IP Traceback-based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks*

Minho Sung and Jun Xu
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

August 17, 2002

Abstract– Distributed Denial of service (DDoS) is one of the most difficult security problems to address. While many existing techniques (e.g., IP traceback) focus on tracking the location of the attackers after-the-fact, little is done to mitigate the effect of an attack while it is raging on. In this paper, we present a novel technique that can effectively filter out the majority of DDoS traffic, thus improving the overall throughput of the legitimate traffic. The proposed scheme leverages on and generalizes the IP traceback schemes to obtain the information concerning whether a network edge is on the attacking path of an attacker ("infected") or not ("clean"). We observe that while an attacker will have all the edges on its path marked as "infected", edges on the path of a legitimate client will mostly be "clean". By preferentially filtering out packets that are inscribed with the marks of "infected" edges, the proposed scheme removes most of the DDoS traffic while affecting legitimate traffic only slightly. Simulation results based on real-world network topologies all demonstrate that the proposed technique can improve the throughput of legitimate traffic by 3 to 7 times during DDoS attacks.

Keywords– Distributed Denial of Service (DDoS), IP traceback, Performance modeling and Simulation.

1 Introduction

Distributed Denial of Service (DDoS) attacks against high-profile web sites such as Yahoo, CNN, Amazon and E*Trade in early 2000 [1] demonstrate how damaging DDoS attacks are, and how defenseless the Internet is under such attacks. The services of these web sites were

*A preliminary version of the paper will be presented at the 10th IEEE International Conference on Network Protocols, to be held in Paris, France, from November 12th to 15th, 2002.

unavailable for hours or even days as a result of the attacks. New instances of DDoS attacks continue to be reported.

In a DDoS attack, a human *adversary* first compromises a large number of Internet-connected hosts by exploiting network software vulnerabilities such as *buffer overflows*. DDoS software such as TFN (Tribe Flood Network) is then installed on them. These hosts will later be commanded by the adversary to simultaneously send a large volume of traffic to a victim host or network. The victim is overwhelmed by so much traffic that it can provide little or no service to its legitimate clients. We refer to such compromised hosts as *attackers* in the rest of paper, although they may be "victims" themselves.

Unfortunately, today's Internet is not equipped with proper defense mechanisms against DDoS attacks[1]. Even tracing the perpetrators of an attack proves a daunting task, since the source IP address of a packet can easily be spoofed. To allow for effective attack origin tracing in the future Internet, a large number of IP traceback schemes [2, 3, 4, 5, 6, 7] have been proposed. These schemes (except [2, 7]) all require intermediate routers to inscribe a mark into the packet header that encodes the identity of an intermediate router or edge. After collecting a sufficient number of such marks, the victim will be able to reconstruct the network paths leading to the attackers. While IP traceback techniques allow the victim to infer the origins of the attack after the fact, they are in general not able to mitigate the effect of a DDoS attack while it is raging on.

In this paper, we propose a *protocol-independent* DDoS defense scheme that is able to dramatically improve the throughput of legitimate traffic during a DDoS attack. It works by performing "smart filtering": dropping DDoS traffic with high probability while allowing most of the legitimate traffic to go through. This clearly requires the victim to be able to

[1]Although many routers have *ingress filtering* capabilities, they are usually not turned on due to their nonnegligible overhead on router performance.

statistically distinguish legitimate traffic from DDoS traffic. The proposed scheme leverages on and extends IP traceback techniques to gather "intelligence": information such as whether or not a network edge is on the path from an attacker ("infected"). By preferentially filtering out packets that are inscribed with the mark (identity) of an "infected" edge, the proposed scheme filters out most of the traffic from attackers, since each and every edge on an attacker's path to the victim is infected. Packets from a legitimate client, on the other hand, with high probability will not be filtered out, since typically most of the edges on the client's path to the victim are not infected. To evaluate its effectiveness in defending against DDoS attacks, the proposed scheme is simulated on three sets of real-world Internet topologies with varying operating parameters. Simulation results demonstrate that the throughput of the legitimate traffic can be increased by 3 to 7 times. To the best of our knowledge, this is the first research that leverages on IP traceback for *automatic* DDoS response. What makes this approach more appealing is that the operations required of routers (probabilistic marking) are fully in line with the operations of IP traceback.

The rest of the paper is organized as follows. In the next section, we present the system model and the overview of the proposed scheme. Sections 3 and 4 present the algorithmic components of the scheme in detail. Its performance is evaluated in Section 5 through simulation studies. Section 6 discusses the potential applications of the proposed scheme. Section 7 surveys the related work and Section 8 concludes the paper.

2 System Model and Overview

The proposed scheme extends and leverages on existing IP traceback schemes [3, 4, 5, 6, 8]. IP traceback allows the victim to infer the paths that packets from the attackers have traversed. The proposed scheme uses this information to preferentially filter out packets that are more

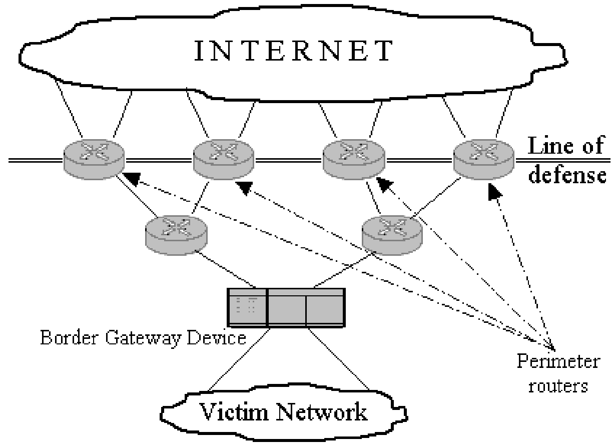

Figure 1: The Perimeter router model

likely to come from the attackers. Here we first discuss where in the network the filtering will actually take place.

2.1 System Model

It was first observed in [9] that, when a DDoS attack occurs, most of the traffic is dropped by the upstream routers even before it reaches the victim. In this case, nothing can be done by the victim to improve the throughput of the legitimate traffic. To mitigate the attack, proper action needs to be taken at upstream routers. Therefore, we adopt a similar system model to [9], shown in Fig. 1. The protected network is connected to the wide area network (WAN) through a gateway access control device (i.e., a firewall). A set of upstream routers will form a "line of defense", referred to as *a perimeter* in the sequel. The routers on the perimeter, referred to as *perimeter routers*, will collaboratively inspect packets going through them. For simplicity of discussion, we assume that all perimeter routers are of the same distance (referred to as *perimeter radius*) away from the victim[2].

[2] In situations where the upstream routers are not bottlenecked, this filtering process may happen at the gateway access control device (firewall). This degenerative case can be referred to as perimeter of radius 0.

