2-2-2007

Feasibility of DDos Attacks with P2P Systems and Prevention Through Robust Membership Management

Xin Sun
Purdue University, sun19@purdue.edu

Ruben Torres
Purdue University, rtorresg@purdue.edu

Sanjay Rao
Purdue University, sanjay@purdue.edu

Feasibility of DDoS Attacks with P2P Systems and Prevention through Robust Membership Management

Xin Sun, Ruben Torres and Sanjay Rao
School of Electrical and Computer Engineering, Purdue University
West Lafayette, IN47907
{sun19, rtorresg, sanjay}@ecn.purdue.edu

ABSTRACT

We show that malicious nodes in a peer-to-peer system may impact the external Internet environment, by causing large-scale distributed denial of service attacks on nodes not even part of the overlay system. This is in contrast to attacks that disrupt the normal functioning, and performance of the overlay system itself. We formulate several principles critical to the design of membership management protocols robust to such attacks. We show that (i) pull-based mechanisms are preferable to push-based mechanisms; (ii) it is critical to validate membership information received by a node, and even simple probe-based techniques can be quite effective; (iii) validating information by requiring corrobaration from multiple sources can provide good security properties with insignificant performance penalties; and (iv) it is important to bound the number of distinct logical identifier (e.g. IDs in a DHT) corresponding to the same physical identifier (e.g., IP address), which a participating node is unable to validate. We demonstrate the importance of these principles in the context of the Kad system for file distribution, and ESM system for video broadcasting. To our knowledge, this is the first systematic study of issues in the design of membership management algorithms in peer-to-peer systems so they may be robust to attacks exploiting them for DDoS attacks on external nodes.

1. INTRODUCTION

Peer-to-peer systems are rapidly maturing from being narrowly associated with copyright violations, to a technology that offers tremendous potential to deploy new services over the Internet. The recently released Windows Vista is equipped with its own, under-the-hood P2P networking system [2], and several commercial efforts are exploring the use of peer-to-peer systems for live media streaming and video distribution [3, 6, 13, 23, 32, 35]. Recent studies [9] indicate that over 60% of network traffic is dominated by peer-to-peer systems, and the emergence of these systems has drastically affected traffic usage and capacity engineering.

With the proliferation of peer-to-peer systems, it becomes critical to consider how they can be deployed in a safe, secure and robust manner, and understand their impact on an Internet environment already suffering from several security problems. Peer-to-peer systems enable rapid deployment by moving functionality to end-systems. However, they are vulnerable to insider attacks coming from (potentially colluding) attackers that infiltrate the overlay or compromise member nodes.

Several recent works [8, 11, 27, 28, 33] have studied how malicious nodes in a peer-to-peer system may disrupt the normal functioning, and performance of the system itself. In this paper, however, we focus on attacks where malicious nodes in a peer-to-peer system may impact the **external** Internet environment, by causing large-scale distributed denial of service (DDoS) attacks on nodes not even part of the overlay system. In particular, an attacker could subvert membership management mechanisms, and force a large fraction of nodes in the system to believe in the existence of, and communicate with a potentially arbitrary node in the Internet. Such attacks may be hard to detect as the packets arriving at a victim are not distinguishable from normal protocol packets. These attacks may be viewed as a particular kind of reflector attacks [22], however the scale and unique properties of peer-to-peer systems make them worthy of study in their own right.

We show that a potential attacker can launch attacks of hundreds of megabits a second on an external node, by exploiting popularly deployed file distribution systems such as eMule [12], and the extensively deployed video broadcast system ESM [10]. While recent work [5, 21] has presented attacks exploiting the Gnutella and Overnet systems, we show that the problem is far more critical than these works demonstrate, and is general to a wider range of systems. We believe these attacks are merely the tip of the iceberg, and unless the challenges are systematically addressed, there is potential for catastrophic consequences on the Internet akin to the congestion collapse in the mid 80's or the large-scale worm outbreaks in the last decade.

We present the first systematic study of issues in the design of membership management algorithms in peer-to-peer systems so they may be robust to such exploits.

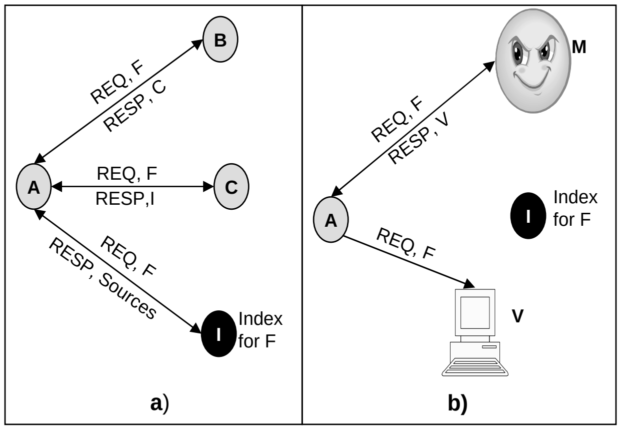

Figure 1: a) Kad search mechanism. b) Redirection attack

We show that pull-based approaches, where information conveyed by a member is always in response to a prior solicitation, are preferable to push-based approaches where a member may disseminate membership information to other members in an unsolicited fashion. We evaluate techniques for validating group membership information communicated from one node to another. These techniques include direct validation through probes, and techniques where information can be used only if it is validated from multiple nodes. While the latter technique is inspired by prior analytical work on Byzantine-tolerant gossip algorithms [16–18, 20], we study the costs in convergence/performance, and benefits in terms of enhanced security properties in the context of an actual system. Finally, we show that it is important to bound the number of distinct logical identifier (e.g. IDs in a DHT) corresponding to the same physical identifier (e.g., IP address), which a participating node is unable to validate.

Based on the principles above, we implement several refinements to the membership management algorithms in Kadand ESM. We conduct detailed evaluation studies of the systems with the various refinements using controlled experiments on Planetlab. Overall, our results demonstrate their effectiveness in minimizing the vulnerability of the systems to DDoS attacks without significant degradation in application performance.

The rest of the paper is organized as follows. Section 2 describes the rationale for choosing the Kadand ESM systems, and the vulnerabilities. Section 3 presents results demonstrating the feasibility of DDoS attacks on these systems. Section 4 describes issues important in the design of resilient membership management algorithms. Section 5 describes our methodology to evaluate the issues, and Section 6 presents results.

2. VULNERABILITIES IN P2P SYSTEMS

In this paper, we focus on DDoS attacks triggered by exploiting the membership management algorithms of peer-to-peer systems. The membership management algorithms in a peer-to-peer system enable a node to join the group, and maintain information about other members, even though nodes may join or leave the system. To scale to large group sizes, typical nodes maintain knowledge of only a small subset of group members. While a large number of peer-to-peer systems have emerged in recent years, two of the most common approaches for membership management involve the use of distributed hash tables (DHTs) [19, 24, 25, 29, 37], and gossip-based algorithms. While popular file-distribution systems like BitTorrent [7] and eMule [12] originally relied on centralized servers (trackers) for group management, more recent versions use decentralized mechanisms based on DHTs. Many other systems such as ESM and CoolStreaming [10,36] employ gossip-like mechanisms to maintain group membership information.

To demonstrate the generality of the issues discussed, we consider peer-to-peer systems targeted at applications with very contrasting propertie very different membership management designs. In particular, we consider file distribution and video broadcasting applications. Video broadcast applications are distinguished by their stringent real-time constraints requiring timely and continuously streaming delivery. Both applications are bandwidth-intensive, and large scale, corresponding to tens of thousands of users simultaneously participating in the application.

The particular systems we consider in this work include Kad [12] for file distribution and ESM [10] for video broadcasting. Kad is a DHT based on Kademlia [19], which is supported by the popular eMule [12] clients and other eMule-like clients such as aMule [4] and xMule [34]. To the extend of our knowledge, Kad is the largest DHT currently used, with more than one million concurrent peers [30]. ESM is a video broadcasting system that employs gossip-based membership algorithms. It is one of the first operationally deployed systems and has seen significant real-world deployment [10]. We are motivated to use these systems given their extensive deployment, the contrasting applications they represent, and the different yet representative membership algorithms they employ.

