Distributed divide-and-conquer techniques for effective DDoS attack defenses

M. Muthuprasanna*
Google, Inc.
Mountain View, CA - 94043
muthup@google.com

G. Manimaran
Iowa State University
Ames, IA - 50011
gmani@iastate.edu

Abstract

Distributed Denial-of-Service (DDoS) attacks have emerged as a popular means of causing mass targeted service disruptions, often for extended periods of time. The relative ease and low costs of launching such attacks, supplemented by the current woeful state of any viable defense mechanism, have made them one of the top threats to the Internet community today. While distributed packet logging and/or packet marking have been explored in the past for DDoS attack traceback/mitigation, we propose to advance the state of the art by using a novel distributed divide-and-conquer approach in designing a new data dissemination architecture that efficiently tracks attack sources. The main focus of our work is to tackle the three disjoint aspects of the problem, namely attack tree construction, attack path frequency detection, and packet to path association, independently and to use succinct recurrence relations to express their individual implementations. We also evaluate the network traffic and storage overhead induced by our proposed deployment on real-life Internet topologies, supporting hundreds of victims each subject to thousands of high-bandwidth flows simultaneously, and conclude that we can truly achieve single packet traceback guarantees with minimal overhead and high efficiency.

1. Introduction

Denial-of-Service (DoS) attacks pose a significant threat to today's Internet. The first widely reported attacks in early 2000, on Yahoo, Ebay, Amazon, etc. had seriously affected Internet operations then. Subsequent attacks on root DNS servers, and others motivated by political and economic reasons on SCO, RIAA, 2Checkout, BlueSecurity, etc. have established a disturbing trend. The growing sophistication of the attacks and the increasing complexity of the Internet architecture, have rendered many previous DDoS attack defenses unviable. Nowadays, automatic attacking tools such as TFN, Trinoo, stacheldraht, etc. allow just about anyone to launch widely distributed DoS attacks with just a few keystrokes [1]; providing not only precise targeting, but also a whole range of attack options. Symantec recently reported that a network of 5500 zombies could be rented for just $350 weekly [2], and launching DDoS attacks is thus no more constrained due to technical know-how and steep costs. The stateless nature and destination-oriented Internet routing, with no restrictions on spoofed sources, have additionally made the tracing of the true origins of the attack sources a hard problem. An ideal DDoS attack defense mechanism for the Internet, should not only enable immediate and precise identification of the true attack sources/perpetrators, but also aid in effective mitigation of the ongoing attacks with minimal collateral damage. With a broad consensus among researchers that an intelligent network is probably the best option, we now need to design a secure, scalable, incrementally deployable architecture, that requires minimal changes to the existing infrastructure.

The research on DDoS attack defenses has proceeded mainly along two lines: *Attack Traceback* and *Attack Mitigation*. The Spoofer Project [3] reveals that approximately one-quarter of all the Autonomous Systems (ASes) in the Internet permit either partial or full spoofing; thus attack attribution is not always easy and straight-forward. *Attack Traceback* addresses the problem of collecting information about individual packet forwarding agents and collating this data to obtain an approximate Internet router-level graph (attack tree rooted at the victim); whereby tracing the routing path that any packet has taken, provides sufficient basis for attack attribution (attack tree leaves). Attack traceback is necessary for cleansing zombie attackers, while also being of critical forensic value to law enforcement.

Attack Mitigation, on the other hand, addresses the problem of scaling down the impact of the ongoing attacks in real-time, using a variety of filtering mechanisms. For DDoS flooding attacks, it is essential to obtain per-path packet rates (attack frequency) to classify a certain traffic

*This author is currently pursuing his Ph.D. at Iowa State University.

source as attack/legitimate. In addition to constructing this frequency-weighted attack tree, we also need to address the crucial filter placement problem [4]. Although placing a single central filter closest to the victim reduces its effective traffic volume, it still considerably strains the network resources, possibly leading to collateral damage to other co-located services. The ideal strategy of deploying ingress filters at each subnet connected to the Internet is also impractical, given the limited support that current networks offer. An optimal filtering strategy would thus place the few available filters at appropriate locations in the entire network, exploiting the attack traffic convergence characteristics evident in the frequency-weighted tree.

In this paper, we identify three independent issues underlying all DDoS attack defense mechanisms, namely attack tree construction, attack path frequency detection, and packet to path association. We then propose a novel *distributed divide-and-conquer* approach to the problem, that represents their individual implementations as succinct *recurrence relations*. While quantifying the operational overhead by performance evaluation on real-life Internet topologies, we also evaluate the feasibility of realistically providing single packet traceback guarantees in the Internet.

2. Related Work

Early Approaches: The earliest work on DDoS defense led to the concept of network traceback [5] by Burch and Cheswick, where they inferred the attack path by flooding all links with large bursts of traffic and measuring the perturbation in the attack traffic. Stone proposed CenterTrack [6], that automated this traditional *input debugging* mechanism for route-inference, by re-routing attack traffic over a specialized overlay network architecture. Bellovin [7] proposed iTrace, a low volume ICMP-based out-of-band messaging channel for the victim to detect packet *audit trails*. The use of explicit network support was first proposed by Snoeren et.al. [8], where the packet audit trails (small-sized packet hashes) were stored in a distributed manner in the network itself, in the form of efficient Bloom Filter data structures. Li et.al. [9] improved the performance of *packet logging* techniques by using sampling techniques, thereby significantly reducing the network resources required to support packet logging for a large number of victims under heavy traffic loads.

Attack Traceback: Belenky and Ansari [10] proposed the use of in-band *deterministic packet marking* (DPM), while optimizations based on local small-worlds were proposed in [11] and [12]. However, these techniques led to the space explosion problem, due to the large number of bits needed per packet to represent all the intermediate routers. Savage et.al. [13] proposed *probabilistic packet marking* (PPM) as a means of generating in-band audit trails, that employed node/edge sampling to reduce the bit-size per packet while spreading the fragments probabilistically across multiple packets. Song and Perrig [14] improved the security of PPM schemes by using an authentication technique for data integrity, while Dean et.al. [15] proposed another novel coding scheme using an algebraic approach to embed path information. Yaar et.al. [16] proposed the use of a 1-bit distance tracking field and an offline map construction technique, while Adler [18] performed an exhaustive theoretical evaluation of traceback. Recently various hierarchical designs employing distributed graph coloring for a scalable design [19], and the use of a hybrid of packet marking and logging [20] have also been evaluated, in addition to the use of various efficient encoding techniques such as Huffman Codes [21], etc. Additionally, various interesting approaches such as highly efficient enhancements to the base PPM scheme [17], the use of offense or compulsive attack volume inflation [22], and the use of bit-level granularities to optimize audit trail detection [23], have also been proposed in literature.

