ARTICLE IN PRESS

Computer Networks xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Preventing DDoS attacks on internet servers exploiting P2P systems

Xin Sun *, Ruben Torres, Sanjay Rao

School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Avenue, West Lafayette, IN 47907, United States

ARTICLE INFO

Article history:
Received 3 June 2009
Received in revised form 4 November 2009
Accepted 13 May 2010
Available online xxxx
Responsible Editor: C. Douligeris

Keywords:
P2P
DDoS
Membership validation
Experimental evaluation

ABSTRACT

Recently, there has been a spurt of work [1–7] showing that a variety of extensively deployed P2P systems may be exploited to launch DDoS attacks on web and other Internet servers, external to the P2P system. In this paper, we dissect these attacks and categorize them based on the underlying cause for attack amplification. We show that the attacks stem from a violation of three key principles: (i) membership information must be validated before use; (ii) innocent participants must only propagate validated information; and (iii) the system must protect against multiple references to the victim. We systematically explore the effectiveness of an active probing approach to validating membership information in thwarting such DDoS attacks. The approach does not rely on centralized authorities for membership verification, and is applicable to both structured (DHT-based) and unstructured P2P systems. We believe these considerations are important to ensure the mechanisms can be integrated with a range of existing P2P deployments. We evaluate the techniques in the context of a widely deployed DHT-based file-sharing system, and a video broadcasting system with stringent performance requirements. Our results show the promise of the approach in limiting DDoS attacks while not sacrificing application performance.

1. Introduction

Peer-to-peer (P2P) systems have matured to the point we have recently seen several commercial offerings [8–10]. Given the increasing prevalence of the technology, it becomes critical to consider how such systems can be deployed in a safe, secure and robust manner.

Several works [11–15] have studied how malicious nodes in a P2P 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 P2P system may impact the *external* Internet environment, by causing large-scale distributed denial of service (DDoS) attacks on web servers and other sites 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. The attacks are hard to detect and track-down as the packets being exchanged between the attacker and innocent nodes are not distinguishable from normal protocol packets.

While the community has been aware of the possibility of exploiting P2P systems to launch DDoS attacks for several years (for example [1]), a number of researchers have highlighted the criticality of the problem in recent years. The feasibility of exploiting the intrinsic characteristics of P2P systems for indirection attacks was first systematically shown in [2]. Since then, several works including our own [3–6,16] have demonstrated the generality of the problem, by showing that a variety of extensively deployed systems may be exploited to launch DDoS attacks. The systems include unstructured file-sharing systems such as Gnutella [3], and BitTorrent [4,5], DHT-based file-sharing systems such as Overnet [2], and Kad [6,16], and a video broadcasting system based on End System Multicast (ESM) [6].

* Corresponding author. Tel.: +1 765 409 1291.
E-mail addresses: sun19@purdue.edu (X. Sun), rtorresg@purdue.edu (R. Torres), sanjay@purdue.edu (S. Rao).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.05.021

Attacks can be significant-for example, [6] showed an attack where over 700 Mbps of UDP traffic was generated at the victim by exploiting Kad using 200 attacker machines. Creating the attack heuristics was as simple as modifying two source files and less than 200 lines.

While traditional ways to launch DDoS attacks such as DNS reflector attacks [17], or botnet-based DDoS attacks [18] are more widespread today, there is evidence to suggest that exploiting P2P systems to launch DDoS attacks in the wild is on the rise [7,19]. For instance, Prolexic Technologies has reported that they have observed what they term a DC++ attack [7] that involved over 300 K IP addresses. Discussions in the eMule forum [19,20] have indicated DDoS attacks on DNS servers exploiting the DHT-based Kad system. We present evidence of this in Section 2.3 through traffic measurements collected at the edge of a MiniPoP of an ISP. We believe it is imperative to systematically study the problem given the large-scale deployments of P2P systems, the emerging reports of attacks in the wild, and the relative lack of attention to the area.