2.1 DHT-Based File Distribution:Kad

In Kad, users and files have IDs that are globally unique and randomly chosen from the same ID space of **128** bits. Each node mantains a routing table with a subset of peers that are part of the system. For any given file, there are "index-nodes" which maintain a list of members who own that file. Index nodes are not dedicated nodes but regular participants, who have an ID close to a file ID. For example, in Figure 1.a), node

A wishes to download a file F. A must first discover the index-node I, and obtain from it a list of members having the file. To discover I, A will query members that it has in its own table, which are either index nodes or can point A to nodes closer in the ID space to the file ID, which are likely to be index nodes. In our example, A will initially query B which is not an index node for file F. B responds with C whose ID is closer to the ID of F. Next, A will query C who will respond with I. This process can repeat several times, but given the properties of distributed hash tables, convergence of the search process is likely. In our case, I is the index node for file F and will respond to A with a set of *sources* that own a partial or complete copy of the file. Finally, A will contact the sources to begin the download process.

Kad performs a similar lookup process for keyword search, file and keyword publishing, and routing table maintenance. Other important mechanisms such as bootstrapping are also implemented in Kad but are not relevant to this paper.

Vulnerability: Kad may be exploited to cause a DDoS attack on a victim that is not part of the Kad network by creating a redirection attack. Whenever the attacker receives a lookup query from a peer, it will return a response containing the victim's IP address and port. For example, Figure 1.b) shows how malicious user M can make user A send a query to V, which is an Internet user, not part of the Kad network. The attack at the victim can be magnified if many valid users contact the attacker when looking for index nodes. In addition, a coalition of attackers could further increase the magnitude of the traffic at the victim.

2.2 Gossip-based Video Broadcast: ESM

ESM is a video broadcasting system built on top of an overlay network. It constructs a multicast tree for data delivery and employs a gossip-based mechanism to propagate the existance of members on the group. This information is later used by the nodes in the system to change parents when necesary in the multicast tree. To be more specific, for the gossip mechanism each member A, periodically (every 1.5 seconds) picks another member B at random, and sends it a subset of group members that it knows. B adds to its list any members that it did not already know, and may send messages to the new nodes as part of normal protocol operations.

Vulnerability: The gossip mechanism to propagate membership information, may be exploited by having malicious users trick valid users into sending protocol related messages to a victim that is not part of ESM. A malicious user M, could generate a gossip message that contains false information about the victim as being part of the group. Valid users will include the victim in their list of known peers. At a later time, the victim could receive a high rate of unsolicited traffic from valid users of the system. The attack can be magnified if malicious users gossip fake messages at a higher rate.

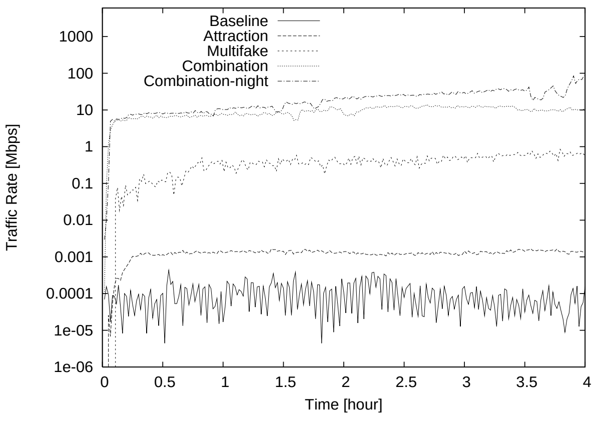

Figure 2: Sensitivity to the heuristic employed by the attacker to increase the attack

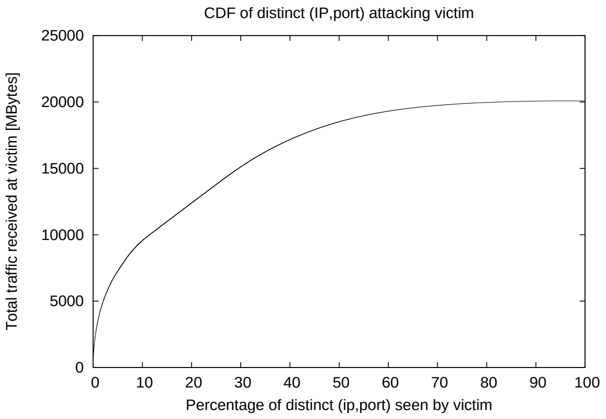

Figure 3: CDF of the distinct (IP,port) pairs generating raffic at the victim

3. ACHIEVING HIGH MAGNITUDE DDOS ATTACKS

In this section, we show the feasibility to exploit vulnerabilities in membership management algorithms of peer-to-peer systems, to create DDoS attacks on nodes not part of the system. We discuss various heuristics used to increase the attack magnitude at the victim. These heuristics provide insights that can help to formulate key principles that must guide the design of robust membership management protocols.

3.1 Attack using Kad

Our approach is to create a redirection attack where every lookup message the attacker receives, will be returned containing the victim's IP address and port. Then, the valid user will query the victim. But query

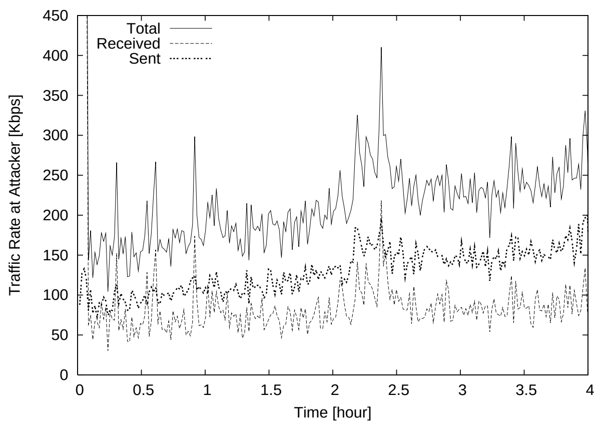

Figure 4: Traffic seen at an attacker using all heuristics to generate attack

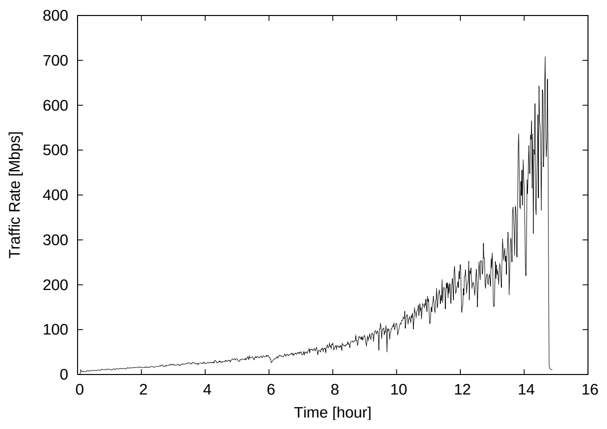

Figure 5: Total traffic seen at victim, with 200 attackers, over a period of 15 hours.

messages are usually small and making a few valid users contact the victim will not cause a high impact at the victim. Therefore, we combine a set of heuristic that will magnify the attack at the victim.

• *Baseline:* As described in Figure 1.a), node A may seek to locate the node nearest to a given target ID F. As part of the operations, it may send a message to a (malicious) node M that A already knows, who in turn responds with the IP address and port of the victim V (indicating that V is part of the group) along with a fake id for V which is closer to F. A then sends query messages to V, as part of its normal operation.

• *Attraction:* The magnitude of the attack above is dependent on the frequency with which other nodes may contact the malicious node. In general, this is small given that the group may involve millions of members, but there are only a few attackers. We found a vulnerability in Kad where a malicious node may proactively push information about itself to a large number of nodes in the system, forcing them to add the node to their routing tables. A key parameter for the attraction heuristic is the rate at which the attacker spreads itself in the tables of other nodes. We discuss the implications of the parameter later in the section.