Attack Mitigation: In addition to DDoS attack traceback, the search for an optimal DDoS attack defense mechanism has also focused a great deal on attack prevention and mitigation. The use of ingress/egress filtering [24] has long been proposed as the ideal solution to the DDoS attack problem, but has achieved very little Internet-wide deployment. Park and Lee [25] extend ingress filtering to the core of the Internet, while Li et.al. [26] have proposed the SAVE protocol for Internet-wide source address validation. The use of path filtering has been proposed in Hop-Count filtering [27], and Secure Overlay Service [28], while the notion of *path fingerprints* was explored in [29]. Researchers have also explored the use of statistical filtering in PacketScore [30], history-based filtering schemes [31], and filtering strategies based on rate limiting using max-min fairness [32] and congestion control [33]. In [34], the authors present a detailed survey of all known DDoS defense mechanisms.

3. Motivation

The three fundamental operations underlying any attack traceback mechanism are attack tree construction, attack path frequency detection, and packet to path association. Traditional traceback/mitigation techniques proposed in the literature provide no clear *separation of duties* with respect to these requirements, with often the same feature satisfying many of them simultaneously, either implicitly or explicitly, and often in a sub-optimal manner.

Attack tree construction is defined as the process of obtain-

ing an abstraction of the router-level Internet graph, called the attack tree, where the attack victim is the tree root, and the different traffic sources (or their egress routers) are the many tree leaves. It has traditionally been achieved by requiring the different intermediate routers to probabilistically/deterministically send in-band/out-of-band edge information, and then using a path reconstruction algorithm at the victim that collates all these different fragments (packet marks) to reconstruct the original attack tree. The use of multiple packets by the individual routers, and the potential namespace collision across multiple routers, makes the path fragments unreliable, thereby leading to inefficient, possibly incorrect, attack tree construction.

Attack path frequency detection is defined as the process of obtaining a weighted attack tree, where each edge is annotated by the number of packets that traversed that network link, in the specified time period. This metric often serves as a differentiator while classifying traffic sources as attack/legitimate, and has traditionally been measured indirectly by counting the number of path fragments from each intermediate router. However, the probabilistic nature of data collection, and the unreliability of the underlying attack tree constructed, often lead to high false positives/negatives in attack source attribution.

Packet to path association is defined as the process of associating any data packet to a particular traffic source (and hence a routing path) in the attack tree. A path has traditionally been defined as the ordered set union of path fragments of all the intermediate routers. However, fragmentation across multiple packets requires multiple buffers at the victim to accumulate the path information, while the lack of efficient inter-packet fragment correlation often leads to incorrect packet to path association.

Thus the use of path fragments leads to sub-optimal performance and also possibly unreliable behavior, while its *centralized* design makes it difficult to harness incremental distributed network support. Our major contribution here has been to show that addressing these issues as three disjoint problems, not only reduces the complexity of their individual implementations, but also achieves higher efficiency across a wide variety of metrics. We propose to use single packet in-band path identifiers for unique packet to path association, while handling attack tree construction and attack path frequency detection as independent out-of-band processes. The use of a novel distributed divide-and-conquer approach also makes it easily extensible for implementing future distributed network defenses and early warning systems, in an incremental fashion.

As an illustration, we now show that handling packet to path association independently makes efficient use of in-band packet marking with a highly compact representation. We model a sample attack tree $T(n, l, d, p)$ as a reference, where (n, l, d, p) represent the total number of routers (tree nodes), the average hop length (tree depth), the average router degree (tree node degree), and the average number of routing paths (tree leaves) respectively. An approximate mapping of the modeled tree to a *balanced k-ary tree* [35] is shown in Eqns. 1, 2, while assuming Internet-measured values of $d \approx 4^+$, $l \approx 16^+$ for analysis [11], [36].

$$p = d^l \tag{1}$$

$$n = \frac{d * (d^l - 1)}{d - 1} = \frac{\sqrt[l]{p} * (p - 1)}{\sqrt[l]{p} - 1} \tag{2}$$

$$Mark_{\ global} = l * \lceil log_2(n) \rceil \approx 512\ bits \tag{3}$$

$$Mark_{\ local} = l * \lceil log_2(d) \rceil \approx 48\ bits \tag{4}$$

$$Mark_{\ proposed} = \lceil log_2(p) \rceil \approx 32\ bits \tag{5}$$

The use of globally unique IP addresses (or their hash fragments) as router identifiers, leads to a large namespace scatter as Internet end-hosts far outnumber the routers. Until recently, we have used an ordered set union of all unique intermediate router identifiers to identify a routing path (Eqn. 3). The recent use of multiple local small-worlds that requires this unique mapping only within some closed domain by employing lazy path discovery [11], also ensures unique routing path identification (Eqn. 4). We propose to reduce the scope of the problem here by assigning unique path identifiers to only the different traffic sources or tree leaves, as every routing path uniquely maps to some traffic source (Eqn. 5), thereby achieving a high degree of compactness. We thus obtain significant gains by de-coupling and addressing these different issues independently.

4. Proposed Approach

We now present our novel data dissemination architecture consisting of three independent modules, namely attack tree construction, attack path frequency detection, and packet to path association. We additionally employ a distributed divide-and-conquer strategy to represent these different modules using simple recurrence relations. We also discuss how the proposed architecture makes intelligent use of the three different layers for data management, namely out-of-band packet marking, in-band packet marking, and network/router storage, for effective DDoS attack defense.