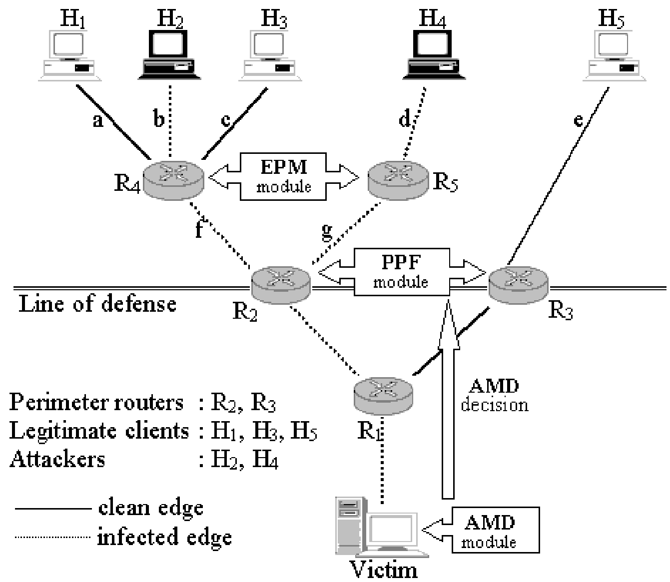

Figure 2: Example network as seen from the victim

2.2 Overview of the Proposed Scheme

The proposed scheme improves the throughput of legitimate traffic during a DDoS attack, by preferentially filtering out traffic that is more likely to come from an attacker than a legitimate host. To make this distinction, it leverages on and extends IP traceback techniques to infer whether or not a network edge is on the path from an attacker. The scheme can be viewed as a distributed algorithm that consists of the following three modules. The coupling of these modules with the actual physical devices and the interactions between these modules are shown in Fig. 2.

- Enhanced Probabilistic Marking (EPM) module. This module is an extension of the probabilistic marking module in IP traceback schemes. This *light-weight* module is running on each participating Internet router, whether or not there are DDoS attacks. Recall that in IP traceback, each Internet router needs to inscribe, with a certain probability, a mark into infrequently-used IP header fields (e.g., fields used for IP fragmentation) [4]. Our scheme further splits such marks into two types, described later in Section 3. Packets of the first type are used by the AMD module below for IP traceback and packets of the second type are used by the PPF module below for intelligent filtering.

- Attack Mitigation Decision-making (AMD) module. Running on the victim or the border gateway device (e.g., firewall) of the victim site, this module implements two functions: (a) reconstructing attack paths using existing IP traceback algorithms based on information contained in the aforementioned marks of the first type once an attack is detected, and (b) making algorithmic decisions as to packets inscribed with what type of marks should be passed with what probability, based on the results from (a). The decisions from (b) will be conveyed to the perimeter routers to be carried out. We will show that there is little communication overhead in transporting such information.

- Preferential Packet Filtering (PPF) module. This module is running on every perimeter router. These modules will differentially filter packets (destined for the victim) that contains the aforementioned marks of the first type, based on the instructions issued to them from the AMD module, once an attack is detected. We will show that little processing overhead is incurred at the perimeter routers: each filter/pass decision requires only the computation of a hash value and a table lookup.

In the following two sections, we describe the design of the three aforementioned modules in detail. Before we proceed, we introduce the concepts to be used in later discussions. Figure 2 shows an (unrealistically small) example network. It can be viewed as a DAG (Directed Acyclic Graph) rooted at the victim. In a real-world attack scenario, the victim could be either a single host or a border gateway device (e.g., firewall) of a network under attack. Every Internet host communicating with or attacking the victim can be viewed as a leaf in this DAG. In Fig. 2, H_1, H_3 and H_5 are legitimate clients, and H_2, and H_4 are attackers. $\{R_i\}_{1 \leq i \leq 5}$ are Internet routers, among which R_2 and R_3 form the perimeter. The "attack path" from H_i is the unique ordered list of routers from V to H_i, in the reversed order. For example, the attack path from H_2 to the victim is $H_2 \rightarrow R_4 \rightarrow R_2 \rightarrow R_1 \rightarrow victim$. Each edge on an attack path is referred to as an *infected edge* while all other edges are called *clean edges*.

3 Design of the Enhanced Probabilistic Module (EPM)

The proposed scheme extends and leverages on existing IP traceback schemes [3, 4, 5, 6]. In IP traceback, each Internet router inscribes, in a probabilistic way, a mark into a low-entropy IP header field (called "mark field" hereafter) set aside for this purpose. These marks *collectively* allow the victim to reconstruct the "attack graph", which consists of the network edges that the packets from the attackers have traversed. We can view the marked fields inscribed at the packet headers by IP traceback schemes as a *communication channel*. IP traceback uses this channel for only one purpose: to convey the encoded information about the "attack graph". Our scheme, on the other hand, splits this channel into the following two subchannels.

- One subchannel, called the *signaling subchannel*, will continue to carry the same information needed for IP traceback. It occupies about 5% of the channel bandwidth, i.e., only 5% of the marks are signaling marks. This implies that the reconstruction of the whole attack graph will be 20 times slower than in the underlying IP traceback scheme. Fortunately, the preferential filtering becomes very effective in improving the throughput of legitimate traffic as soon as a critical portion of the attack graph is reconstructed. As we will show, the nature of the probabilistic marking used in IP

```
for each packet pkt
  u := uniform(0,1)
  if (u < q) then
    v := uniform(0,1)
    if (v < r) then
      encode mark in a way suitable for IP traceback
      set flag to 1 to indicate signaling subchannel
      pkt.hopcount := 0
    else
      encode mark in a way suitable for preferential filtering
      set flag to 0 to indicate data subchannel
  else
    if (flag == 1) pkt.hopcount++
```

Figure 3: Marking procedure at an upstream router

traceback determines that it takes much less (less than 10% of the latter) packets/time to obtain this critical portion than obtaining the whole attack graph.

- The other subchannel, called the *data subchannel*, will consume the remaining 95% of the channel bandwidth. Information contained in the data subchannel, combined with the "attack graph" reconstructed from the signaling subchannel, will allow the perimeter routers to infer whether the packet is more likely to come from an attacker or a legitimate host. A packet will be preferentially filtered out or passed by the perimeter routers if it is determined to be more likely to come from the attackers or legitimate hosts, respectively.

The marks contained in packet headers will correspondingly be split into two types, namely, signaling marks and data marks. They carry signaling and data subchannel information respectively. The enhanced marking algorithm that performs such splitting is shown in Fig. 3. When a packet *pkt* arrives at a router, the router decides whether it will overwrite the current mark with marking probability q. If it decides to overwrite the mark, the router then decides whether this mark should be a signaling mark or a data mark, with probability

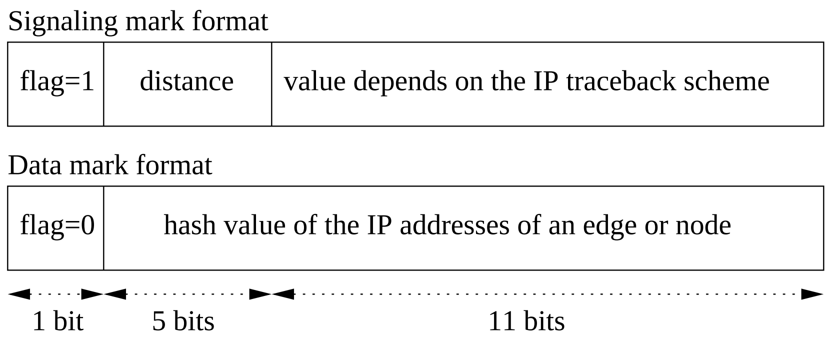

Figure 4: Mark format for signaling and data subchannels

r or $1 - r$, respectively. For a signaling mark, the hop count is maintained in same way as in IP traceback; it will be reset to 0 if the router overwrites the mark or simply incremented by 1 otherwise. For a data mark, there is no hop count field.