• *Multifake:* While the attacks above cause several clients to contact the attacker and be redirected to the victim, better amplification can be achieved if the attacker includes the victim's information several times in response to a query. The key insight behind the attack is the distinction between the *physical identifier* of a participating node such as its IP address, and its *logical identifier,* the node-id in the DHT space. Kad, and indeed many peer-to-peer systems, are designed to allow a participating node to communicate with multiple logical identifiers even though they share the same physical identifier (IP address). This has several advantages, for instance, enabling distinct users behind the same Network Address Translator (NAT) to participate in the system, even though they share the same physical IP address (see Section 4.3 for further discussions). The *Multifake* heuristic exploits this to achieve large amplifications by having the attacker redirect innocent clients to multiple logical identifiers, all sharing the IP address of the victim. Further, it seeks to achieve even greater magnification by having the attacker include itself in the query responses a small number of times, so that the valid user could be repeatedly attracted to the attacker and redirected to the victim.

Results: We instrumented an aMule client to implement attackers with the heuristics above and make them join the real live Kad network. The victim node is in our laboratory. Each experiment runs for several hours, and we report on magnitudes of attack seen. The experiments employ 5 attackers unless otherwise mentioned.

Figure 2 shows the traffic at the victim with the three heuristics. The X-Axis is the time since the start of the experiment. The Y-Axis is the amount of traffic seen in Mbps. From bottom to top, the first three curves correspond to the attack magnitude with the given heuristic alone. The last two curves correspond to the combination of the previous heuristics at different times of the day. High magnitudes of over 10Mbps seen at the victim when all heuristics are turned on. It is also interesting to see that the entire set of heuristics is required to generate the high attack magnitudes, and any subset is insufficient. We also observed that the magnitude of the attack traffic is sensitive to the time of day. As shown in the last two curves *Combination* was obtained during the day and *Combination-night* was obtained late at night. In our experiments, the attacks could go as high as 100 Mbps in some runs during the night.

Figure 3 shows the distribution of distinct (IP address,port) pairs of innocent clients that are being redirected to the victim when all heuristics are turned on. A point (X,Y) in this graph means that traffic Y is contributed by X percent of distinct IP and port. Over 200,000 distinct (IP,port) pairs were redirected in the attack. As shown, the distribution is not sharply skewed indicating the traffic is not coming from any single client alone which can make the attack difficult to contain.

Figure 4 shows the traffic observed at one attacker, both in terms of traffic sent and received. The Y-Axis is traffic rate in Kbps. The X-Axis is time in hours. The main observation from this graph is that the traffic seen at the attacker is only about 250Kbps, which while higher than what a normal user sees, is 40 times lower than the traffic seen by the victim. Even if the total traffic at all attackers is consider, there is still a magnification factor of 8. A point to note is the spike at the start of the experiment. This is due to the attraction heuristic where a malicious node attempts to insert itself in the routing tables of several other nodes. We discuss the implications in the next paragraph.

We considered whether even higher attack magnitudes are possible by combining larger number of nodes and increasing the rate of the attraction heuristic. Figure 5 shows an attack generated using 200 malicious nodes scattered around Planetlab, with the victim in our laboratory. Traffic of over 700 Mbps was received by the victim after 14 hours of experiment. This far exceeded what we feared, and indicates the criticality and seriousness of the problem. We abandoned further experiments on this line given the seriousness of the attacks. An ISP of one of the attacker nodes was concerned whether the node was running a random port-scan attack. This was because each attacker probed around 100,000 Kad nodes as part of the attraction heuristic, and not all nodes responsed since they were no longer in the system. While this offers hope that such attacks could be detected, it may be feasible to evade detection by reducing the rate at which malicious nodes spread information about themselves to others. Significant attack magnitudes may still be achieved, though it may take longer for the attacks to ramp up to these values. We have conducted (carefully controlled) experiments to confirm this observation.

3.2 Attack using ESM

We exploit the vulnerability described in Section 2 where a malicious node M sends gossip messages to an innocent client C, falsely indicating that the victim V is part of the group. Similiar to *Multifake* in Kad, we augmented the heuristic to achieve greater attack magnitudes by including the IP address of the victim several times, each time with a different logical identifier. ESM also makes use of logical identifiers (called uid in [10]) distinct from IP address and port information, primarily to handle issues with NATs. Like Kad, ESM allows a participating node to communicate with multiple logical identifiers even though they share the same physical identifier (IP address). Again this is motivated by NATs.

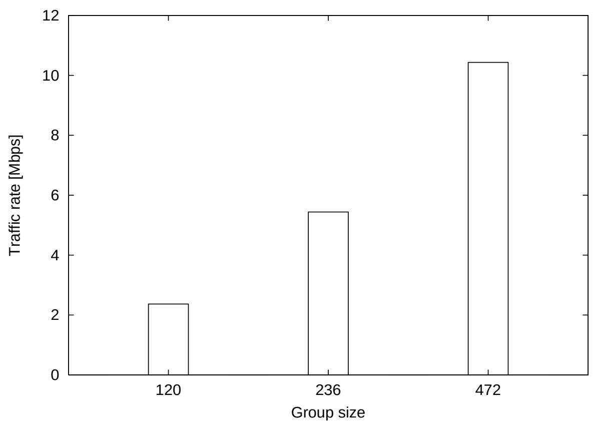

Figure 6: Sensitivity to number of clients. Percentage of malicious clients fixed to 10%

Results: We conducted experiments in Planetlab using the attacker described above. Figure 6 shows the magnitude of the DDoS attack in comparison to the total number of ESM clients. We fixed the percentage of malicious clients to be 10% and varied the total number of clients. The traffic seen by a victim is several Megabits a second, a factor of 1000 more than control traffic seen by a normal ESM member (about 3 Kbps). Further, the attack traffic increases approximately linearly as the number of participants increase. In a real scenario, involving tens of thousands of participating nodes, the attack magnitude could be orders of magnitude higher. The experiments above assume that 10% of the hosts are malicious. Figure 7 plots the attack traffic fixing the number of clients at 472, and varying the percentage of malicious clients. Even a very small fraction of malicious clients can cause a serious attack at the victim, with 1% of nodes being malicious resulting in attacks of 4Mbps at the victim.

4. RESILIENT MEMBERSHIP MANAGEMENT

While several "point-solutions" may be feasible to limit the specific attacks we presented in Section 3, we believe these attacks are symptomatic of more fundamental issues that must be carefully addressed in designing robust membership management protocols. We elevate and highlight some of the principles involved. These principles then guide the design of various refinements to the ESM and Kad systems.

4.1 Pull vs. Push

A fundamental factor that impacts the vulnerability of membership management algorithms is whether they are push-based, or pull-based. In a push-based design, members may disseminate membership information to

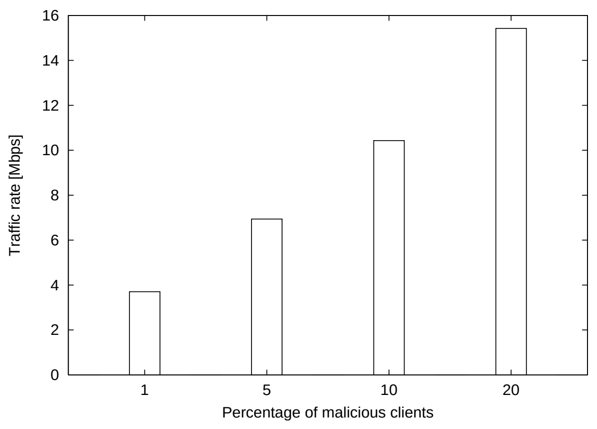

Figure 7: Sensitivity to percentage of malicious clients. Total number of clients fixed to 472

other members in an unsolicited fashion. In contrast, in a pull-based design, any information conveyed by a member is always in response to a prior solicitation. Systems like ESM and CoolStreaming [10,36] use push-based gossip algorithms, while BitTorrent and eMule use pull-based techniques.