4.1. Attack Tree Construction

The attack tree rooted at the victim is essentially an abstraction of the Internet router-level graph based on

instantaneous packet flows to the victim (where path insertion needs to be immediate, but path removal can possibly be lazy), and hence remains static over moderately short intervals of time. Stated differently, it suffices to refresh the attack tree either infrequently or in an interrupt-driven fashion, where triggering happens only when some critical structural modification occurs. Thus, overloading in-band packet marking to handle attack tree construction, would lead to unnecessary *repetitive transmissions* on a per-packet basis. Additionally, the resulting packet size explosion would necessitate a fragmentation scheme, thereby breaking any feasible single packet traceback guarantees. Hence we propose to use out-of-band packet marking as the preferred means of data transmission for the attack tree. We also avoid the use of independent communication channels between the victim and the intermediate routers as in [7], as it is grossly inefficient not only due to possible redundant transmissions, but also due to the inability of the intermediate routers deploying tracers/filters to infer their attack sub-trees without expensive computations.

Recursive Approach: We propose to use a *distributed divide-and-conquer* approach by recursively breaking down the problem at each router into multiple sub-problems, each in turn handled by that router's neighbors (tree children) respectively. The solutions to the sub-problems are then combined and propagated up the attack tree from the traffic sources to the victim. Thus we adopt a *bottom-up* approach rather than the traditional *top-down* approach controlled by the victim. If an intermediate router assigns unique labels to all its immediate children, then the maximum value of the local identifier is at most its degree in the attack tree. Each router then aggregates the attack sub-trees (T_{R_i}) of its neighbors (children), and forwards it to its immediate upstream neighbor. This when implemented by every router in the attack tree, leads to an incremental attack tree evolution in a bottom-up yet distributed fashion.

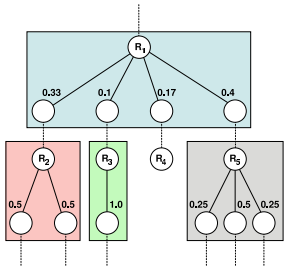

Figure 1. Modular Path Tree

Logical Representation: Consider an abstraction of the attack tree as shown in Fig. 1, showing the attack sub-tree of some router R_1, having 4 different tree children, namely R_2, R_3, R_4, and R_5. The logical representation of this sub-tree is then given by Eqn. 6, where D_{R_i} and T_{R_i} represent the degree and attack sub-tree of router R_i respectively. Eqn. 7 then generalizes this expression for every router in the attack tree, thus representing the proposed distributed divide-and-conquer approach as a succinct *recurrence relation*, where Δ_{R_i} represents the immediate children of R_i.

$$T_{R_1} = D_{R_1} \cup T_{R_2} \cup T_{R_3} \cup T_{R_4} \cup T_{R_5} \tag{6}$$

$$T_{R_i} = D_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j} \tag{7}$$

Thus the proposed attack tree construction employing distributed divide-and-conquer techniques, ensures that each router not only performs minimal computation with no redundant messaging, but also obtains a global view of its entire attack sub-tree for (early) incremental filtering.

Physical Representation: The attack sub-tree of router R_1 is expressed as in Eqn. 8, where $\circ$ represents further recursive expansion not shown due to abstraction. Interestingly, if we tag every node in the attack tree with its degree, then Eqn. 8 also represents the *pre-order traversal* (prefix notation) of the attack tree. In [37], the authors discuss the standard technique used to reconstruct the original k-ary tree, from its (prefix) *Polish Notation*, if the arity of all the intermediate nodes are known. As every attack tree node is tagged with its degree information, any intermediate router (or victim) can thus easily reconstruct the unique attack sub-tree structure from its pre-order traversal.

$$T_{R_1} = \underline{4} \cup \underline{2 \circ \circ} \cup \underline{1 \circ} \cup \underline{0} \cup \underline{3 \circ \circ \circ} \tag{8}$$

The power of this recurrence relation lies in its modularity. Any structural modification to the attack tree thus supports a simple *plug-n-play* design that can propagate up the tree to the victim, without needing a complete re-computation of the entire attack tree or affecting other independent attack sub-trees, as shown by different shaded regions in Fig. 1. Thus we can closely model the dynamic Internet routing characteristics, by periodic or triggered update messages containing only the attack sub-trees that have been structurally modified. It is to be noted in this context, that no other scheme in literature provides more robust and explicit support for dynamic changes to the attack tree, without complete re-transmission of the attack tree.

Tree Pruning: The maximum size of the attack tree rooted at the victim is represented by Eqn. 9. For high traffic portals such as Google, Yahoo, etc., the attack tree size can potentially reach unmanageable levels. Hence, we propose a *tree pruning* technique ($Prune_{R_i}$) to dynamically reduce the attack tree size to more manageable levels. As the tree size grows linearly with n and hence with p, we choose to limit the number of actively tracked traffic sources in the attack tree, to bound its size to more practical limits. Eqn. 10

thus represents the new form of the *recurrence relation* with pruning possibly supported at multiple intermediate routers in the network. The distributed nature of the pruning mechanism enables individual service/network providers to use independent custom strategies to ensure local optimality in tree pruning. A few parameters that can potentially impact its design include: active path frequency or utilization, feedback from the victim (service subscribers), feedback from distributed monitoring systems, round-robin dropping of randomly chosen paths, white/black listing using local signature databases, etc.

$$TreeSize_{\ max} = n * \lceil log_2(d) \rceil \tag{9}$$

$$T_{R_i} = Prune_{R_i}\left(D_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j}\right) \tag{10}$$

The deployment of tree pruning techniques is completely optional and does not adversely affect the proposed attack tree construction technique. However harnessing it provides significant gains, as the maximum tree size for monitoring 1 million packet (attack traffic) sources simultaneously, is only $\approx$ 507KB. Note that $p = 1,000,000$ and $l \approx 16^+ \Rightarrow d \approx 2.4$, in the modeled attack tree $T(n, l, d, p)$.

4.2. Packet to Path Association

We propose to use both in-band packet marking and minimal network/router storage, for establishing unique packet to path association. As the attack tree is built in a *bottom-up* fashion, each router has a snapshot of only its attack sub-tree, and not the global view of the entire attack tree rooted at the victim. Hence, it is infeasible for the intermediate routers to predict the global path identifier for any packet in the attack tree rooted at the victim.