In this paper, we seek to obtain a deeper understanding into the threats by studying the intrinsic design limitations of existing P2P systems which leave them vulnerable to such DDoS attacks. As a first contribution of this paper, we categorize all known DDoS attacks presented to date. In our classification, we focus on the underlying cause for achieving *amplification*. By amplification, we refer to the ratio of the number of messages (or bytes) received by the victim to the total number of messages (or bytes) sent by all malicious nodes. We focus on amplification since this is the key factor that determines whether an attack is attractive to a malicious node. We then articulate key design principles that P2P designers must follow to avoid these sources of amplification. The principles highlight the need to validate membership information before they are further used or propagated, and the need to protect against multiple references to the victim. While these principles are almost obvious in retrospect, the failure to follow the guidelines in a wide range of deployed systems, and the resulting repercussions are striking.

As a second contribution of this paper, we systematically explore the effectiveness of an active probing approach to validating membership information in mitigating the threats. We focus on this approach since it does not rely on centralized authorities for membership verification, and since it is applicable to both structured (DHT-based) and unstructured approaches. The key issues with an active probing approach are ensuring that the probes used for validation themselves do not become a source of DDoS, and dealing with benign validation failures that may occur due to packet loss, churn in group membership, and the presence of hosts behind Network Address Translators (NATs). We present simple mechanisms to address these issues, and show that with the mechanisms, the maximum amplification achievable by attackers can be bounded. We have incorporated these mechanisms in two contrasting applications – a DHT-based file-sharing system (Kad [21]), and a video broadcasting system (ESM [22]) with stringent performance requirements. Through extensive experimental evaluation, we show that the schemes may be suitably parameterized to ensure that DDoS attacks are effectively limited, while not sacrificing application performance.

2. DDoS attacks by exploiting P2P systems

Recently researchers have shown how a variety of P2P systems can be exploited to launch DDoS attacks on any Internet host such as a web server [3–6,23,16]. Each of these works presents attack heuristics on a specific system, and to date there have been several different attacks reported on five widely deployed P2P systems. In this section, we begin by presenting an overview of these systems in Section 2.1, and then summarize the attacks exploiting them in Section 2.2.

2.1. Systems background

Kad and *Overnet* are large-scale DHT-based file-sharing systems with each having more than one million concurrent users. Kad is supported by the popular eMule [21] client and its clones, while Overnet is supported by eDonkey [24]. These two systems are similar because they both implement the Kademlia [25] protocol, and differ primarily in implementation issues. In both systems, each participating node maintains a routing table with a subset of peers. For any given file, there are multiple "*index nodes*", each of which maintains a list of members who own that file. Index nodes are regular participants, who have an ID close to a file ID. Every node periodically publishes to the index nodes what files it owns. When a node wants to download a file, it first employs an iterative query lookup mechanism to locate an index node, and it obtains a list of members having the file from the index node.

BitTorrent is a very popular tracker based unstructured P2P system for file sharing. In BitTorrent, if a node wants to download a file (say a movie), it must first download the torrent file for that movie, which is published through out-of-band channels such as websites. The torrent file contains a list of trackers. A tracker is a central server which maintains the membership information of a swarm. Each node contacts one or more trackers to obtain a list of peers in the swarm, and starts exchanging data with the peers. Each node also contacts the trackers periodically afterwards to discover more peers.

Gnutella is another popular unstructured P2P file-sharing system which has a two-tier hierarchy. In Gnutella, when a node launches a query for a file, other nodes may reply with a query hit message that includes the IP and port of a peer that has the file. In addition, when the node requests the peer for the file, the format of the file request message is HTTP based.

ESM is one of the first operationally deployed P2P video streaming systems. It constructs a multicast tree on top of an unstructured overlay for data delivery, and employs a gossip-based membership management protocol. Each node periodically picks another node at random, and sends it a subset of the peers it knows. A node adds to its routing table any peer that it has not already known, and may use

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

these peers for various protocol operations such as parent selection.

2.2. Attacks

In any scalable P2P system, a node *A* may learn about another node *C* from a peer node *B*, as shown in Fig. 1(a). For example, this is required when a node locates index nodes, or obtains a list of sources for a file. However, this operation can be exploited by a malicious node *M* to redirect *A* to the victim *V*, as shown in Fig. 1(b). V can potentially be any Internet host such as a web server. In this section we summarize various ways in which this vulnerability has been exploited to cause large-scale DDoS attacks.