We argue that pull-based protocols are preferable from the perspective of robust design since an attacker does not have control over the rate at which it can propogate malicious information. In contrast, push-based protocols are more vulnerable to compromise, since an attacker can control the rate at which it can contact other victim nodes. That said, a few points are in order. First, attacks where a node pushes malicious information at higher rates than normal in push-based approaches are potentially detectable since the amount of traffic that must be generated to spread false information is high. However, solutions for detection may not be straight-forward. They may either require all service providers of all nodes taking part in the P2P system to detect abnormal variations in traffic, or they may require correlating observations across multiple nodes in the system, neither of which are trivial. Second, pull-based algorithms may themselves not suffice. In particular, a factor dictating the vulnerability of pull-based algorithms is the ability of an attacker to attract queries from innocent nodes towards itself. This may in turn depend on the number of nodes that know an attacker, as well as the skew in distribution of requests to any node - for example, arising due to variations in popularity of files owned by various nodes. In fact, our attacks on the Kad system leveraged a vulnerability which enabled an attacker to populate routing tables of a large number of innocent clients, thereby attracting queries towards itself. Finally, a subtle implementation issue with pull-based algorithms is that a member must be able to verify that any reply is actually in response to a prior request. This could be handled through a variety of well-known mechanisms. In fact, both Bittorrent and Kad handle this by storing transaction identifiers of outgoing pull requests and requiring that responses have identifiers matching outgoing requests.

4.2 Validating Information

When A receives information about a member C from member B, clearly it would be desirable to have a means of validating the information. However, validation could incur costs as well. We discuss a list of techniques we consider for validating information, and their pros and cons:

• *No-Validation:* This is a strawman solution where when information is learnt about a new member C, that information is accepted without any further validation. Evidently, this solution is the most vulnerable to being exploited in a DDoS attack, however no costs are incurred in terms of performance or overheads. The ESM system does not have mechanisms for validation.

• *Direct-Validation:* In this solution, when a node learns information about C, it directly contacts C to ensure C is alive and can respond to the message. C is accepted as a valid group member only if C responds to the message. The Kad system employs a direct validation scheme. A potential concern is that this kind of validation itself can become subject to exploit if a large number of innocent nodes try to conduct the validation, as our results with Kad have shown. Further, under lossy conditions, loss of validation packets (or responses) may mistakenly lead a node to conclude a remote member is not actually part of the group. While this could be potentially offset by transmitting a larger number of validation packets, this could increase overhead, and magnitude of attack traffic as well.

• *Multi-Node-Validation:* In this solution, a node does not directly contact C to validate it, but waits until it learns about C from at least m nodes, where m is a parameter. This approach has been used in several prior works on Byzantine-tolerant diffusion algorithms [16–18, 20]. These works assume the maximum number of faulty nodes is known apriori, and require corroboration from at least 1 more than the number of faulty nodes. We adopt a more probabilistic variant, where the parameter m determines the number of sources from which corroboration is required. In our heuristic, a node may still receive false information (for example, when the number of attackers exceeds m). However, further confirmation could be obtained using Direct-Validation. The scheme helps by minimizing the number of direct validation requests sent to the victim. The disadvantage however is that waiting for responses from several nodes may take a long while, and can slow down convergence and performance. Clearly, the parameter m is critical - a higher value offers potentially stronger security properties, but slower convergence.

One additional consideration in schemes that employ validation is whether A must spread information about a member C when it is still in the process of validating whether C is really in the group. Doing this has the advantage that good information can propagate fast in the absence of attackers. However, it has the disadvantage that malicious information could also propagate fast, with minimal effort from the attacker, and making the attacker harder to detect. This trade-off is interesting only if the "length" of validation interval is long. Direct validation schemes that involve a single probe-response, have a validation period which is short, and hence the trade-offs involved in whether information yet to be validated is spread or not is likely insignificant. The trade-off is however interesting in schemes involving longer validation periods, for instance a repeated series of pings, or with multi-node validation. We defer an investigation of these trade-offs to future work.

4.3 Bounding Logical Ids for a Physical Id

Many peer-to-peer systems use logical identifiers to identify participating nodes, rather than physical identifiers such as its IP address, as mentioned in Section 3. Participating nodes with different logical identifiers may share the same physical identifier. For example, hosts behind the same NAT, likely have the same IP, though their logical identifiers differ. Other examples include two users working on a time-shared machine, or multiple reincarnations of a host (a user that leaves the group, and rejoins with a different logical identifier).

The attacks in Section 3 on both the Kad (*Multifake* heuristic) and the ESM systems exploit the fact that a malicious node could repeatedly redirect an innocent client to a victim IP address, but using different logical identifiers for the victim IP each time. This aspect was instrumental in magnifying the traffic at the victim.

An intuitive heuristic to mitigate such attacks is to bound the total number of distinct logical identifiers corresponding to the same physical identifier, which a participating node is unable to validate. We refer to this as the *Bound-LogIDforPhyID* heuristic. The key issue in doing this is designing bounds that are small enough to help limit DDoS attacks, yet are realistic enough to take into account the prevalence in real deployments of instances where nodes with different logical identifiers share the same physical identifier due to factors such as NATs discussed above.

While the *Bound-LogIDforPhyID* heuristic has potential to limit DDoS attacks on victims not in the group, it has a vulnerability that could be exploited by an attacker. In particular, an attacker could disconnect a client (victim) who is really part of the group, by flooding other clients with several logical identifiers for the victim, and the physical identifier (IP address) of the victim. This causes other clients to exceed their bound for the victim, and not communicate with the victim. To address this, we consider the *Bound-LogIDforPhyID-Src* heuristic which bounds the total number of distinct logical identifiers corresponding to the same physical identifier that have been *learned from the same member* and which a participating node is unable to validate. While this requires more state information to be maintained, it can prevent disconnection attacks above. Further, it can still limit DDoS attacks, however the magnitude that can be achieved depends on the bounds used for the above, and the number of attackers.

Finally, we have considered whether to simply bound the total number of logical identifiers (irrespective of physical identifier) learnt from a given member which a participating node is unable to validate. This heuristic is simpler than the *Bound-LogIDforPhyID-Src* described above and requires less state, however, the bounds involved need to be larger, which in turn implies higher DDoS magnitudes at a victim.

5. EVALUATION METHODOLOGY

Our evaluations explore the importance of the principles discussed in Section 4, study security and performance trade-offs involved between design heuristics, and parameterize heuristics. In this section, we present our evaluation goals, metrics, refinements implemented in the Kad and ESM system and our experimental methodology.

5.1 Evaluation Goals

Our evaluations are motivated by several goals:

• How effective are pull-based mechanisms in lowering attack magnitudes as compared to push-based mechanisms?

• How effective is *Direct-Validation* compared to *No-Validation* in limiting the magnitude of DDoS attacks? Are the additional overheads acceptable? How well does *Direct-Validation* perform under lossy conditions, given that loss of validation request/response packets may lead to invalidation of a genuine node part of the group?

• How effective is *Multi-Node-Validation* in preventing DDoS attacks? How critical are the performance concerns introduced? What factors do the parameter m depend on? We recall that m is the number of distinct sources that must verify membership information before it is used.

• How prevalent is the use of multiple logical identifiers for the same physical identifier in actual deployments? How do they influence choice of realistic bounds for the *Bound-LogIDforPhyID* and *Bound-LogIDforPhyID-Src* heuristics? How serious are disconnection attacks using the *Bound-LogIDforPhyID* ? How effective is *Bound-LogIDforPhyID-Src* in mitigating DDoS attacks?

5.2 Performance Metrics

We characterize the performance and security tradeoffs in our heuristics as follows:

• *Application performance:* For file distribution (Kad), we consider the time taken for a search for a given target id to be successful. Since some searches may not be successful at all, the fraction of searches that are successful is also considered. For video broadcast (ESM), we consider the fraction of the streaming video rate received by participating nodes.

• *Attack Impact:* We evaluate the impact of an attack on a victim by considering the rate of traffic received at the victim.