3.1 Encodings for Data and Signaling Marks

Fig. 4 shows the data formats of signaling and data marks. A mark, for both subchannels, will consist of 17 bits, one more bit than current IP traceback schemes [4, 5]. The first bit is a flag that indicates whether the mark is a signaling or a data mark. For a data mark, the remaining 16 bits will be the hash value of an edge or node. For a signaling mark, the remaining part will be further split into two parts. The first five bits encode the *hop count*, the distance in number of hops from the router where the mark is inscribed to the victim, provided that the mark is not overwritten on its way to the victim. The encoding for the remaining 11 bits varies from one IP traceback scheme to another. Note that the format for control marks follows the same convention as in IP traceback schemes [4, 5] for backward compatibility.

Fig. 5 shows how a 17-bit-long mark fits into an IP header. A mark occupies the IP fragmentation field (as suggested in [4] and also used by others [5]) and one reserved bit[3] in

[3]Its neighboring bits indicate whether the packet allows fragmentation (DF) and whether there are more

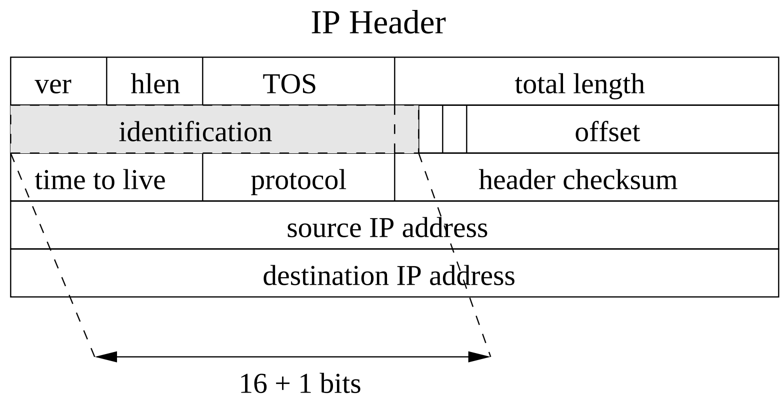

Figure 5: Encoding into the IP header

IP header. Note that we use one bit more than IP traceback for backward compatibility[4].

Since, to certain a degree, both data and signaling subchannels convey the information as to what edges a packet has traversed, a careful comparison is necessary. As the mark field is in generally small (typically 16 bits), it is not enough for holding the pair of IP addresses (that of two vertices) needed to represent a network edge. In the signaling subchannel, such an address pair needs to be broken down into several pieces using various coding techniques. For example, an infected edge X may be broken down into mark values a, b, c, and d, while a clean edge/node Y could be broken down to values a, b, e, and f. Such a breakdown brings accuracy to the traceback schemes (i.e., fewer false positives) [5], but it also introduces the following ambiguity for preferential filtering purposes. When a packet with mark value a or b arrives, it could come through either X or Y, making it hard to determine whether the packet should be passed (if from Y) or filtered out (if from X).

Therefore, the 16-bit field in the data subchannel encoding needs to "summarize" the whole identity of a network edge rather than a portion in order to eliminate such ambiguity.

fragments (MF).

[4]It is possible to squeeze everything into 16 bits, at the cost of less accuracy in traceback and preferential filtering. The level of deterioration is dependent on parameters such as the number of attackers and the network topology.

In our scheme, encoding using a hash function is used: $v := \mathcal{H}(e)$. Here $\mathcal{H}$ is a public uniform hash function, used by all Internet routers participating in the proposed scheme. e represents the edge "summarized" by the hash value v. Compared with perfect encoding of an edge (using the full 64 bits for encoding the IP address pair), such hash encoding causes only negligible loss of accuracy due to hash function collision, as shown by our simulation study.

3.2 Choice of the Underlying IP Traceback Scheme

Although the proposed scheme may leverage on any of the existing IP traceback schemes [2, 3, 4, 5, 6, 7], in this paper, we show how it builds on the Advanced Marking Scheme (AMS) proposed by Song and Perrig [5]. The advantage of the AMS is that it provides faster reconstruction and higher accuracy (hence fewer false positives in identifying attackers) than other IP traceback schemes, when there are more than one attackers. However, it assumes that the victim is able to obtain a map of upstream routers, which is a stronger (arguably less practical) assumption than used in other IP traceback schemes[5]. Our future research will study how our scheme works with other IP traceback techniques.

AMS [5] employs a technique similar to the Bloom filter [11] as follows. It uses 8 independent hash functions $\{\mathcal{H}_i\}_{1\leq i\leq 8}$ to encode network edges. When a packet goes through an edge e and the identity of the edge is to be marked, i will be chosen uniformly between 1 and 8, and the mark $i||H_i(e)$ is written into the IP header of the packet ($||$ representing concatenation). The reconstruction algorithm determines that an attacker has e on its path, if and only if, the algorithm has received attacking packets with at least k out of the 8 mark values $\{\mathcal{H}_i(e)\}_{1\leq i\leq 8}$. The tunable parameter k is between 6 and 8; larger k results in longer

[5]The practical issues in constructing and maintaining such a map are discussed in [5], based on the techniques proposed by Cheswick et al. [10].

"attack graph" reconstruction time but fewer false positives when identifying infected edges. We view this as a variant of Bloom filter since all the edges that an attacker has traversed can be viewed as a set, and it is represented by a bit array indexed by the values generated by these hash functions.

In our scheme, we do not consider the effect of faking the marks to make the victim "wrongfully convict" clean edges (to generate packets with marks $\mathcal{H}_1(e)$, $\mathcal{H}_2(e)$, ..., $\mathcal{H}_8(e)$ to make the victim convict a clean edge e). In IP traceback schemes, an attacker's ability to fake the mark is limited by the following two factors. First, due to the presence of the hop count field, it is impossible for an attacker to fake a mark with hop count smaller than its own [4]. Second, when most of the attackers are far away, the percentage of faked marks reaching the victim is very small (1% to 2% in average), given the nature of the probabilistic marking. Additionally, cryptographic solutions such as the authenticated version of the AMS, also by Song and Perrig [5], completely eliminate this problem. However, the scheme requires that authentication using Message Authentication Code (MAC) and time-released key chains are supported on all participating routers [5]. The ability of an attacker to cause "wrongful conviction" and the techniques to minimize this when authentication is not feasible will be a topic for further research.

In the rest of paper, we assume that AMS [5] is used as the underlying IP traceback scheme. In [5], trace-driven simulation is used to estimate the average number of packets that needs to be received by the victim in order to reconstruct the attack graph. However, results obtained through such simulation do not explain how this value changes when the parameters such as the network topology and the number of attackers vary. We derive the exact closed-form mathematical formula for calculating a closely-related metric: the average number of marks (denoted as n_l) that needs to be received from an attacker l hops away for the victim in order to reconstruct the attack path from this attacker. This metric is more accurate since it is independent of the topology of the network and the sending rate of other attackers. The formula for n_l is characterized by the following theorem.