Recursive Approach: We exploit the recursive nature of the proposed distributed divide-and-conquer approach to address this critical issue. Each intermediate router marks only the local (not global) unique path identifier in its attack sub-tree for a particular packet, and its upstream router during aggregation then translates this in-band packet marking audit trail to its own local namespace of unique path identifiers. Thus the different path identifiers evolve from some local namespace to a globally unique namespace, as they propagate through each intermediate router from the tree leaf to the victim in the attack tree. We propose to use simple hash lookup tables (L_{R_i}) at each intermediate router for per-hop namespace translation of the path identifiers (P_{R_i}). Eqn. 11 thus expresses the path identifiers as *recurrence relations*, while Eqn. 5 bounds the maximum size of the different path identifiers in the victim's attack tree.

$$P_{R_i} = L_{R_i}(P_{R_j}), \quad R_j \in \Delta_{R_i} \tag{11}$$

Consider a packet received by router R_1 from router R_3 (Fig. 1) with an in-band packet mark of k. Then assuming that the attack sub-tree of router R_2 has t unique paths, the new path identifier at router R_1 would now be $(t + k)$. Thus each intermediate router stores the initial offset for any packet received from each of its immediate children in a local hash lookup table, whose size is linearly dependent on its degree. It is to be noted that when used in conjunction with the tree pruning algorithm discussed previously, we need to provide an extra layer of indirection for appropriate *namespace scaling*. In Fig. 1, if router R_1 prunes the attack sub-tree of router R_2 from t to t' unique paths, $(t + k)$ would lead to unnecessary namespace scatter and potential size explosion due to inefficient utilization. Hence optimal namespace utilization is achieved by translating k to $(t'+k)$ instead of $(t+k)$. Thus we see that unique packet to path association with single packet traceback guarantees and no false positives/negatives is easily achieved using the proposed recursive approach to namespace translations.

5. Attack Path Frequency Detection

During a highly disruptive distributed DoS attack, it is probably easy to weed out persistent high volume attackers. However, DDoS attacks employing large botnets with a low median traffic volume per source, often make it difficult to classify packet sources as legitimate or those with malicious intent. A traffic *hot-spot* due to a sudden spike in Internet activity geared towards a particular destination (say, *Slashdot effect*), also leads to an unintended denial-of-service due to unprovisioned capacity problems, both at the end-host and in the network. Thus the problem of DDoS attack defense reduces to that of prioritizing certain traffic flows/sources over others, the granularity of such demarcation being determined by the destination under heavy traffic load. Any such traffic classification would require both packet-level metrics (content signature, protocols/flags in use, etc.), and flow/path-level metrics (frequency, flow pulse characteristics, source subnet, etc.). As packet-level metrics can easily be gleaned off the packets themselves, we focus on the flow/path-level metrics here.

5.1. Frequency Measurement

Simple path frequency detection using active measurement requires just one counter per path in the attack tree, an increment being triggered on receipt of a packet associated with that path (counter). Hence, frequency detection on a per-packet granularity can easily be achieved at the victim, as guaranteed by unique packet to path association.

Similarly, we can employ active frequency measurement for proactive attack tree pruning and also for monitoring at-

tack sub-trees at intermediate routers in the network. Fig. 1 shows a sample measured frequency distribution for the attack sub-tree of router R_1. Although active frequency measurement serves as a simple tool, a large number of path frequency counters need to be deployed at multiple nodes in the attack tree for it to be effective. Additionally, the aggregate nature of frequency measurement implies that every link (edge) in the attack tree is also implicitly measured by each of its upstream routers, which seems grossly redundant. Hence, we now propose a path frequency inference model, which not only significantly reduces the number of these distributed counters, but also ensures that each link is monitored/measured only once in a recursive (evolutionary) fashion, and in a light-weight manner.

5.2. Frequency Inference

The attack tree we have obtained thus far using out-of-band packet marking, is essentially an *attack path tree*, embedding only the router connectivity information. We propose to overload this attack path tree to also embed path frequency information, to construct a novel *attack path frequency tree*. Along similar lines of the proposed distributed divide-and-conquer tree construction mechanism, we now *encode* path frequency distributions as simple tree data structures, and then *embed* them into the original attack tree, to yield a novel *bottom-up attack path frequency tree construction* mechanism. Although any frequency encoding and embedding techniques could be used, we evaluate Huffman Encoding [38] and *Balanced Parentheses* embedding [39] here, as an example.

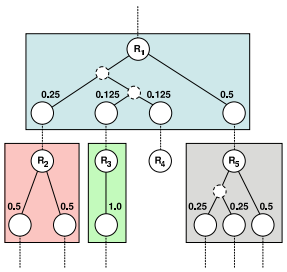

Figure 2. Modular Path Frequency Tree

Frequency Encoding: Huffman encoding [38] is an entropy encoding data compression algorithm that assigns variable-length prefix-free codes to symbols so as to approximately match code lengths with the probabilities of the symbols themselves. Consider three symbols with probabilities 0.4, 0.3, 0.3 respectively. While a naive compression technique would map them to the binary representations 00, 01, 10 respectively, the Huffman code would map them to 0, 10, 11 respectively, so as to achieve a smaller average output size for transmitting a large number of these symbols. For a sorted set of m symbol probabilities, the linear-time Huffman code generator uses a *bottom-up tree construction* mechanism to map the different symbols to efficient binary codes ($(2m$-1)-node binary tree), rounding the given probabilities to closest negative powers of two. Huffman codes can thus structurally represent any rounded frequency distribution as a binary tree, and vice-versa. Thus Huffman codes provide a means of translating the path frequency distribution at any router into an equivalent binary tree representation.

