\sum\limits_{i\in\Gamma} X_i \times (1-\delta_i) + \sum\limits_{i\notin\Gamma} X_i}{\sum\limits_{i} X_i} \tag{11}$$

Since the total dropped packets should be equal to the difference between all packets arriving at Router R_0 and its threshold T_1, the dropping rate δ_i for each attacking-significant router should satisfy the following to constraints:

$$\sum_{i\in\Gamma(p)} \delta_i \times (X_i + Y_i) = p \times (\sum_{i=1}^{N} (X_i + Y_i) - T_1) \tag{12}$$

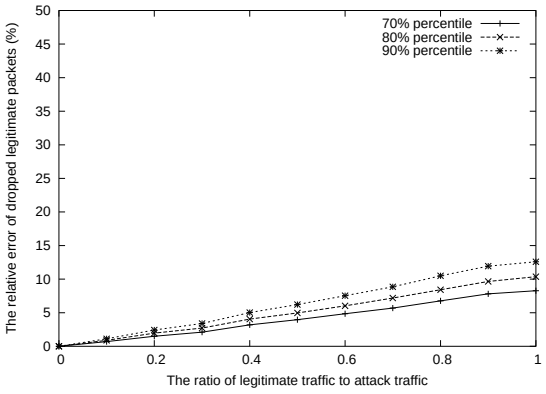
Furthermore, the dropping rate δ_i for router i should be proportional to the traffic volume it contributes to the victim, so we have:

$$\delta_i \propto (X_i + Y_i) \tag{13}$$

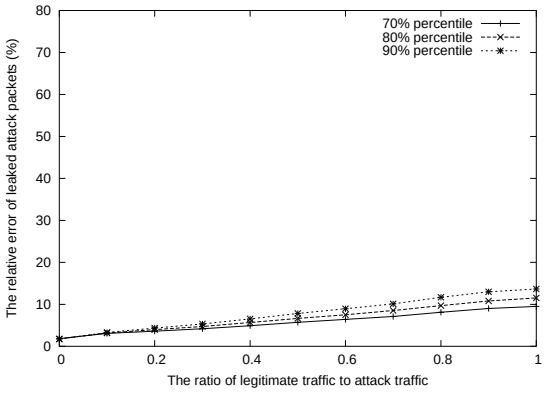
After solving the above Equations 12 and 13, we have:

$$\delta_i = \frac{p \times (\sum\limits_{i=1}^{N} (X_i + Y_i) - T_1) \times (X_i + Y_i)}{\sum\limits_{i\in\Gamma} (X_i + Y_i)^2} \tag{14}$$

Figure 11(a) shows that the relative error of dropped legitimate packets ε_l increases from around 2% to around 10% when the ratio ξ increases from 0.2 to 1.0 for a network of 50 routers with percentile p equal to 80%. This is a highly encouraging results because the pushback action only slightly affects the legitimate flows, even when the amount of legitimate traffic is high (i.e., at $\xi = 1$). When percentile p is equal to 70% and 90%, ε_l increases in a similar fashion when the ratio ξ increases. For the same ratio ξ, the relative error also increases when the percentile p increases. Figure 11(b) also illustrates similar results for the relative error of leaked attack packets ε_a. The relative error ε_a increases from around 4% to 10% when the ratio ξ increases from 0.2 to 1.0. When the ratio ξ is small, the difference between the relative error



(a) False negative rate of dropped legitimate packets



(b) False positive rate of leaked attack packets

Figure 11: Packet dropping accuracy versus the ratio of legitimate traffic to attack traffic.

ε_a for different percentile p is insignificant. Furthermore, when the ratio ξ becomes larger, the difference between the relative error ε_a for different percentile p stays more or less the same. Overall, we can see that given just a simple uniform packet filtering scheme, the false positive and and false negative rates are very low.

6 Conclusions

DDoS attacks are severe "diseases" on the Internet that should be "cured" urgently. However, due its low cost in launching and high stakes involved in the success of such an attack, it is a great challenge to devise effective solutions which must be highly scalable and readily practicable. In this paper, we propose a simple yet efficient traffic statistics tracking scheme based on the elegant probabilistic counting theory. Our proposed tracking scheme is efficient both in time and space, and can be easily implemented in a typical router. Based on traffic tracking algorithm, we also suggest a simple pushback scheme in which preferential packet dropping rates are set in the ingress routers according to the traffic volumes observed at the last hop router. Our NS-2 simulation results indicate that our proposed traffic scheme is highly accurate with only a tiny storage (e.g., 2 Kbytes) maintained at each router, and the pushback scheme is efficient in that the false positive and negative rates are reasonably low. We are currently implementing the proposed schemes in the DETER testbed [8] so that we can observe their practical behaviors and efficiency in a large-scale simulated Internet environment.

References

[1] H. Aljifri, "IP Traceback: A New Denial-of-Service Deterrent," *IEEE Security and Privacy Magazine*, May/June 2003, pp. 24–31.

[2] A. Belenky and N. Ansari, "On IP Traceback," *IEEE Communications Magazine*, July 2003, pp. 142–153.

[3] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Comm. ACM*, vol. 13, no. 7, July 1970, pp. 422–426.

[4] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Proc. 2002 Allerton Conference*.

[5] H. Burch and B. Cheswick, "Tracing Anonymous Packets to their Approximate Source," *Proc. 2000 USENIX LISA Conf.*

[6] R. K. C. Chang, "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Communications Magazine*, Oct. 2002, pp. 42–51.