Theorem 1 *Let q be the probability of writing a mark into the IP header by an upstream router and r be the probability for the mark to be a signaling mark. Let h be the number of independent hash functions $\mathcal{H}_i'$s used in [5] and k be the number of $\mathcal{H}_i(e)'s$ to "convict" an edge e. Then, the average number of packets n_l that needs to be received for the victim to reconstruct a path of length l (in number of hops) is:*

$$n_l = \frac{1}{r}\int_0^\infty \left[1 - \prod_{i=0}^{l}\left(\sum_{j=k}^{h}\binom{h}{j}(1-e^{-\lambda_i t})^j e^{-\lambda_i t(h-j)}\right)\right] dt \tag{1}$$

where $\lambda_i = \frac{q(1-q)^i}{h}, i = 0, 1, \cdots, l$

Proof (Sketch): This problem can be viewed as the following variant of the coupon collector's problem, in which each coupon has a different occurrence probability. On each path, marks representing all l edges need to be collected (i.e., there are l types of coupons). Upon the arrival of a mark, edge i (i hops away from the perimeter) will occur with probability $q(1-q)^i$. Each edge can have h different hash values (i.e., each coupon has h different colors). Also, the scheme proposed in [5] requires that **each type of coupon has more than k colors collected**, denoted as **condition (*)**. We denote as $N(l, h, k, q)$ (a random variable) the number of marks that needs to be collected for condition (*) to hold. The task is to find out $n_l = E[N(l, h, k, q)]$.

This problem can be solved using the following *stochastic embedding* method, i.e., embedding a discrete stochastic process into a continuous one such as Poisson. Consider a Poisson process of rate 1, in which coupons are drawn when an event in this Poisson process happens. We can see that in this way, the discrete coupon collection process is *embedded* into the continuous Poisson process. According to Wald's equality[6], $E[N(l, h, k, q)]$ is equal to the

[6]Wald's equality: given an integer random variable N, and N random variables $\{X_i\}_{1\leq i\leq N}$, each of which

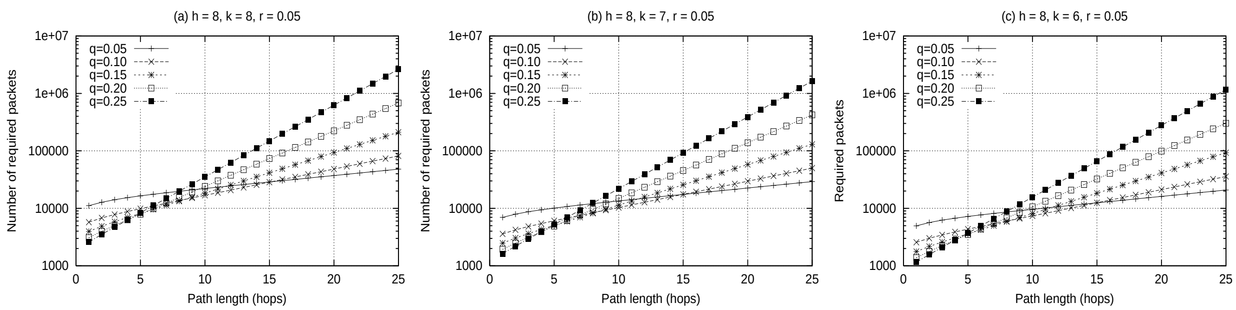

Figure 6: `Expected number of marks (Y-axis) needed for reconstructing a path of certain length (X-axis)`

average amount of time it takes to satisfy condition (*). According to the survivor representation of the expectation of a nonnegative random variable (i.e., $E[X] = \int_0^\infty P(X > x)dx$), the latter is equal to the integration of the probability that condition (*) has not been satisfied up to time t. It can be shown that this probability is exactly the integrand. Finally, the term $\frac{1}{r}$ comes from the fact that only r percent of the marks are signaling marks. $\square$

3.3 On Tuning the Parameter r

The parameter r (the percentage of marks being signaling marks) is set at a small number (5%) since, for the reason of "ambiguity" explained above, packets containing signaling marks generally cannot be *selectively filtered out* to improve the throughput of legitimate traffic. Therefore, for our scheme to work, the majority of packets should bear data marks.

In Fig. 6, we show the expected number of packets (Y axis is in log scale) that needs to be received for reconstructing an attack path of length l (X axis), with parameter k set to 8, 7, and 6, respectively. Recall that an edge is convicted if k marks out of the set $\{\mathcal{H}_i\}_{1 \leq i \leq 8}$ (here h is 8) are found in DDoS packets. It was shown in [5] that when the number of attackers

has the same distribution as a random variable X, then $E[\sum_{i=1}^{N} X_i] = E[N]E[X]$.

is no more than a few thousands, the false positive percentage is low enough even when k is 6, and improves when k is bigger. In each figure, there are five curves corresponding to different q parameters. It can be seen that, other parameters being equal, n_l increases with the k value. So a curve in Fig. 6a is always higher than the corresponding curve (with the same q value) in Fig. 6b, which is in turn higher than the corresponding curve in Fig. 6c. This represents the tradeoff between the false positives and n_l. However, the differences between these curves are small in the multiplicative sense. In the same figure, larger q values usually result in larger n_l values when the hop counts are reasonably large. This is because, when q becomes larger, it becomes harder to gather marks that are close to the attackers, since they are far away from the victim[7].

Our simulation results (shown in Section 5.2) indicate that the scheme works best when the q value is set to between 0.15 and 0.2. When $q = 0.15$, we can see from Fig. 6 that it would take between 92972 ($k = 6$) and 210911 ($k = 8$) packets to reconstruct a path of length 25. For a host attacking at 300 Kbps (considered typical broadband speed) sending 300 packets per second and 1000 bits per packet, this would take at least 300 to 700 seconds to gather all the marks needed for "attack graph" reconstruction. Fortunately, we do not need to construct the whole path to benefit tremendously from the preferential filtering scheme. Our simulation results also show (see Section 5.2) that as long as most of the infected edges that are within 15 hops from the victim are reconstructed, the proposed scheme can achieve the vast majority (over 98%) of the performance improvement. To construct a path of length 15 when $q = 0.15$, only between 18312 and 41531 packets from an attacker are needed. This will reduce the delay to between 60 and 140 seconds! In summary, when q is set to 0.15, it

[7]The converse is true when the hop count is very small (e.g., less than 8), since it becomes harder to gather marks that are close to the victim. This explains why the curves are pairwisely intersecting each other.

```
Task I (executed continuously in the background)
1.  Incrementally reconstruct the attack graph
2.  Measure incoming rates at the border gateway device

Task II (executed periodically)
1.  From Task I
2.     Obtain an attack graph ``snapshot'' and current incoming rates
3.  Based on this information
4.     Construct the hash bitmap corresponding to clean/infected edges
5.     Evaluate A_signaling, A_clean, and A_unsure
6.  Send the results obtained from steps 3--5 to perimeter routers
```

Figure 7: Attack Mitigation Decision-making (AMD) algorithm

takes very little time for the victim to construct this critical portion of the attack graph for our preferential filtering scheme to achieve most of the performance improvement.