Attacks exploiting Kad and Overnet: While [6,2] have presented attacks on Kad and Overnet, respectively, we believe that most of these attacks are applicable to both systems. Thus we summarize them together here.

Index poisoning [2]: This attack is depicted in Fig. 2. Here, a malicious node *M* publishes to an index node *I* that the victim holds some file. Later any innocent participant *A* looking for a set of sources for the file will contact *I* and be redirected to the victim *V*. The redirected participants will try to establish TCP connections to the victim in order to download the file. This could not only result in TCP SYNs, but also result in successful TCP connections if for instance an actual web or mail server were running on the victim. The bar for such an attack could be raised by not requiring nodes to insert their IP and port information in application layer messages to begin with, however we note that index poisoning attacks could still occur if malicious nodes could conduct packet level spoofing.

NAT-buddy exploit: This attack may be viewed as a variant of the *Index poisoning* attack. It exploits a NAT traversal mechanism that is commonly used in today's P2P systems, including both Kad and Overnet. In such a mechanism, a node behind a NAT could select a public node as its "buddy". When the NAT node publishes a file, information about the buddy is included in its publish message to the index nodes. A malicious node exploits this to launch a DDoS attack, by advertising the victim as its buddy. When innocent participants obtain a set of sources from the index node, they contact the buddy (victim) as per normal protocol operations resulting in a DDoS attack on the victim. This attack is briefly discussed in [2]. The attack could potentially be prevented by modifying the underlying protocols so that NAT nodes are required to publish content

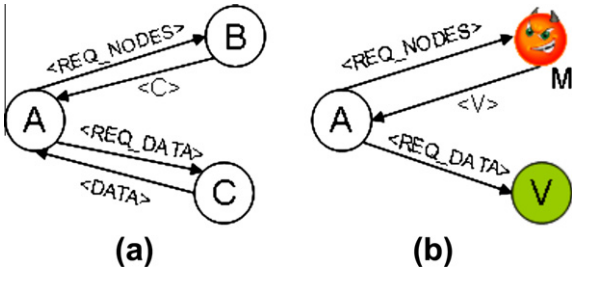


Fig. 1. (a) Normal operation and (b) attack.

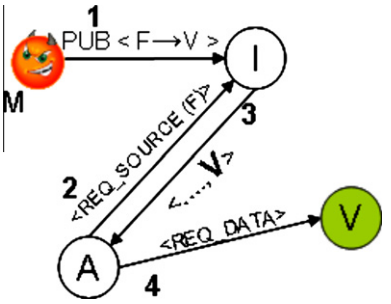


Fig. 2. Index poisoning attack in Overnet.

through their buddies, however the protocol modifications must ensure the overheads at the buddy are not increased significantly, and must include mechanisms to ensure malicious nodes cannot deliberately increase the overheads on the buddy, for instance by providing information about non-existent files.

Search hijack [6]: This attack exploits the parallel lookup mechanism of Kad and Overnet, where multiple nodes may be included in a reply to a query. In particular, when a malicious node receives a search query from an innocent participant, it includes in the reply multiple (fake) logical identifiers, all sharing the IP address of the victim. This results in the innocent participant querying the victim multiple times. Note that in order to enable distinct users behind the same NAT to participate in the system, Kad, Overnet, and many other P2P systems allow a participating node to communicate with multiple logical identifiers even though they share the same IP address.

Routing table poisoning [2]: This attack is specific to Overnet. It exploits the announcement messages, which enable nodes to announce themselves to others. In particular, a malicious node may put the victim's IP address in an announcement message and send it to an innocent participant, by exploiting a vulnerability specific to Overnet. This results in the innocent participant adding the victim to its routing table, and using it for normal protocol operations. Like with index poisoning attacks, the bar for such an attack could be raised by not requiring nodes to insert their IP and port information in application layer messages, however the attacks could still occur if malicious nodes could conduct packet level spoofing.