5.3 Methodology

To evaluate the heuristics we introduce, we have implemented several refinements to the ESM and Kad systems, and conducted evaluations over Planetlab. The base ESM system does not employ validation, and uses push-based mechanisms. Each member picks a random member it knows, and sends it a random subset of members that it knows. We implemented a pull-based version, where a member picks a random member at the same frequency, and requests it to send it a subset of group members it knows. We also implemented a scheme for direct validation where a member A probes any member B that it learns about and accepts B only if it receives a response from B. The base Kad system does adopt validation, and uses pull-based mechanisms. We modified it to implement the Multi-Node-Validation scheme, and the Bound-LogIDforPhyID, and Bound-LogIDforPhyID-Src heuristics.

ESM experiments: Our experiments with ESM leveraged traces from real broadcast events [10] to model the group dynamics pattterns, and bandwidth-resource constraints of nodes. We emulate twenty minute segments of the trace. The clients already present in the trace at the start of the segment join in a burst over the first two minutes, then follow join/leave patterns exactly as in the trace for the next twenty minutes. For the trace segment used, 363 nodes participate, with a group peak size of 108 nodes. The streaming video rate employed was 475Kbps, which represents typical media streaming rates in real settings like [10]. We used two versions of the attacker. In the first version, the attacker behaved similar to a regular node, except that whenever it needed to push membership information (or was contacted by another member pulling membership information), fake information regarding the victim was sent. We term such an attacker *undetectable-attacker*, since the traffic patterns it introduces are not distinguishable from a normal node in the system. We also considered a second attacker that we term *aggressive-attacker,* which periodically pushes fake membership information

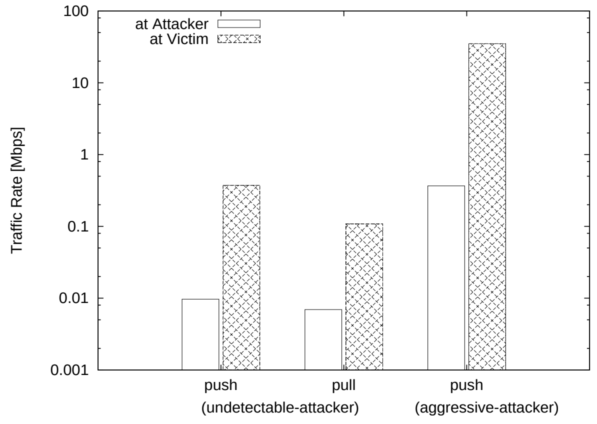

Figure 8: Traffic seen by victim and attacker, for undetectable and aggressive attackers

about the victim to a large number of nodes in the system. This kind of attacker only impacts a push-based solution. We note that an aggressive-attacker may be sending more traffic than regular nodes in the system.

Kad experiments: Our Kad experiments used a synthetic trace of join/leave patterns where the inter-arrival patterns of nodes, and the stay time duration of nodes followed a Weibull distribution. This was based on recommendations by a recent measurement study [31]. The trace has 1455 instances in total, with peak group size around 300. Nodes executed a search pattern where each node periodically (30 seconds) conducted a search for a random logical identifier, which was intended to simulate a search request for a file. We assumed that the search successfully reached an index-node if it reached any of the k members closest to the logical identifier for which the search was conducted. We used k values of 5, but conducted sensitivity of our results to this parameter. To model attackers in Kad, we used the heuristics in Section 3.1. To study disconnection attacks alone, we slightly modified the attacker to redirect the client to a victim IP actually in the group rather than outside, still returning several fake logical identifiers for that IP. The remaining heuristics were similar to those described in Section 3.1.

Kad Measurement Study: To understand the prevalence of multiple logical identifiers for the same physical identifier in actual deployments, we have conducted a measurement study of the real Kad infrastructure. We implemented a Kad crawler which constantly performed searches for random logical identifiers at a very high rate and extracted (Logical identifier, IP, Port) triples from the responses it received. We ran 6 such crawlers from 6 machines for a time period of 20 hours, and were able to collect about 4 million distinct(Logical identifiers , IP, Port) triples.

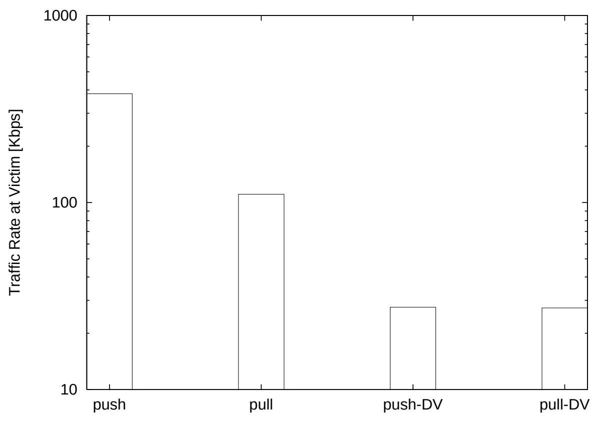

Figure 9: Traffic seen by victim for pull and Direct-Validation, where 10% of the members of the group are attackers

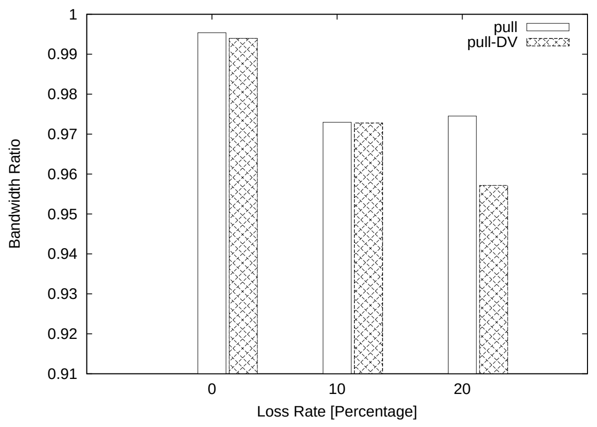

Figure 10: Bandwidth Ratio seen by members of the multicast group, for different loss rate percentages.

6. RESULTS

Section 6.1 discusses potential benefits of pull-based over push-based approaches. Section 6.2 discusses benefits and issues with Direct-Validation. ESM uses push, and does not employ validation. Kad uses pull, and employs validation. Therefore, our experiments in the first two sections employ ESM. Section 6.3 focus on benefits of a more sophisticated validation scheme, Multi-Node-Validation. Section 6.4 shows issues with logical and physical identifiers. The last two sections are relevant to both Kad and ESM, and we focus our discussions on Kad.

6.1 Pull vs. Push Approaches

We present evaluation results of the base ESM system with push-based mechanisms, as well as our refinements that employ pull-based mechanisms. Our evaluations

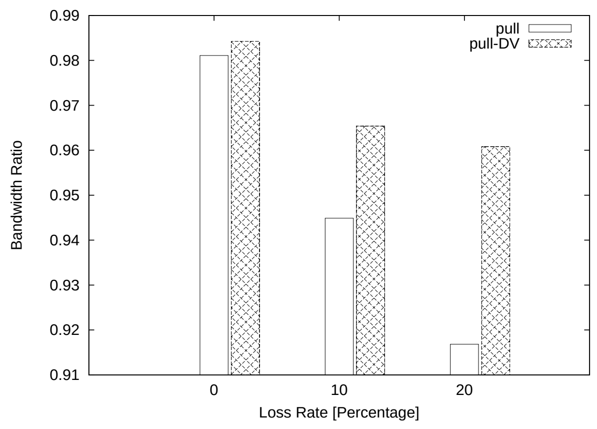

Figure 11: Bandwidth Ratio seen by members of multicast group, where 10% of the members of the group are attackers, for different loss rate percentages

are conducted using both the undetectable-attacker and aggressive-attacker described in Section 5.3. Figure 8 plots the average traffic seen at the victim and an attacker during the run, for both the undetectable-attacker, and the aggressive-attacker. Further, for the undetectable-attacker, traffic is shown for both pull and push cases and for the aggressive-attacker, traffic is shown only for the push case since pull is not relevant here. We note that the traffic at the attacker is extremely small for the undetectable-attacker but about 300 Kbps for the aggressive-attacker. This is consistent with the amount of work the attacker does in each case. We also note that the traffic at the attacker, for all cases, is at least one order of magnitude less than the traffic at the victim, which demostrate the potency of the attack.