4 Design of the AMD and PPF Modules

The function of the AMD (Attack Mitigation Decision) module is to leverage on an underlying IP traceback scheme to reconstruct the attack graph, once a DDoS attack has been detected. Based on this information, it makes the decision as to how the packets should be preferentially filtered at the perimeter routers. This decision will be transported to the perimeter routers, to be carried out by the PPF modules running on them. The functionality of the AMD module and the PPF module will be discussed in detail in Sections 4.1 and 4.2, respectively.

4.1 Design of the AMD Module

AMD modules are executed at the victim or the border gateway device on behalf of the victim. As shown in Fig. 7, the AMD module consists two tasks:

- **Task I:** This task is continuously executed in the background after a DDoS attack has been detected. It employs an existing IP traceback scheme such as [5] to reconstruct the attack graph, based on the signaling marks they have received. It also measures the current incoming rates to the victim during two probe periods (discussed in detail in Section 4.1.1), which will be used to adjust parameters for filtering.

- **Task II:** This task is executed periodically after the DDoS attack has been detected. At the beginning of each period, the most recent "snapshot" of the attack graph and the incoming rates during the previous period will be obtained from the task I. Based on this information, it decides with which probability packets containing a certain type of mark should be passed. Note these probabilities will change when the attack graph evolves and when the incoming rates changes. This decision will then be sent to the perimeter routers implementing PPF modules to be carried out. This task will be discussed in detail in Section 4.1.2.

4.1.1 Details of Task I

In Task I, the signaling marks will be processed by the underlying IP traceback scheme [5] to reconstruct the attack graph. Such a recent snapshot of the attack graph will be periodically probed and used by the Task II. In other words, the filtering can happen in parallel with the reconstruction process, and the benefit of filtering increases as the attack graph evolves. We will show through simulation that when all edges on the attack graph within perimeter 15 are recovered, 98% of the potential performance improvement can be achieved. As we have discussed in Section 3.3, it takes only 60 to 140 seconds to reconstruct this critical portion.

As we will show, adjusting the filtering parameters in Task II require the measurement of the incoming traffic rates. The measured values will be reset and recalculated periodically by Task I, after it is probed by Task II.

4.1.2 Details of Task II

Task II decides the probabilities with which packets carrying various "mark types" are passed. The marks contained in the packets are classified into the following three types:

- (I) Signaling marks that carry information for IP traceback.
- (II) Data marks that corresponds to the hash value of a "clean" edge.
- (III) Data marks that correspond to the hash value of an "infected" edge[8] or are left empty (all-zero). The latter happens when no router on the path inscribes a mark on the packet due to the probabilistic nature of the marking algorithm[9].

Corresponding to these three mark types, the following three probability values are computed:

- $A_{signaling}$: the probability of passing packets with type I marks.
- A_{clean}: the probability of passing packets with type II marks
- A_{unsure}: the probability of passing packets with type III marks.

The following Knapsack algorithm is executed to decide these three probabilities:

1. At the beginning of the attack, a target bandwidth of $B_{signaling}$ (initially set to 50% of the available bandwidth at the victim) is set to allow packets bearing type I marks to pass. This allows the "attack graph" to be reconstructed (by Task I) as quickly as possible. As more and more edges of the attack graph are reconstructed, this

[8]Hash collisions may happen, but as we have discussed, they make negligible differences on the metrics we are interested in.

[9]This happens very rarely when the path is reasonably long. In the real Internet topology, the percentage of such packets is about 1% to 2% when the marking probability is set to 0.15.

percentage can be gradually reduced to 20% of the victim bandwidth. The value of $A_{signaling}$ should be set such that, when each perimeter router passes packets bearing type I marks with probability $A_{signaling}$, the total incoming traffic bearing type I marks is equal to $B_{signaling}$. $A_{signaling}$ is set to 100% when no such value exists.

2. Like in the last step, set the target rate for the packets with marks of type II to the remaining bandwidth (50% at the beginning and eventually becomes 80%) at the victim. The value of A_{clean} should be set such that either this target bandwidth is exactly reached or $A_{clean} = 100\%$.

3. If there are still bandwidths left after steps 1 and 2, set parameter A_{unsure} so that the remaining bandwidth can be filled by packets bearing marks of type III.

The scheme needs to adapt to the changes in attack intensity and the availability of new information such as the evolution of the "attack graph". This is done through the following iterative algorithm ((2)–(4)) that is executed periodically. Task I periodically measures the incoming rates of the traffic bearing the aforementioned three types of marks, denoted as $\tilde{R}_{signaling}$, $\tilde{R}_{clean}$, and $\tilde{R}_{unsure}$, respectively. All these three probabilities $A_{signaling}$, A_{clean}, and A_{unsure} are initially set to 100%. Based on the values (A^{prev}) measured during the previous time period, the values to be set during the new time period (A^{next}) are:

$$A^{next}_{signaling} = min(1, A^{prev}_{signaling} * B_{signaling}/\tilde{R}_{signaling}) \tag{2}$$

$$A^{next}_{clean} = min(1, A^{prev}_{clean} * B_{clean}/\tilde{R}_{clean}) \tag{3}$$

$$A^{next}_{unsure} = min(1, (1+\epsilon) * A^{prev}_{unsure}, A^{prev}_{unsure} * B_{unsure}/\tilde{R}_{unsure}) \tag{4}$$

Note that in iterations (2)–(4), the parameter values are probabilities and therefore should never exceed 1. These iterations allow the parameters quickly adapt to a resource allocation

```
for each packet pkt
  if pkt contains a signaling mark then
    pass pkt with probability A_signaling
  else /*pkt contains a data mark*/
    index:=H(pkt.src_IP)
    if (bitmap[index] == CLEAN) then
      pass pkt with probability A_clean
    else
      pass pkt with probability A_unsure
```

Figure 8: Preferential Packet Filtering (PPF) algorithm

"point" where the victim bandwidth is fully utilized and this bandwidth is partitioned among perimeter routers according to the knapsack-filling priority as specified above. In (4), ϵ is a small positive constant (e.g., 0.05), which ensures that A_{unsure} is incremented by no more than a percentage per iteration. This prevents the adversary from causing severe system oscillation by varying the value $\tilde{R}_{unsure}$ significantly from one iteration to another.

In addition, the AMD module needs to compute a hash bitmap M of 64K entries (corresponding to 16 bit hash values), which encodes the "attack graph", as follows. Let S be the set of edges that are known to be infected so far (obtained in Task I). Then $M[i] := 1$ if $i \in \{\mathcal{H}(e) | e \in S\}$, and $M[i] := 0$ otherwise. Note that more entries of M are set to 1 when the attack graph evolves.

The new parameters calculated from (2)–(4) and the hash bitmap will then be transported to the perimeter routers to be used for preferential packet filtering. Clearly, a communication protocol is needed to disseminate such information to perimeter routers securely. A protocol similar to those proposed in [9] may be adapted for this purpose. This is an important issue to be addressed in our future research. It is, however, outside the scope of this paper.

4.2 Design of the Preferential Packet Filtering (PPF) Module

Preferential packet filtering is carried out at the perimeter routers based on the filtering parameters ($A_{signaling}$, A_{clean}, and A_{unsure}) and the hash bitmap received from the AMD module. Figure 8 presents the filtering algorithm in detail. Its operation is straightforward: passing packets with type I, II, and III marks with probabilities $A_{signaling}$, A_{clean}, and A_{unsure}, respectively. Note that the execution of the algorithm incurs negligible overhead at perimeter routers: one hash function computation plus one table lookup.