[7] D. Dean, M. Franklin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *Proc. 2001 Network and Distributed System Security Symp.*

[8] DETER and EMIST Team Members, "Cyber Defence Technology Networking and Evaluation," *Comm. ACM*, vol. 47, no. 3, Mar. 2004, pp. 58–61.

[9] T. Doeppner, P. Klein, and A. Koyfman, "Using Router Stamping to Identify the Source of IP Packets," *Proc. 2000 ACM Conf. Computer and Comm. Security*.

[10] N. G. Duffield and M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, June 2001, pp. 280–292.

[11] M. Durand and P. Flajolet, "LogLog Counting of Large Cardinalities," *Proc. European Symposium on Algorithms 2003*.

[12] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," *Proc. ACM SIGCOMM 2004*.

[13] P. Flajolet and G. N. Martin, "Probabilistic Counting Algorithm for Data Base Applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, Oct. 1985, pp. 182–209.

[14] L. Garber, "Denial-of-Service Attacks Rip the Internet," *Computer*, vol. 33, no. 4, Apr. 2000, pp. 12–17.

[15] F. Hao, M. Kodialam, and T. V. Lakshman, "ACCEL-RATE: A Faster Mechanism for Memory Efficient Per-Flow Traffic Estimation," *Proc. ACM SIGMETRICS 2004*.

[16] K. Hwang, H. Liu, and Y. Chen, "Protecting Network-Centric Systems with Joint Anomaly/Intrusion Detection over Internet Episode," submitted for publication.

[17] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," *Proc. ACM SIGCOMM 2003*.

[18] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defence Against DDoS Attacks," *Proc. 2002 Network and Distributed System Security Symp.*

[19] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: Statistics-Based Overload Control Against Distributed Denial-of-Service Attacks," *Proc. INFOCOM 2004*.

[20] M. Kodialam, T. V. Lakshman, and W. C. Lau, "High-speed Traffic Measurement and Analysis Methodologies and Protocols," pending U.S. patent, filed on Aug. 2, 2004.

[21] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defence Mechanisms," *ACM Computer Communications Review*, vol. 34, no. 2, Apr. 2004, pp. 39–54.

[22] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proc. USENIX Security 2001*.

[23] NS-2, http://www.isi.edu/nsnam/ns/, 2004.

[24] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proc. ACM SIGCOMM 2001*.

[25] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM 2000.*

[26] C. Schuba et al., "Analysis of a Denial of Service Attack on TCP," *Proc. 1997 IEEE Symp. Security and Privacy*.

[27] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," *Proc. ACM SIGCOMM 2001*.

[28] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-Packet IP Traceback," *IEEE/ACM Trans. Networking*, vol. 10, no. 6, Dec. 2002, pp. 721–734.

[29] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. INFOCOM 2001*.

[30] S. M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures," *Proc. PDCS 2004*.

[31] M. Sung and J. Xu, "IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 9, Sept. 2003, pp. 861–872.

[32] H. Wang and K. G. Shin, "Transport-Aware IP Routers: A Built-In Protection Mechanism to Counter DDoS Attacks," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 9, Sept. 2003, pp. 873–884.

[33] A. Yaar, A. Perrig, and D. Song, "PI: A Path Identification Mechanism to Defend Against DDoS Attacks," *Proc. 2003 IEEE Symp. Security and Privacy*.

[34] D. K. Y. Yau, J. C. S. Lui, and F. Liang, "Defending Against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles," *Proc. 2002 IEEE Workshop on Quality of Service*.

[35] E. W. Zegura, K. Calvert, and M. J. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, Dec. 1997, pp. 770–783.

Authors' Biographical Sketches

Min Cai received his B.S. and M.S. degrees in Computer Science from Southeast University, China, in 1998 and 2001. He joined IBM Research, Beijing, in 2001, where he worked on multimedia networking. He is currently a Ph.D. student in the Computer Science Department at the University of Southern California and a Graduate Research Assistant in the Distributed Scalable Systems Division at Information Sciences Institute (ISI). His research interests include distributed resource allocation, peer-to-peer systems, and grid computing. His email address is mcai@isi.edu.

Kai Hwang is a Professor and Director of Internet and Grid Computing Laboratory at the University of Southern California. He received the Ph.D. from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, and distributed computing systems. He has authored or coauthored 6 scientific books and 170 journal/conference papers in the above areas. Professor Hwang is the founding Editor-in-Chief of the *Journal of Parallel and Distributed Computing*. He has performed advisory and consulting work for IBM Fishkill, Intel SSD, MIT Lincoln Lab., ETL in Japan, and GMD in Germany. Presently, he leads the NSF-supported ITR GridSec project at USC. The GridSec group develops security-binding techniques for trusted outsourcing in Grid computing. They also build self-defense software systems for protecting Grid and distributed computing resources. Professor Hwang can be reached at kaihwang@usc.edu or through the URL: http://GridSec.usc.edu/Hwang.html.

Yu-Kwong Kwok received the B.S. degree in Computer Engineering from the University of Hong Kong, Hong Kong, in 1991, and the M.Phil. and Ph.D. degrees in Computer Science from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 1994 and 1997, respectively. He is an Associate Professor in the Department of Electrical and Electronic Engineering, University of Hong Kong. Dr. Kwok is currently on leave from the University of Hong Kong and is a Visiting Associate Professor at the University of Southern California. His research interests include Grid computing, mobile computing, wireless communications, network protocols, and distributed computing algorithms. Dr. Kwok is a member of the Association for Computing Machinery (ACM), the IEEE Computer Society, and the IEEE Communications Society. He is a Senior Member of the IEEE. He can be reached at ykwok@hku.hk.