Attacks exploiting BitTorrent: Sia [5] presents an attack where malicious nodes in a BitTorrent swarm can report to the tracker that the victim is a participating peer, through packet spoofing. This would cause the tracker to redirect innocent participants to the victim, who in turn repeatedly initiate TCP connections to the victim. Harrington et al. [23] present a variant of this attack where the tracker itself is malicious and falsely tells innocent participants in the swarm that the victim is a participating peer. Defrawy et al. [4] present an attack where the attacker publishes fake torrent files to web sites of known torrent search engines. Each of the fake torrent files includes the victim's IP several times, each with a different port. Innocent participants who download the torrent files believe that there are different trackers running on the same IP, and repeatedly try to connect to each of the trackers.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

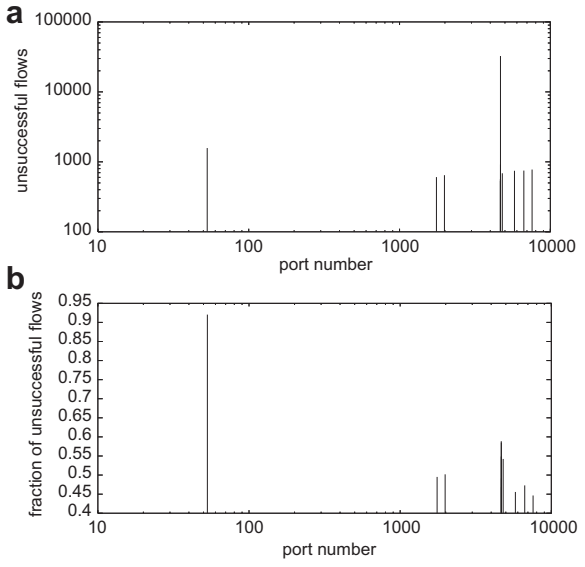


Fig. 3. (a) Number of unsuccessful Kad flows sent to ports that received more than 500 unsuccessful flows. (b) Fraction of unsuccessful Kad flows to ports that received more than 500 unsuccessful flows.

Attacks exploiting Gnutella: Athanasopoulos et al. [3] present an attack where malicious nodes include the victim's IP address in their replies to file query messages sent by innocent participants. The victim in this attack is a web server which hosts some files. Further, the file names in the reply messages are constructed in such a way that the file requests sent by innocent participants to the victim will look exactly like genuine HTTP requests from normal web clients. This causes the victim to upload an entire file to the redirected nodes.

Attacks exploiting ESM: This attack is presented in [6], where a malicious node *M* exploits the push-based nature of the gossip protocol in ESM. In particular, a malicious node generates false information about the victim being part of the group, and aggressively pushes the information as part of its gossip messages to innocent participants. In addition, the malicious node includes the victim's IP several times in a gossip message, each with a different logical ID, similar to the *Search hijack* attack in Kad.

2.3. Potential evidence for real attacks in the wild

We have analyzed a one day trace collected at the edge of a MiniPoP of an ISP, where we found potential evidence of abnormal traffic to DNS servers from peers running the Kad system. Our trace consists of flow-level logs.[1] Our analysis considers all Kad UDP traffic between hosts inside the network and hosts in the Internet. Fig. 3(a) shows the number of unanswered flows (i.e., flows for which only outbound traffic was seen), as a function of the destination port number. Impulses were plotted only for ports that received more than 500 unanswered flows. The spike at port 4672 is expected since this is the default UDP Kad port. However, it is interesting that there is a spike at port 53 (DNS port). Fig. 3(b) shows the fraction of unanswered flows out of the total outgoing flows, as a function of the port number. The graph shows that port 53 has the highest ratio of unanswered flows with 92%.

We considered whether these results could be due to actual Kad clients running on port 53, which are aiming to hide their presence in firewalled networks. We believe there are two reasons this is unlikely to be the case. First, a majority of flows (92%) to port 53 are unanswered, while for other ports, the percentage of unanswered flows was at most 60%. Second, manual investigation (e.g., by doing an nslookup on the target IP addresses), indicated that most destinations were DNS servers (e.g., in China and Thailand).

We believe these results are abnormal, and potentially point to a DDoS attack exploiting the Kad system. Our results also corroborate discussions in the eMule forum [19], and works by other researchers [20], which further strengthens our belief that these results may be due to a DDoS attack. Finally, we note that others have observed DDoS attacks in the wild exploiting other P2P systems [7].