Path Frequency Embedding: Every intermediate router in the attack tree deploys path frequency counters to measure the number of packets it has received from each of its immediate children in the attack path tree, the total number of counters now being linearly dependent on that router's degree only, and not on the total number of paths in its attack sub-tree. The periodic snapshots of the frequency distribution are then run through a on-board Huffman encoder to generate the corresponding frequency-encoded Huffman binary trees. Finally, the edges linking any router to its immediate children in the attack path tree are replaced by these time-varying Huffman trees (H_{R_i}), for efficient frequency embedding to obtain the attack path frequency tree. Fig. 2 shows the frequency embedding for the attack tree in Fig. 1, with the path frequencies rounded off to the closest negative power of two at each depth in the tree. Eqn. 12 then represents the modified *recurrence relation* for the bottom-up attack path frequency tree construction mechanism.

$$T_{R_1} = H_{R_1} \cup T_{R_2} \cup T_{R_3} \cup T_{R_4} \cup T_{R_5} \qquad (12)$$

Physical Representation: The ordering that the Huffman encoding imposes in every iteration might potentially vary from the original ordering of its immediate children for any router (see router R_5 in Figs. 1, 2 with different ordering of its children), and hence the physical representation of the attack path frequency tree should be capable of correctly associating the different attack sub-trees it has received and the computed frequency counts, with its immediate children in the attack path tree. We propose to use the Balanced Parentheses optimization here for both strict ordering and a compact representation.

In [39], the authors propose an optimal tree representation, needing just $2m$ bits for compactly representing a m-node tree structure. This optimal representation is obtained by a *pre-order* tree traversal, producing a "(" while visiting a node for the first time, and a ")" while visiting the node after completely visiting its sub-tree. The two parentheses can easily be replaced by bits 0 and 1 respectively, for a compact $2m$-bit tree structure representation. For a node-labeled tree, the authors propose to also use a pre-order traversal of the node labels, easily represented in $m*log(m)$ bits. Thus, the information-theoretic lower bound for representing a

node-labeled binary tree is $\approx m*(2+log(m))$ bits [39]. We propose to use the node-labeled balanced parentheses for explicitly ensuring correct association and ordering of immediate children, for every intermediate router. The physical representation of the attack sub-tree of router R_1 is thus expressed as in Eqn. 13, where $\circ$ represents further recursive expansion not shown due to abstraction.

$$\begin{aligned} T_{R_1} &= \underline{((()(()()))()) \ 1234} \\ &\cup \underline{((())) \ 12 \ \cup \circ \cup \circ} \\ &\cup \underline{(()) \ 1 \ \cup \circ} \\ &\cup \underline{()} \\ &\cup \underline{((()())()) \ 132 \ \cup \circ \cup \circ \cup \circ} \end{aligned} \quad (13)$$

The maximum attack path frequency tree size is shown in Eqn. 14, while Eqn. 15 represents the generalization of the bottom-up attack path frequency tree construction mechanism at each intermediate router with optional tree pruning, as a succinct *recurrence relation*, where Δ_{R_i} represents the immediate children of R_i.

$$TreeSize_{\ max} = n * (d * (4 + \lceil log_2(d) \rceil)) \quad (14)$$

$$T_{R_i} = Prune_{R_i}\left(H_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j}\right) \quad (15)$$

The bottom-up path frequency computation thus helps us infer tree edge frequencies at upstream routers, as measured by some downstream router in the attack tree. Thus we achieve efficient frequency detection at each router with far fewer path frequency counters, at the expense of minor increase in the attack tree size, and minor perturbation in inferred frequencies due to Huffman rounding. Thus the proposed data dissemination architecture provides realistic single packet traceback guarantees, with incremental attack tracing/filtering and a truly distributed implementation.

5.3. Illustration

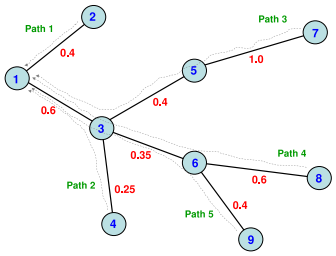

Figure 3. Sample Attack Tree

We now illustrate the working of the proposed scheme at each of the intermediate routers in the attack tree. Consider a sample attack tree as in Fig. 3, where the node and edge

Router	Attack Path Tree	Attack Path Frequency Tree
R_1	$2 \cup R_2 \cup R_3$	$((())) \ 12 \cup R_2 \cup R_3$
R_2	0	()
R_3	$3 \cup R_4 \cup R_5 \cup R_6$	$((()())()) \ 132 \cup R_4 \cup R_6 \cup R_5$
R_4	0	()
R_5	$1 \cup R_7$	$(()) \ 1 \cup R_7$
R_6	$2 \cup R_8 \cup R_9$	$((())) \ 12 \cup R_8 \cup R_9$
R_7	0	()
R_8	0	()
R_9	0	()

Table 1. Path Tree & Path Frequency Tree

Path 2			Path 4			
R_1	R_3	R_4	R_1	R_3	R_6	R_8
2	1	0	4	3	1	0

Table 2. Path Identifiers

labels indicate the different intermediate routers and the actual traffic distribution respectively. The path labels are derived based on the natural (say, sorted) ordering imposed by any intermediate router on its immediate children. Table 1 illustrates the path tree and path frequency tree representations at each of the intermediate routers, while Table 2 illustrates the in-band path identifiers as they evolve across different depths, for packets along paths 2 and 4.

6. Performance Evaluation

We now evaluate the feasibility and the potential overhead associated with an Internet-wide deployment of the proposed approach, as measured on real-life Internet topologies. We analyze the three different layers for data management, namely out-of-band packet marking, in-band packet marking, and network/router storage, by measuring the *attack path (frequency) tree size*, the *unique path identifier size*, and the *router lookup table size* respectively as the evaluation criteria in our analysis.

We evaluate multiple attack scenarios (Fig. 7) tracking variable number of attack sources at each router, namely different tree pruning limits of 1k, 2k, 4k, 8k, 16k, 32k, 64k, 128k and unlimited flows (packet sources) at each tree depth. We also capture both the average (A) and maximum (M) values for all the metrics at different tree depths, as they realistically indicate the *utilization* and *provisioning* requirements respectively. Finally, we define an attribute called *estimate* that shows the average value of these metrics across all the different tree depths. We have assumed, in this estimation, that an intermediate router has an equal probability of being present at any of the different tree depths, when viewed globally for all the potential attack victims in the Internet. Although this assumption might seem inaccurate, it helps us realistically *estimate* different benchmarks for any router in today's Internet.