From the perspective of the victim, there are two points to note from the graph. First, the magnitude of attack traffic can be very high with push-based approaches when an aggressive attacker is employed, which may be prevented by pull-based approaches. Second, even with an undetectable-attacker, the magnitude of attack traffic with pull-based mechanisms is slightly less than push-based mechanisms. This is because with push-based techniques, an attacker may always infect a client in any iteration - in pull-based techniques, this is dependent on the probability with which a normal client contacts an attacker machine.

6.2 Direct validation

While the results above show the importance of pull-based mechanisms, we next consider the need for a node to directly validate any information it receives by probing any new peer it learns about.

Figure 9 shows the average attack traffic generated at the victim in the presence of malicious nodes in experiments over 20 minute periods. Four schemes are considered. The first scheme is push without any validation (push) which refers to the original ESM system. The remaining schemes are pull without validation (pull), push with Direct-Validation (push-DV), and pull with Direct-Validation (pull-DV), depending on whether they use pull alone, validation alone, or both. We see that the use of Direct-Validation greatly reduces the magnitude of traffic seen at the victim, irrespective of whether pull or push based mechanisms are used.

While the results above indicate the benefits of Direct-Validation, this mechanism by itself is insufficient, and must be used in addition to pull. This is because an aggressive-attacker could push information at high-rates and still make normal users query the victim a significant number of times. We performed experiments with this approach and noted that while push-DV does perform better than push, the magnitude of the attack was still high (almost 1 Mbps) - avoiding this requires the use of pull-DV.

Loss Experiments: One concern with direct validation is the performance under lossy conditions. Good nodes may be mistakenly assumed as invalid when a validation request is lost. We conducted experiments to evaluate this concern by measuring the bandwidth ratio seen by nodes under lossy conditions. We emulated a loss rate on Planetlab, by dropping a random fraction L of packets when they arrived at the receiver. We varied L, so 0%, 10% and 20% of the packets were dropped. We note these losses are in addition to natural losses that occur on the Planetlab (which we observed was small).

Figure 10 shows the bandwidth ratio, or the fraction of the source rate received by users participating in the broadcast, for the pull and pull-DV schemes under lossy conditions, when no malicious nodes are part of the system. There are 3 sets of bars, corresponding to the three different loss rates, and each set has 2 bars corresponding to the two schemes. While the performance of pull and pull-DV are very similar under no loss and 10% loss, the performace of pull-DV does show more degradation under 20% loss. This is because loss of validation requests/responses can force a node to not consider certain members as candidate parent choices, thereby not making the best use of available choices in the tree.

While the results above were obtained under non-malicious workloads, Figure 11 shows the bandwidth ratio of the pull and pull-DV scheme under lossy conditions, with 10% of the nodes being malicious. While both schemes degrade under loss, the degradation for pull is more pronounced than pull-DV. With malicious nodes and the pull scheme, nodes may be filled with fake membership information, thereby greatly reducing

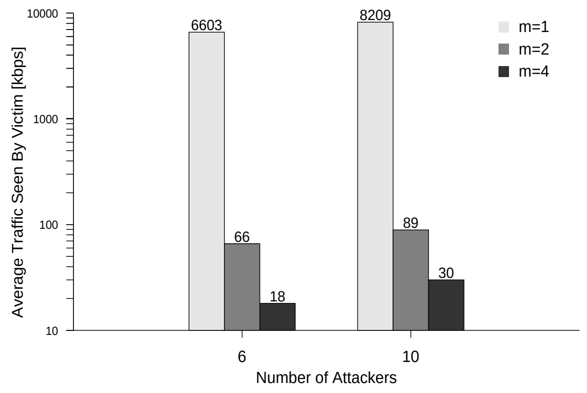

Figure 12: Traffic seen by victim as a funcion of the number of attackers. Traffic is averaged accross the whole run.

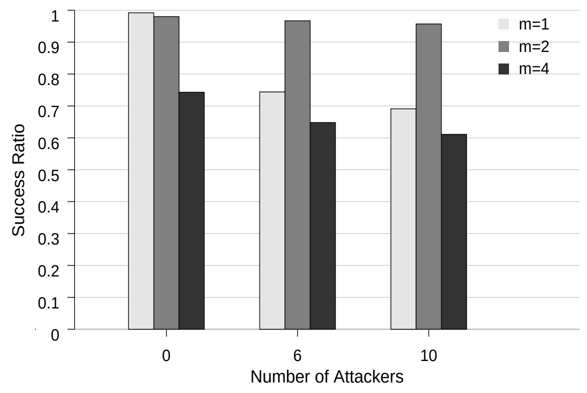

Figure 13: Average success ratio of searches, as a function of the number of attackers.

the performance of the system. In contrast, with pull-DV, while some genuine parent choices may be missed due to loss of packets in the validation process, there is no false membership information at nodes. Thus, the benefits of avoiding consideration of false members outweighs the costs of missing a few genuine members.

To summarize, with non-malicious workloads, and significantly lossy conditions, pull-DV can have modest performance degradation compared to pull. However, when malicious workloads are considered, pull-DV outperforms pull considerably even under loss. This indicates the promise of using the pull-DV heuristic.

6.3 Multiple member validation

Our results so far demonstrated the importance of pull-based approaches and direct validation. We next consider the trade-offs involved in using the more sophisticated Multi-Node-Validation scheme. We believe these results are applicable to both ESM and Kad, but

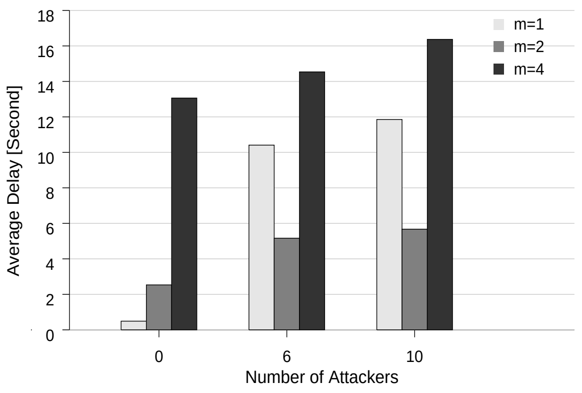

Figure 14: Average delay of searches, as a function of the number of attackers.

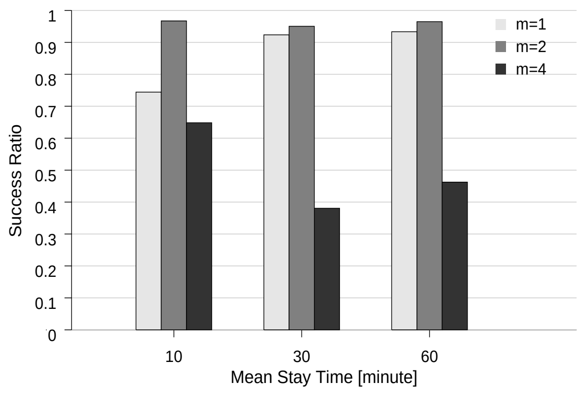

Figure 15: Sensitivity to dynamics: node success ratio

we focus our investigation on Kad.

Figure 12 shows the traffic generated at the victim as a function of the number of attackers. There are two sets of bars, each set corresponding to a setting with a particular number of attackers. Each set has 3 bars corresponding to the parameter m (the number of nodes required for validation). A value of 1 indicates the default Kad system. Results are not shown for the number of attackers being zero, as clearly there is no traffic at the victim in such a case. Even in the presence of a moderate number of attackers, the overheads at the victim are high for $m = 1$ - about 6 Mbps for 6 attackers. Using $m = 2$ is effective in reducing the overheads at the victim, with attack magnitudes about 89 Kbps even for settings with 10 attackers. Using a higher value of m enables further reduction in overheads, but the benefits are marginal.