5 Performance Evaluation

Extensive simulation studies have been conducted on three real-world network topologies to evaluate the effectiveness of the proposed scheme in improving the throughput of the legitimate traffic. Simulation results demonstrate that such throughput can be improved by three to seven times during an attack.

5.1 Simulation Set-up: Topologies and Metrics

The following three real-world network topologies are used in our simulation study:

- Cheswick's Internet topology dataset – Cheswick et al. [10] have designed a software tool to construct the routes from an origin host to other hosts across the Internet. The results are recorded in six Internet topology databases. In our simulation, we merged them into one and trimmed out incomplete routes (in which end hosts have not been reached). The final data has the complete routes from an origin (a Bell Labs host) to 86,813 other hosts.

- Skitter data I (aroot.paths.20011128) – Using the software tool designed by Cheswick et al. [10], CAIDA (Cooperative Association for Internet Data Analysis) has collected topology originated from a CAIDA-owned host (a-root.skitter.caida.org) on 11/28/2001 [12]. This data contains the traceroute data from this server to 192,900 destinations. The data collection process is a part of CAIDA's Skitter project [12].

- Skitter data II (eroot.paths.20011127) - This is traceroute data from another CAIDA host (e-root.skitter.caida.org) to 158,181 destinations collected on 11/27/2001, also as a part of the Skitter project [12].

Note that all three topologies are routes from a single origin to multiple hosts on the Internet. In our simulations, we assume that this origin is the victim and the attackers and legitimate clients are randomly distributed among the destination hosts in the topologies. We fix the number of legitimate clients to 250, and the number of attackers vary as other parameters vary.

Since it will be clear that the simulated metric of our scheme is not dependent on the actual values of the parameters concerning bandwidth and data rates but rather on their relative ratios, we will use abstract "units" for characterizing these parameters instead of the actual rates. We assume that each legitimate user sends at the rate of 1 unit per second. The bandwidth of the victim site is set to 250 units so that it is 100% utilized by legitimate traffic when there is no attack. For performance evaluation purposes, we also assume that each attacker attacks at the same rate (not needed for the correctness of our scheme). The number of attackers will be a function of the severity of the attack and the bandwidth per attacker, to be discussed later.

Table 1 shows the performance metrics and control parameters used in our simulation. The good traffic percentage (GTP) is the metric we would like to improve, which is the arrival rate of the legitimate traffic at the victim divided by the victim bandwidth. GDR

Performance Metrics		GDR (Good Drop Ratio): the percentage of the legitimate traffic dropped
		BDR (Bad Drop Patio): the percentage of the DDoS traffic dropped
		GTP (Good Traffic Percentage): the percentage of the traffic arriving at the victim being legitimate
Control Parameters		q: The marking probability at upstream routers
		R_p: The perimeter radius
		g: The percentage of generated traffic being legitimate
		b: The ratio of sending rate from a legitimate host to the sending rate from an attacker

Table 1: Performance Metrics and Control Parameters

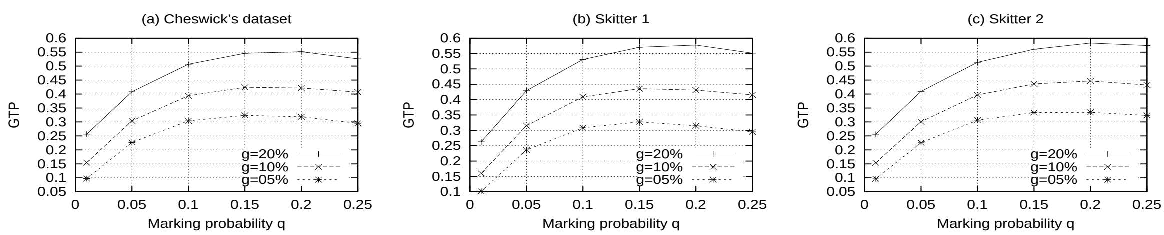

Figure 9: GTP (Good Traffic Percentage) on three topologies

and BDR represent the percentage of legitimate traffic and DDoS traffic that are dropped by the PPF module, respectively. Note that when there is no protection, GDR is equal to BDR since upstream routers would have to drop legitimate and DDoS traffic indiscriminately.

Among the control parameters, q denotes the marking probability at upstream routers. R_p is the perimeter radius (the distance from each router on the perimeter to the victim). The parameter g represents the percentage of incoming traffic being legitimate and b is the ratio of the rate of an attacker over the rate of a legitimate host. Note that when g and b are set, the number of attackers is calculated as $\frac{250*(1-g)}{b*g}$ (recall that there are 250 legitimate hosts).

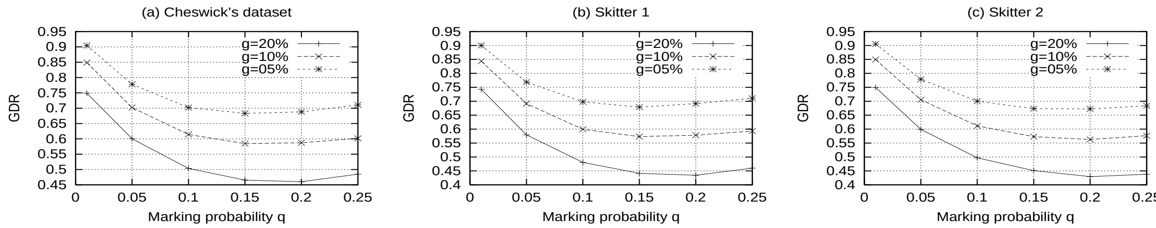

Figure 10: GDR (Good Drop Ratio) on three topologies

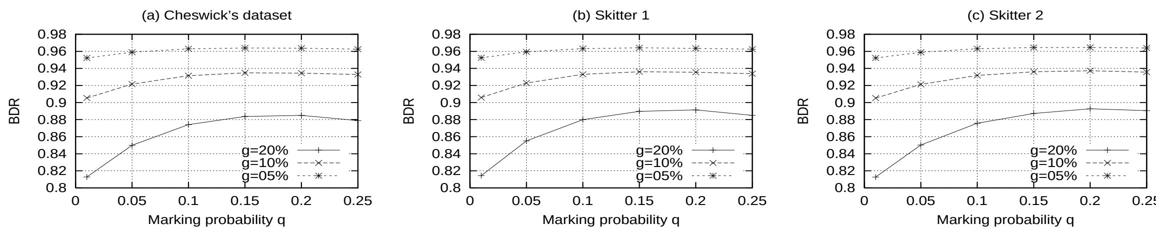

Figure 11: BDR (Bad Drop Ratio) on three topologies

5.2 Simulation Results

Figure 9(a)–(c) shows GTP as a function of the marking probability q, under the three aforementioned Internet topologies. Here perimeter radius R_p is set to 2, and b is set to 20. There are three curves in each figure, corresponding to three different g values (5%, 10% and 20%). In other words, if there is no DDoS defense mechanism, the throughput of the legitimate traffic would only be 5%, 10% and 20% of the total victim bandwidth, respectively. In these three situations, the rate of DDoS traffic is 4750, 2250 and 1000 units