We now use two real-life Internet topologies for our per-

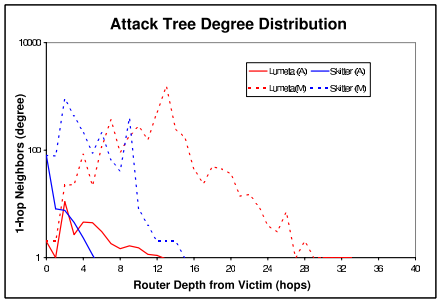

Figure 4. Degree Distribution

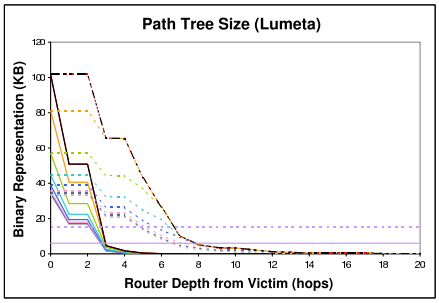

Figure 5. Freq. Measure (L)

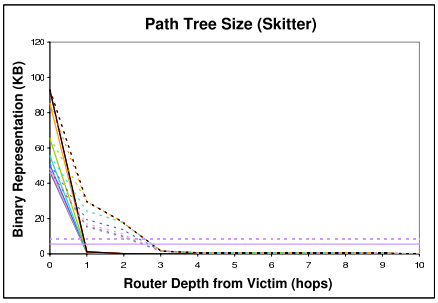

Figure 6. Freq. Measure (S)

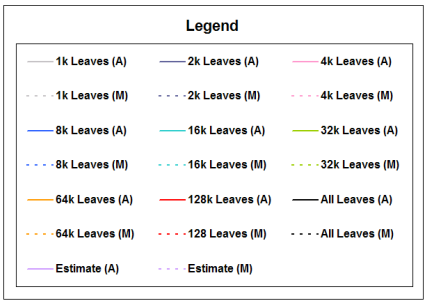

Figure 7. Attack Scenarios

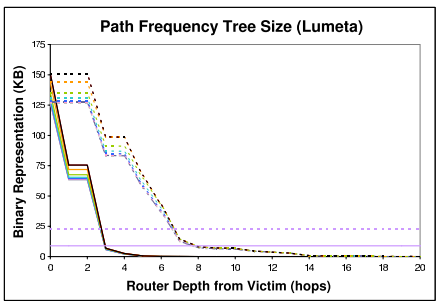

Figure 8. Freq. Infer (L)

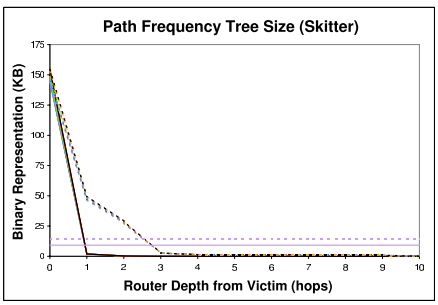

Figure 9. Freq. Infer (S)

formance evaluation: the datasets obtained from CAIDA's Skitter [40] and Lumeta's Internet Mapping projects [41]. Table 3 shows the statistics for the attack trees obtained from these different sources. In Fig. 4, we show the average and maximum router degree at each depth in the attack tree, for the two datasets. While characterizing the high variability in the degree distribution, we also notice that the Skitter dataset shows high clustering near the victim, while the Lumeta dataset shows a more uniform distribution.

Topology Source	Total Routers	Degree (avg/max)	Total Leaves	Depth (avg/max)
Lumeta	208332	1.78 / 1533	91294	13.33 / 34
Skitter	190112	1.67 / 896	76144	6.22 / 16

Table 3. Attack Tree Statistics

Out-of-band Traffic: Depending on the frequency detection model being used, Figs. 5, 6 represent the periodically transmitted attack path tree size, while Figs. 8, 9 represent the periodically transmitted attack path frequency tree size, for the two topologies respectively. We see that the average values at each tree depth closely model the maximum values at each of those depths. We also notice the *Long Tail* phenomenon, indicating the large number of packet sources, and also the relative sparsity of routers at depths closer to the packet source (rather than the victim). Also distinctly visible is the effect of the different attack tree pruning limits at multiple intermediate routers. While the maximum attack tree size is $\approx$ 100 & 150KB (highest at depth 0) for the two models respectively, its average *estimate* for any router is a mere $\approx$ 15 & 25KB respectively. Thus the average data transmission per router per victim in every refresh interval, is a few kilobytes of control plane traffic, for multiple gigabytes of data plane attack traffic.

In-band Traffic: We use in-band packet marking for establishing unique packet to path association in our proposed approach. Figs. 10, 11 suggest that the attack tree is dense for the top one-third of its depth, and is significantly sparse for further depths from the victim. An in-band packet marking field of 17 bits thus suffices to track 100k packet sources, while 20 bits could track upward of a million sources, thereby realistically providing single packet traceback guarantees in today's Internet. As the different packet marking schemes in the literature use varying number of packets each with different bit-sizes for in-band marking, we propose to use the total number of bits marked in all those packets for obtaining a certain attack path detection probability [13], as our evaluation criterion. Fig. 12 shows the total in-band data transmission for path detection probabilities of 50% and 95%, for PPM [13], FIT [16], EPM [17], TPM [11], Huffman [21], and our proposed scheme. It is to be noted that both TPM and Huffman additionally require a pushback mechanism, while our proposed scheme requires an out-of-band periodic ($\frac{1}{100k}$ packets) transmission of the attack path (frequency) tree.