While these results indicate the promise of using Multi-Node-Validation in minimizing DDoS attacks, an important concern is performance. Figure 13 shows the success ratio, or the fraction of searches that are successful. Again, each set of bars corresponds to a setting with a particular number of attackers, and there are multiple bars in each set corresponding to different values of m. There are several observations to make. First, in the absence of attackers, a value of $m = 1$ has a success ratio of close to 1, but $m = 2$ performs well too, with a success ratio of 0.98. The success ratio with $m = 4$ is much lower, only around 0.7. Second, when attackers are considered, the results are particularly interesting. While the performance with $m = 1$ degrades significantly, the performance with $m = 2$ is relatively unaffected, and $m = 4$ continues to perform poorly. This indicates that using modest degrees of Multi-Node-Validation not only minimize traffic at the victim, but also sustain good application performance in the presence of attackers; and has modest peerformance degradaton in the absence of them. However using higher degrees of Multi-Node-Validation may have an unacceptably high performance cost.

While the success ratio is one metric of performance, Figure 14 shows the delays incurred by successful searches. For any setting, all searches conducted across all nodes are considered, and the average delays are computed. The lower the bars, the better the performance. The results are consistent with the success ratio metric. In the absence of attackers, $m = 1$ does perform the best with average delays across searches less than 1 second. The performance of $m = 2$ is slightly worse, but still acceptable, with average delays around 2 seconds. With a value of $m = 4$ however, the average delays are high, and over 12 seconds. In the presence of attackers, the performance with $m = 1$ degrades significantly, with searches taking over 10 seconds with 6 attackers. The performance with $m = 2$ shows only modest degradation, with average delays of less than 6 seconds even with 10 attackers. The delays with $m = 4$ continue to remain high in the presence of attackers.

We further considered why the performance with $m = 2$ performs better in the presence of attackers as compared to $m = 1$ and higher m values. The performance of $m = 1$ degrades with attackers because search requests to a given target id may be redirected to non-existent nodes. Using $m = 2$ requires validation from multiple nodes, and offers a better degree of protection in ensuring searches are not misdirected. While $m = 4$ could potentially provide even stronger validation, the downside is that the penalty or "wait-time" involved in confirming the observations from so many sources becomes high, and outweighs the benefits of invalidating malicious information.

Figure 15 studies the sensitivity of our results to the dynamicity of the trace. Each group of bars corresponds to the success ratio with a particular group dynamics

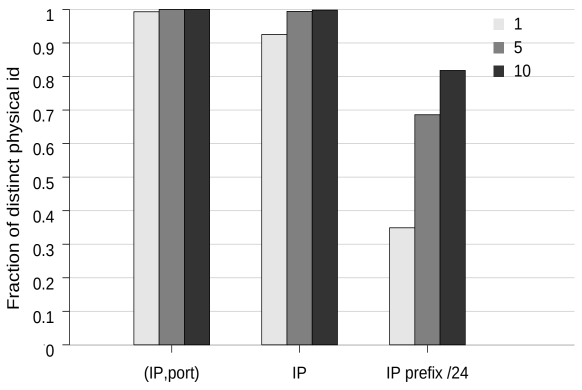

Figure 16: Fraction of distinct physical identifiers associated with a sertain number of logical identifiers, or less

pattern, and each bar corresponds to a different scheme. The three patterns all follow the Weibull distribution with an inter-arrival pattern of 0.4 joins per second - however the mean stay time differs and values of 10, 30 and 60 minutes are used. The number of attackers is fixed at 6. Across all traces, the performance with $m = 2$ is better than the performance with $m = 1$, but the performance with $m = 4$ is significantly worse.

Overall, our results show that modest degrees of Multi-Node-Validation significantly reduces vulnerability to DDoS exploits, has modest performance degradation in the absence of attackers, and offers performance benefits in the presence of attackers. Higher degrees of Multi-Node-Validation however has too significant a performance cost to be viable. While using $m = 2$ was effective in the particularly settings we considered, our results are limited by the size of our test-bed. Our ongoing work seeks to parametrize m using simulations in settings with hundreds of thousands of nodes and hundreds of attackers.

6.4 Bounding Logical Ids for a Physical Id

As discussed in Section 4.3, an important source of attack amplification arises when a malicious node repeatedly redirects an innocent client to a victim IP address (or (IP,port) pair, or even IP prefix), but using different logical identifiers for the same physical identifier each time. The *Bound-LogIDforPhyID,* and *Bound-LogIDforPhyID-Src* heuristics seek to bound the number of distinct logical identifiers corresponding to the same physical identifier, which a participating node is unable to validate. We present our results below parametrizing and evaluating these heuristics.

To guide the design of bounds for the heuristics, we analyzed a trace from a real Kad deployment which contains 4 million clients, and includes the logical identifier, the IP address and the port for each client, obtained as discussed in Section 5. Our objective is to understand how common it is to find nodes with different logical identifiers that share the same physical identifier due to factors such as NATs.

Figure 16 presents results from the analysis of the traces. There are 3 sets of bars, each corresponding to a different notion of a physical identifier - (IP,port) pair, IP address, and IP prefix. For each set, there are 3 bars, corresponding to 1, 5, and 10 logical identifiers. Each bar shows the fraction of distinct physical identifiers that are associated with a certain number of logical identifiers, or less. Almost close to 100% of the (IP,port) pairs involve only 1 logical identifier. There is a tail, and there exists one (IP,port) pair for which over 6000 logical identifiers are used (not shown in the graph). When the IP address alone is considered as the physical identifier, over 92% of the IP addresses are associated with only 1 logical identifier. Over 99.8% of IP addresses are associated with 10 logical identifiers or less, and there exists a tail as before. Barring the tail, the results indicate that reasonably small bounds can be used with the *Bound-LogIDforPhyID,* and *Bound-LogIDforPhyID-Src* heuristics.

So far, we have considered attacks where innocent clients are repeatedly redirected to a victim IP address using different logical identifiers. However, a more sophisticated attacker could target a network by redirecting traffic to hosts with particular IP address prefixes. Figure 16 also shows the fraction of 24-bit IP prefixes that are associated with a certain number of logical identifiers (or less). Only 35% of the prefixes are associated with at most 1 logical identifier, and about 81% of the prefixes are associated with less than 10 distinct logical identifiers. This indicates that larger bounds may be required when limiting the rate of packets sent to a given prefix. We believe the bounds could be lower if we not only consider nodes that share the same prefix, but also consider the number of these which do not respond to validation requests (for example, because they left the group). Obtaining these estimates is subject of our ongoing extensions to our measurement study.

We have conducted experiments with the *Bound-LogIDforPhyID* and *Bound-LogIDforPhyID-Src* heuristics. Our experiments confirm our hypothesis that the *Bound-LogIDforPhyID* heuristic is vulnerable to disconnection attacks. In our experiments, each client applies a bound on the number of logical identifiers for a particular IP address that it cannot validate. A single attacker conducts a disconnection attack as described in Section 5.3. The attacker redirects innocent clients to a victim IP actually in the group rather than outside, still returning several fake logical identifiers for that IP. Figure 17 shows the success ratio of the searches. The success ratio is less than 0.1 for searches conducted for

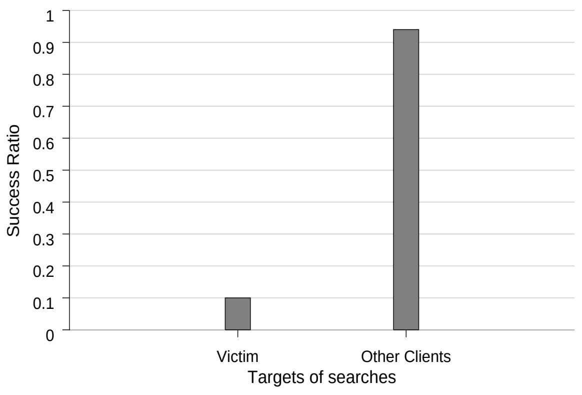

Figure 17: Impact of Disconnection attack to *Bound-LogIDforPhyID* heuristic

the victim id, while it is over 0.9 for searches conducted for other client ids. We have also conducted experiments with the *Bound-LogIDforPhyID-Src.* While we omit results, overall, they confirm our intuition that this heuristic is not only effective in limiting the magnitudes of DDoS attacks, but also is effective in preventing disconnection attacks.