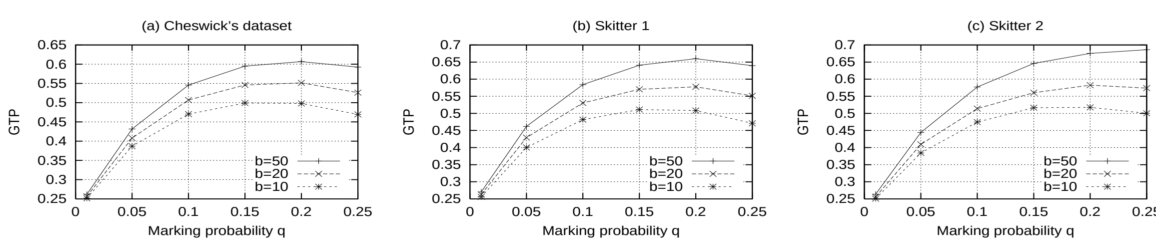

Figure 12: GTP (Good traffic Percentage) by varying parameter b

per second, respectively. Since $b = 20$, it can be computed that the number of attackers in these three situations are 237, 112 and 50, respectively. All three figures indicate that the best improvement on the legitimate traffic throughput is achieved when q is set to between 0.15 and 0.2. We can see from Fig. 9(a) that when $q = 0.15$, GTP is improved from 5%, 10%, and 20%, to 32.4%, 42.4% and 54.6%, respectively. This represents an improvement of three to seven times. Figures 9(b) and 9(c) demonstrate improvements of similar magnitudes, under the other two topologies. Note that in all three figures, the GTP results have accounted for the hash collisions (hence the false positives) in the data mark field.

The reason behind such an improvement is illustrated in Figures 10(a)–(c) and Figures 11(a)–(c), which show the corresponding GDR and BDR values respectively. In Fig. 10a, we can see when q is set to 0.15, the percentages of legitimate traffic being filtered are 68.3%, 58.5%, and 46.5% respectively, whereas these percentages would be 95%, 90%, 80% if there were no DDoS defense. Also, in Fig. 11a, we can see that the percentages of DDoS traffic being filtered are higher than if there were no defense, at 96.4%, 93.5% and 88.4%, respectively. Figures 10(b)–(c) and Figures 11(b)–(c) tell similar stories. Since a much higher percentage of the DDoS traffic is dropped than legitimate traffic, the resulting traffic mix contains a much higher percentage of legitimate traffic.

Figures 12(a)–(c) show how GTP varies as the parameter b varies. Recall that b represents the ratio of the sending rate of an attacker over that of a legitimate host. Here, the perimeter radius R_p is set to 2 like above and g is set to 20% (i.e., GTP would be 20% if there is no DDoS defense). The three curves in each figure represents the GTP values when the parameter b is 10, 20 and 50, respectively. We can see that GTP curve is lower when b is lower. This is because, when b is lower, the number of attackers (as g is fixed) becomes larger and these attackers are more densely distributed in the topology. This increases the average percentage of edges on a legitimate host's path to be infected, resulting in the increasing percentage of

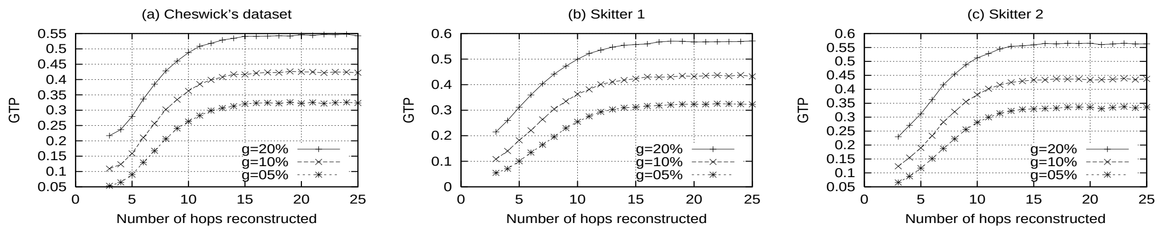

Figure 13: GTP (Good Traffic Percentage) with incremental path reconstruction

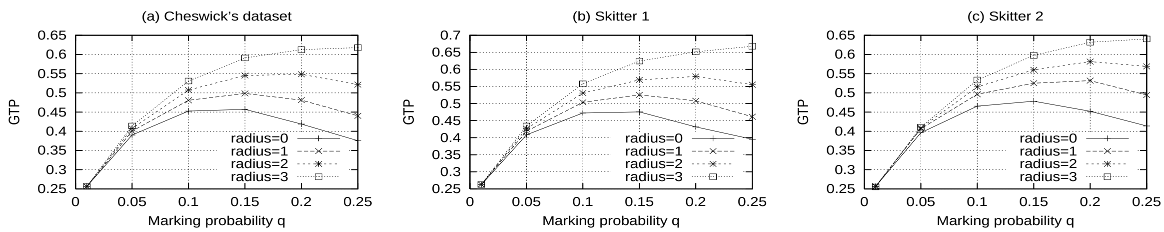

Figure 14: GTP (Good traffic Percentage) by increasing perimeter radius

the legitimate traffic to be filtered out due to "guilt by association" (sharing part of a path with an attacker).

Figures 13(a)–(c) show how GTP improves when "infected edges" within a given radius are reconstructed. Here R_p and q are set to 2 and 0.15, respectively. We can see that in all three figures, the curves (corresponding to three g settings) become almost flat after the radius reaches 15. This confirms the claim we have made in Section 3.3 that the vast majority (over 98%) of the performance improvement is attained when infected edges within radius 15 are reconstructed.

Figures 14(a)–(c) show how GTP improves when the perimeter radius R_p increases. In Fig. 14a, when q is set to 0.15 as above and g is set to 20%, GTP values are 45.8%, 49.7%, 54.6%, and 59.1% respectively, when the perimeter radius size is set to 0, 1, 2, and 3, respectively. The other two figures also show that GTP values increase when the perimeter radius becomes larger. Our explanation for this is the following. When perimeter radius becomes larger, all edges become "closer" to the victim, and for each clean edge, there is a higher probability for its mark to go through. Much better results can be obtained when the perimeter radius becomes even higher. However, the number of routers on a perimeter increases dramatically with the size of perimeter radius, shown in Table 2. Recall that the victim needs to communicate with all perimeter routers for the proposed scheme to work properly. Therefore, there is a tradeoff between the performance improvement and the amount of coordination between the victim and the perimeter routers. We find that a perimeter size of 2 or 3 represents a reasonable tradeoff between these two factors.

Hop from the Victim	Cheswick's topology	Skitter I	Skitter II
1	6	2	1
2	24	18	61
3	34	46	224
4	197	240	588

Table 2: Number of perimeter routers as a function of perimeter radius

6 Potential Applications of the Proposed Scheme

While the proposed scheme is able to dramatically improve the throughput of legitimate traffic, a high percentage of legitimate traffic may still have to be dropped as "collateral damage". At such a high packet loss probability, the performance of a TCP flow will suffer considerably. Nevertheless, when coupled with other DDoS defense techniques, such an improvement can be very important and useful:

- Under a DDoS attack, a web server may be able to forward the service of a legitimate client to a remote web server that is not directly affected by the attack. Such service redirections have been used extensively in content distribution networks (e.g., Akamai

[13]). Even at such a high packet loss probability, this short redirection process (taking a couple of packets) can be finished before the client gives up the web transaction.