Thus we see that our proposed approach not only pro-

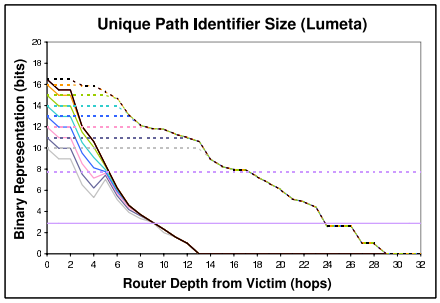

Figure 10. Packet Mark (L)

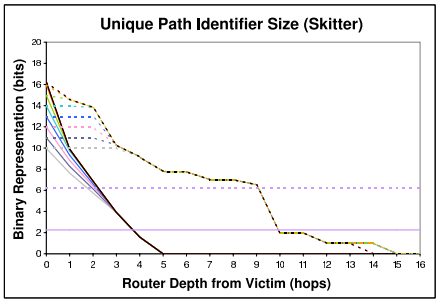

Figure 11. Packet Mark (S)

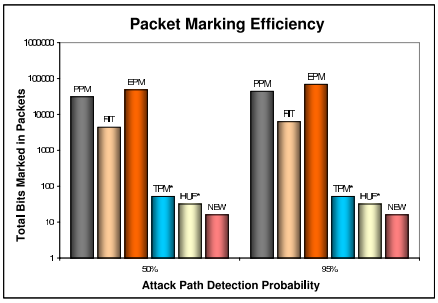

Figure 12. Packet Marking

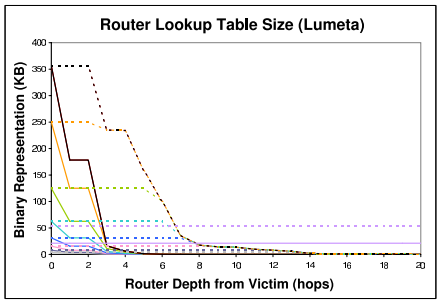

Figure 13. Router Log (L)

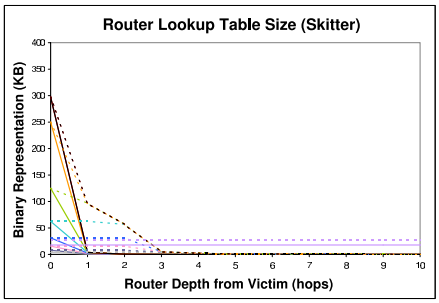

Figure 14. Router Log (S)

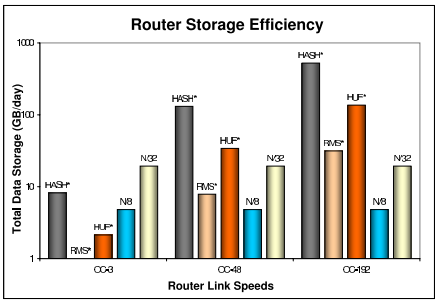

Figure 15. Router Storage

vides single packet traceback guarantees with no false positives/negatives, but also achieves multiple orders of magnitude improvement with respect to network traffic (aggregate of in-band and out-of-band) overhead, over other well-known schemes in the literature.

Router Storage: We use a router hash lookup table for namespace translation to ensure correctness of the unique packet to path association. Figs. 13, 14 represent the router lookup table size at each depth in the attack tree, for the two topologies respectively. While the maximum lookup table size is $\approx$ 4KB, its average *estimate* for any Internet router is just $\approx$ 0.4KB. As any router needs to store both the lookup table and the attack path (frequency) tree in local memory, we now compare the total router storage requirements of the proposed scheme with other well-known packet logging schemes, namely HASH [8], RMS [9], Huffman [21]. In Fig. 15, we notice that the router storage requirements for the other schemes are dependent on the link speeds, and also the duration for which they are cached. The proposed scheme however depends only on the number of victims being simultaneously supported (100k in Fig. 15) and the individual tree pruning limits (8k and 32k in Fig. 15); thus providing greater scalability and flexibility by being link speed and time agnostic, unlike other well-known schemes.

The above analysis defends 100k different victims each targeted by 32k independent high-bandwidth DDoS attackers simultaneously, and this far exceeds the maximum number of parallel attack flows that other well-known schemes can handle. We can further use *statistical multiplexing* of the router storage across multiple victims, thereby easily achieving more than an order of magnitude improvement over other well-known defense strategies.

We thus see that our proposed approach employing *separation of duties*, while using a bottom-up tree construction mechanism with optional tree pruning, and utilizing both packet marking and packet logging paradigms, provides significant improvement over other well-known schemes in the literature. Most importantly, it realistically provides single packet traceback guarantees in today's Internet.

7. Conclusions

The steady evolution of distributed denial-of-service (DDoS) attacks as a vehicle for achieving political, economic and commercial gains, and the relative ease, low costs, and limited accountability in launching such attacks, have rendered them one of the top threats to today's Internet services. Although various independent DDoS attack prevention, mitigation, and traceback techniques have been proposed, their relative uptake has been minimal at best, due to the lack of a robust, fool-proof, and universal DDoS attack defense mechanism. In this paper, we propose a new data dissemination architecture in advancing the state of the art in DDoS attack traceback and mitigation. We look at the

problem of DDoS attack defense as three disjoint issues, namely attack tree construction, attack path frequency detection, and packet to path association, and address them independently in a locally optimal manner. We also propose a novel distributed divide-and-conquer approach to represent their individual implementations as succinct recurrence relations. Using performance evaluation on real-life Internet topologies, we show that we can realistically provide single packet traceback guarantees for a large number of victims under heavy traffic loads simultaneously, with very high efficiency and practically no false positives/negatives.

8. Future Work

We now discuss a few critical issues that must be addressed before the proposed approach can become practically viable to deploy in today's Internet. Due to space constraints, we limit our discussion to just a broad outline of the different issues and our current approach in tackling these problems. The issues of incremental deployment and scalability determine the viability of any new technique, and we propose to use the concept of *black-holes* in the attack tree to address these concerns. The tree nodes that hide more information than they actually reveal would be tagged as black-holes. Any closed logical boundary of routers when abstracted to form a single attack tree node (black-hole), such that all internal routers are legacy routers while all the peripheral routers support the proposed scheme, can easily solve the incremental deployment problem. The attack tree size explosion due to the growth of the Internet, can also be addressed by defining multiple pseudo-victims in the original attack tree, such that they appear as black-holes to the original victim, while they launch their own internal DDoS attack defense mechanism on their (black-holed) attack sub-tree. Finally, the crucial black-hole placement problem can be solved as a special instance of the filter placement problem [4].