7. RELATED WORK

Researchers have recently shown the potential to exploit the Gnutella and Overnet systems to launch DDoS attacks on the external Internet [5, 21]. Our research seeks to generalize the issues involved, and to propose and design solutions. Ours is the first work to systematically study defenses to the best of our knowledge.

The attack on the Overnet system [21] relies on a attacker impersonating the victim when talking to an innocent client, which forces the client to communicate with the victim. Further, due to an implementation vulnerability in Overnet, spoofing at the IP source address level is not required, but merely at the application payload level. The paper also proposed another type of DDoS attack in which the attacker tries to falsely publish the victim as sources for popular files so other peers will try to download the files by initiating TCP connections to the victim. In contrast, the attacks we present exploit issues with push-based protocols, having attackers attract query traffic, and achieving amplification through multiple logical identifiers being mapped to the same physical identifier. The attack magnitudes we report are also orders of magnitude higher than those reported in [21].

Other works have explored attacks caused by DNS and web-server reflectors, and misuse of web-browsers and bot-nets [1,14,15,22]. We believe the rich and complex design space of peer-to-peer systems makes them worthy of study in their own right. Several works [8, 11, 27, 28, 33] focus on how malicious nodes in a peer-to-peer system may disrupt the normal functioning, and performance of the overlay itself. In contrast, we focus on DDoS attacks on the external Internet environment. Our work benefits from work on the design of byzantine resilient gossip protocols in the traditional distributed systems community [16–18, 20]. While we leverage insights from these works, we believe the scalability, heterogeneity and performance requirements with peer-to-peer networks and applications pose unique challenges and it is necessary to investigate the issues in the context of actual systems. Finally [26] has looked at detecting faults, anomalies and security vulnerabilities in overlays using query processing.

8. SUMMARY AND CONCLUSIONS

We have shown the feasibility of exploiting the membership management algorithms in peer-to-peer systems to create large-scale DDoS attacks on the Internet. Our results shown on two mature and extensively deployed peer-to-peer systems (Kad and ESM), along with recent attacks shown on Gnutella and Overnet [5,21] highlight the generality and criticality of the problem.

We have conducted the first systematic study of the design of membership management algorithms in peer-to-peer systems so they may be robust to such exploits. We summarize our key findings below:
• The use of pull-based protocols is preferable to push-based protocols, given that an attacker has less control over the rate at which it can infect innocent clients.
• Validation of membership information received from another member is important. Even simple mechanisms like Direct-Validation can significantly reduce the magnitude of DDoS attacks compared to No-Validation. While Direct-Validation incurs modest performance degradation under lossy conditions and non-malicious workloads, it outperforms No-Validation under malicious workloads even under loss. Though Direct-Validation is already used in some systems such as Kadtoday, its usage is ad-hoc - our results highlight its criticality, and systematically evaluates its effectiveness and concerns.
• Direct-Validation could be exploited to cause attack traffic to be launched at a victim. This may be avoided by corroborating information from multiple members. Multi-Node-Validation with small values of the parameter m can significantly reduce vulnerability to DDoS exploits, has modest performance degradation in the absence of attackers, and improves performance in the presence of attackers. Larger values of m involve too significant a performance cost to be viable.
• It is important to bound the number of distinct logical identifiers corresponding to the same physical identifier (such as (IP,port) pair, IP address, or IP prefix), which a participating node is unable to validate. A

measurement study of a real Kad deployment indicates low bounds are feasible. We considered two heuristics to achieve this bound - *Bound-LogIDforPhyID,* and *Bound-LogIDforPhyID-Src.* While both heuristics can help to contain DDoS attacks on an external victim, the former heuristic is vulnerable to attacks that disconnect genuine nodes part of the system.

We believe this work has taken a first but important step towards ensuring the safe, secure and robust deployment of peer-to-peer systems. Our ongoing and future work involves considering a wider range of systems, vulnerabilities in other components of peer-to-peer systems, evaluating the performance of heuristics at scale using analysis and simulations, and exploring techniques that can *detect* DDoS attacks exploiting peer-to-peer systems.

9. REFERENCES

[1] DNS amplification attacks. http://www.isotf.org/news/DNS-Amplification-Attacks.pdf.

[2] Windows peer-to-peer networking. http://www.microsoft.com/p2p.

[3] S. Ali, A. Mathur, and H. Zhang. "Measurement of commercial peer-to-peer live video streaming". In *Proc. of Workshop on Recent Advances in P2P Streaming*, 2006.

[4] aMule. http://www.amule.org/.

[5] E. Athanasopoulos, K.G.Anagnostakis, and E.P. Markatos. "Misusing unstructured p2p systems to perform dos attacks: The network that never forgets". In *Proceedings of the 4th International Conference on Applied Cryptography and Network Security (ACNS'06)*, 2006.

[6] BitTorrent. http://www.bittorrent.com/press.html.

[7] BitTorrent. http://www.bittorrent.org.

[8] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S.Wallach. Security for structured peer-to-peer overlay networks. In *Proceedings of OSDI 2002*, December 2002.

[9] K. Cho, K. Fukuda, and H. Esaki. "The impact and implications of the growth in residential user-to-user traffic". In *Proc. ACM SIGCOMM*, 2006.

[10] Y. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *Proceedings of USENIX*, June 2004.

[11] J.R. Douceur. The Sybil Attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[12] EMule. http://www.emule-project.net.

[13] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross. "Insights into pplive: A measurement study of a large-scale p2p iptv system". In Proc. Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW2006, May 2006.

[14] S. Kandula, D. Katabi, M. Jacob, and A. Berger. "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds". In *Proc. NSDI*, 2005.

[15] V.T. Lam, S. Antonatos, P. Akritidis, and K. G. Anagnostakis. "Puppetnets: Misusing web browsers as a distributed attack infrastructure". In *"Proc. of ACM Computer and Communication Security"*, 2006.

[16] D. Malkhi, Y. Mansour, and M. Reiter. Diffusing without false rumors: On propagating updates in a byzantine environment. *Theoretical Computer Science*, 299((1-3)):289–306, 2003.

[17] D. Malkhi, E. Pavlov, and Y. Sella. Optimal unconditional information diffusion. In *15th International Symposium on DIStributed Computing (DISC 2001)*, 2001.

[18] D. Malkhi, M. Reiter, O. Rodeh, and Y. Sella. Efficient update diffusion in byzantine environments. In *20th Symposium on Reliable Distributed Systems (SRDS 2001)*, 2001.

[19] P. Maymounkov and D. Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric.". In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2002.

[20] Yaron Minsky and Fred Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, February 2003.

[21] N. Naoumov and K.W. Ross. "Exploiting p2p systems for DDoS attacks". In *"International Workshop on Peer-to-Peer Information Management"*, May 2006.

[22] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. Computer Communication Review 31(3), July 2001.

[23] PPLive. http://www.pplive.com/en/index.html.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM*, 2001.

[25] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[26] A. Singh, P. Maniatis, T. Roscoe, and P. Druschel. Using Queries for Distributed Monitoring and Forensics. In *Proc. of EuroSys*, 2006.

[27] A. Singh, T-W. Ngan, Druschel P, and D. Wallach. Eclipse Attacks on Overlays: Threats and Defenses. In *Proceedings of INFOCOM 2006*, April 2006.

[28] E. Sit and R.Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[29] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, 2001.

[30] D. Stutzbach and R. Rejaie. Improving Lookup Performance over a Widely-Deployed DHT. *proceedings of IEEE INFOCOM*, 2006.

[31] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC)*, 2006.

[32] TVUPlayer. http://tvunetworks.com/.

[33] D.S. Wallach. A Survey of Peer-to-Peer Security Issues. In *International Symposium on Software Security*, November 2002.

[34] xMule. http://www.xmule.ws/.

[35] Zattoo. http://www.zattoo.com/.

[36] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming. In *Proceedings of IEEE INFOCOM*, 2005.

[37] B.Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and John Kubiatowicz. "Tapestry: A resilient global-scale overlay for service deployment". *IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, Pgs. 41-53.*, January 2004.