- One of our companion work on DDoS defense is a web-server-based DDoS mitigation scheme [14]. By employing a cryptographic cookie-based [15] HTTP redirection technique, it is able to provide service to a large percentage of legitimate clients, even during severe DDoS attacks. The technique is able to protect almost all the traffic in a legitimate web transaction from DDoS attacks, except for its very first TCP SYN packet. As shown in [14], the effectiveness of the technique is constrained by (i.e., as an increasing function of) the "success rate" of this SYN packet. Since the proposed scheme is able to significantly enhance this success rate, it can further improve the effectiveness of the web-server-based mitigation technique considerably.

In summary, although the proposed scheme alone may not be sufficient for maintaining normal quality of service during heavy attacks, it becomes very effective when used in combination with other DDoS defense techniques.

7 Related Work

Internet Denial of service incidents started to be reported frequently after 1996 [16]. The most popular type of DoS attack is the TCP SYN flood attack [17], and cryptographic [15, 18] and noncryptographic [19, 20] solutions have been proposed. Recent large-scale distributed DoS attacks have drawn considerable attention [1]. Most of the proposed solutions focus on IP traceback [8, 2, 3, 4, 5, 6, 7], that is, to trace the origin(s) of an attack. Their common approach (except [2, 7]) is to require upstream routers to write, with a certain probability, a mark into a very-infrequently-used IP header field. The mark is an encoding of the identity of an intermediate router or edge. Upon the receipt of the packets from attackers, the victim decodes the marks to reconstruct the node/edges that the packets have traversed. While the traceback schemes are valuable in finding the exact location of the attacker and (hopefully) punishing the hacker after the attack, they are in general not able to mitigate the effect of a DoS attack while it is raging on.

Techniques to mitigate the effect of distributed DoS attacks have been proposed in [9], [21], and [22]. The type of DDoS attacks targeted in [21] is that attackers send bogus traffic aggressively using their real IP addresses. Their technique is to isolate traffic sent by aggressive IP addresses, from other traffic sources. However, this scheme is vulnerable to the forms of DDoS attacks in which source IP addresses are spoofed. Schemes in both [9] and [22] use router throttles to allocate the victim bandwidth equally (in [9]) or in a min-max fashion ([22]) among perimeter routers. The drawback of these schemes is that since none of them can distinguish between traffic from legitimate clients and DDoS traffic, the improvement in the throughput of legitimate traffic may be small when the DDoS traffic "contaminates" the incoming traffic mix at the perimeter routers evenly. In comparison, the proposed scheme will perform much better, given the same topology and perimeter radius[10], than these schemes, since it is able to make such a distinction.

Other proposals attack the DoS problem by calling for the tightening of "global security." Ingress filtering [23] at every ISP is recommended to detect and drop packets sent using spoofed IP addresses. It is unclear whether these schemes are likely to be practical in the near future since they require global cooperation. Park [24] proposes to install packet filters at autonomous systems in the Internet to filter packets traveling between them. It is shown in [24] that when 20% of strategically chosen autonomous systems install such filters, most of the packets with randomly generated IP address (usual sense of IP spoofing) can be

[10]Note that the perimeter radius size for the scheme proposed in [22] for the scheme to be effective is much larger (e.g., 8) than the typical radius for our scheme to be effective (e.g., 1+)

dropped. However, this requires the cooperation of thousands of autonomous systems, every ingress/egress router of which has to install the filter. Nevertheless, the attacker can still spoof IP addresses, albeit within a much smaller domain (e.g., a few autonomous systems).

8 Conclusions

This paper presents IP-traceback-based preferential packet filtering, a novel scheme to defend against Internet DDoS attacks. It leverages on the "attack graph" information obtained through IP traceback, and uses such information to preferentially filter out packets that are more likely to come from attackers. Our simulation studies based on real-world Internet topologies demonstrate that the scheme is very effective in improving the throughput of legitimate traffic during DDoS attacks. The operations required of routers (probabilistic marking) are fully in line with the operations of IP traceback, making the scheme as practically deployable as possible.

References

[1] L. Garber. Denial-of-service attacks rip the internet. *IEEE Computer*, 33(4):12–17, April 2000.

[2] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proc. Usenix LISA 2000*, December 2000.

[3] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to ip traceback. In *Proc. NDSS 2001*, pages 3–12, February 2001.

[4] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. In *Proc. ACM SIGCOMM 2000*, pages 295–306, August 2000.

[5] D. Song and A. Perrig. Advanced and authenticated marking schemes for ip traceback. In *Proc. Infocom 2001*, April 2001.

[6] T. Doeppner, P. Klein, and A. Koyfman. Using router stamping to identify the source of ip packets. In *Proc. ACM CCS-7*, pages 184–189, November 2000.

[7] A. Snoeren, C. Partridge, et al. Hash-based ip traceback. In *Proc. ACM Sigcomm'2001*, August 2001.

[8] S. Bellovin. Internet draft: Icmp traceback messages. Technical report, Network Working Group, March 2000.

[9] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. Technical report, ACIRI and AT&T Labs Research, February 2001.

[10] B. Cheswick. Internet mapping. Available at http://cm.bell-labs.com/who/ches/map/dbs/index.html, 1999.

[11] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the Association for Computing Machinery*, 13(7):422–426, 1970.

[12] Caida's skitter project web page. Available at http://www.caida.org/Tools/Skitter/.

[13] Akamai technologies inc. http://www.akamai.com/.

[14] J. Xu and W. Lee. Sustaining availability of web services under severe denial of service attacks. *IEEE Transaction on Computers, special issue on Reliable Distributed Systems*, February 2003. to appear.

[15] P. Karn and W. Simpson. *Photuris: Session-Key Management Protocol.* IETF RFC 2522, March 1999.

[16] J. Howard. *An Analysis of Security Incidents on the Internet.* PhD thesis, Carnegie Mellon University, August 1998.

[17] CERT. Tcp syn flooding and ip spoofing attacks. Advisory CA-96.21, September 1996.

[18] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. of NDSS'99.* Internet Society, March 1999.

[19] C. Schuba et al. Analysis of a denial of service attack on tcp. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1997.

[20] Checkpoint Inc. Tcp syn flooding attack and the firewall-1 syndefender. Available at http://www.checkpoint.com/products/firewall-1/syndefender.html, 1997.

[21] F. Kargl, J. Maier, S. Schlott, and M. Weber. Protecting web servers from distributed denial of service attacks. In *WWW-10*, May 2001.

[22] David K.Y. Yau, John C.S. Lui, and Feng Liang. Defending Against Distributed Denial-of-Service Attacks with Max-min Fair Server-centric Router Throttles. In *Proc. of IEEE IWQoS*, May 2002.

[23] P. Ferguson. *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing.* RFC 2267, January 1998.

[24] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *Proc. ACM Sigcomm'2001*, August 2001.