Although the proposed technique does adapt reasonably well to frequently changing paths and unpredictable routing dynamics, we could potentially address this issue better by temporarily black-holing every node under routing transition to abstract out the dynamics momentarily. The potential gains achieved due to this optimization need to be studied more carefully. Various other performance related issues such as choosing optimal frequency encoding and embedding techniques, avoiding repeated instances and propagation of similar attack sub-trees due to Internet route changes, etc. also need to be addressed independently.

References

[1] CERT Advisory CA-2000-01, "Denial-of-Service developments", *http://www.cert.org/advisories/ca-2000-01.html*, 2000.

[2] Joris Evers, "Hacking for Dollars", *http://news.zdnet.com/2100-1009_22-5772238.html*, 2005.

[3] R. Beverly, S. Bauer, "The Spoofer Project: Inferring the extent of Source Address Filtering on the Internet", *USENIX SRUTI*, 2005.

[4] Chun-Hsin Wang et.al., "Tracers placement for IP Traceback against DDoS Attacks", *ACM IWCMC*, 2006.

[5] H. Burch, B. Cheswick, "Tracing Anonymous Packets to their approximate source", *USENIX LISA*, 2000.

[6] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods", *USENIX Security Symposium*, 2000.

[7] S. M. Bellovin, "ICMP Traceback Messages", *Internet Draft: draft-bellovin-itrace-00.txt*, 2000.

[8] A. Snoeren et.al., "Single-Packet IP Traceback", *IEEE/ACM Trans. on Networking*, 10(6), pp. 721-734, 2002.

[9] J. Li et.al., "Large-Scale IP Traceback in High-Speed Internet", *IEEE Symp. on Security & Privacy*, 2004.

[10] A. Belenky, N. Ansari, "IP Traceback with Deterministic Packet Marking", *IEEE Communication Letters*, vol. 7(4), 2003.

[11] B. Al-Duwairi, T. Daniels, "Topology-based Packet Marking", *ICCCN*, 2004.

[12] M. Muthuprasanna, G. Manimaran, "Space-Time Encoding Scheme for DDoS Attack Traceback", *IEEE GLOBECOM*, 2005.

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

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

[15] D. Dean et.al., "An Algebraic Approach to IP Traceback", *ACM TISSEC*, 5(2), pp. 119-137, 2000.

[16] A. Yaar, A. Perrig, D. Song, "FIT: Fast Internet Traceback", *IEEE INFOCOM*, 2005.

[17] M. Goodrich, "Efficient Packet Marking for Large-Scale IP Traceback", *ACM CCS*, 2002.

[18] M. Adler, "Tradeoffs in Probabilistic Packet Marking for IP Traceback", *STOC*, pp. 407-418, 2002.

[19] M. Muthuprasanna, G. Manimaran, M. Alicherry, V. Kumar, "Coloring the Internet: IP Traceback", *IEEE ICPADS*, 2006.

[20] B. Al-Duwairi, G. Manimaran, "Novel Hybrid Schemes employing Packet Marking & Logging for Traceback", *IEEE TPDS*, vol. 17(5), 2005.

[21] K. H. Choi, H. K. Dai, "A marking scheme using Huffman codes for IP Traceback", *IEEE ISPAN*, 2004.

[22] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, S. Shenker, "DDoS Defense by Offense", *ACM SIGCOMM*, 2006.

[23] M. Muthuprasanna, G. Manimaran, Z. Wang, "Unified Defense against DDoS Attacks", *IFIP NETWORKING*, 2007.

[24] P. Ferguson, D. Senie, "Network ingress filtering: Defeating denial of service attacks with employ source address spoofing", *RFC 2267*, 1998.

[25] K. Park, H. Lee, "On the Effectiveness of Route-Based Packet filtering for DDoS Attack Prevention in Power-Law Internets", *SIGCOMM*, 2001.

[26] J. Li, J. Mirkovic, M. Wang, M. Reiher, L. Zhang, "SAVE: Source address validity enforcement protocol", *IEEE INFOCOM*, 2001.

[27] C. Jin, H. Wang, K. Shin, "Hop-Count Filtering: An effective defense against spoofed DDoS traffic", *ACM CCS*, 2003.

[28] A. Keromytis, V. Misra, D. Rubenstein, "SOS: An architecture for mitigating DDoS attacks", *IEEE JSAC*, vol. 22(1), 2004.

[29] A. Yaar et.al., "StackPi: New Packet Marking Filtering Mechanisms for DDoS & IP Spoofing Defense", *IEEE JSAC*, pp. 1853-1863, 2006.

[30] Y. Kim et.al., "PacketScore: A statistical-based overload control against DDoS attacks", *IEEE INFOCOM*, 2004.

[31] T. Peng, C. Leckie, K. Ramamohanarao, "Protection from DDoS attacks using history-based IP filtering", *IEEE ICC*, 2003.

[32] D. Yau, J. Lui, F. Liang, "Defending against DDoS attacks with max-min fair server-centric router throttles", *IEEE IWQoS*, 2002.

[33] J. Ioannidis, S. Bellovin, "Implementing Pushback: Router-based defense against DDoS attacks", *NDSS*, 2002.

[34] T. Peng et.al., "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems", *ACM Computing Surveys*, vol. 39(1), 2007.

[35] T. Cormen, C. Leiserson, R. Rivest, C. Stein, "Introduction to Algorithms", *MIT Press*, 2001.

[36] A. Fei, G. Pei, R. Liu, L. Zhang, "Measurements on Delay and Hop-Count of the Internet", *IEEE GLOBECOM*, 1998.

[37] C. Hamblin, "Translation to and from Polish Notation", *Computer Journal*, vol. 5, pp. 210-213, 1962.

[38] D. A. Huffman, "A method for the construction of minimum redundancy codes", *IRE 40*, vol. 10, pp. 1098-1101, 1952.

[39] V. Arya, T. Turletti, S. Kalyanaraman, "Encodings of Multicast Trees", *IFIP Networking*, 2005.

[40] CAIDA, "Router-level Topology Measurements", *http://www.caida.org/tools/measurement/skitter/router_topology/*, 2003.

[41] Bill Cheswick, "Internet Mapping Project", *http://www.cheswick.com/ches/map/index.html*, 2000.