3. Eliminating attack amplification

While specific solutions may potentially be designed for each of the attacks listed in Section 2, the fact that a multitude of attacks have been reported against a range of systems leads us to explore more general principles and techniques to protect against the entire class of attacks. In this section, we dissect the attacks in Section 2.2, and identify a few underlying patterns which lead to large attack amplification. Based on the insights, we articulate a few generic design principles for P2P developers.

3.1. Classifying attacks by amplification causes

Based on the source of amplification, the attacks described in Section 2.2 can be classified as follows (also in Table 1):

3.1.1. Repeated packets to victim

This is the simplest source of amplification, where an innocent participant may keep using the victim it learned from a malicious node for various protocol operations, despite the fact that it has never been able to communicate with the victim. Surprisingly, many of the systems have this vulnerability.

3.1.2. Multifake

In this case, a malicious node may convey the same victim multiple times to an innocent participant, by disguising the victim as multiple different members of the system. This is a major source of amplification in the case of the *Search hijack* attack on Kad and Overnet, the attack on ESM, and the attack on BitTorrent involving fake torrent files [4]. This heuristic can be generalized to mount an attack on a *network*, by having the malicious node including

[1] A flow is identified by the 5-tuple comprising source and destination addresses and ports, and protocol. A UDP flow starts when the first packet is observed, and is considered to end if no packet is seen in any direction for 200 s. If a packet is never observed in the reverse path, then the UDP flow is said to be unanswered.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

Table 1
Classification of DDoS attacks by their major source of amplification.

System	Attack	Repeated packets to victim	Multifake	Delegation	Triggering large reply from victim
Kad & Overnet	Index poisoning [2]			√	
	NAT-buddy exploit [2]			√	
	Search hijack [6]		√		
Overnet	RT poisoning [2]	√			
BitTorrent	Fake report to tracker [5]			√	
	Malicious tracker [23]	√			
	Fake torrent file [4]		√	√	
Gnutella	Gnutella attack [3]				√
ESM	ESM attack [6]	√	√	√	

fake membership information about several different IP addresses, all of them belonging to the same network.

3.1.3. Delegation

For attacks in this class, amplification is achieved because the innocent participants spread fake membership information. This is a major source of amplification in the *Index poisoning* and *Nat-buddy exploit* attacks in Kad and Overnet where the index nodes propagate the victim, in the attacks on BitTorrent, where the trackers and torrent websites propagate the victim, and in the attacks on ESM, where innocent participants gossip about the victim to each other.

3.1.4. Triggering large reply from victim

This class includes attacks such as those on Gnutella [3], where an entire file is sent in response to a query, leading to high amplification.

3.2. Principles for robust design

We next present several key design principles which if followed could eliminate all the amplification causes.

- *Validate before use*: Each node must validate membership information it learns, before adding the information to its routing table and/or using it for protocol operations. This eliminates the *Repeated packets to victim* vulnerability since it ensures that no protocol packets will be sent to a node until it has been validated. Further, it also eliminates the *Triggering large reply from victim* vulnerability because an innocent node would not send file requests to the victim server since the server cannot be successfully validated.
- *Validate before propagation*: Any membership information must be first validated before being propagated to any other nodes. This ensures that innocent participants do not propagate fake membership information. A potential source of attack amplification such as *Delegation* is avoided as attackers are now constrained to infecting the innocent participants directly. It is worth pointing out that this principle must be followed for all operations involving exchanging membership information. For instance, in Kad, the principle is followed when nodes learn other members through search process, but not followed when index nodes receive publish messages from other nodes, thus Kad is still vulnerable to the *Index poisoning attack*.
- *Preventing multiple references to the victim*: This principle guards against amplification due to *Multifake*, where an attacker conveys the same victim multiple times to an innocent participant by disguising the victim as multiple distinct members of the system. A particularly interesting case is where the attacker is able to convey the victim multiple times in a single membership message, as in the *Search hijack* attack, because this has interesting implications for our defense mechanisms as we will see in Section 4.

4. Enhancing DDoS resilience

As discussed in Section 3, the key principles involved in limiting DDoS amplification is validating membership information. Several approaches may be adopted to this end, and we discuss this further in Section 4.1. In this paper, we explore in depth the potential of an active probing approach to validating membership information. We discuss the considerations that motivated us to focus on such an approach, and details of the approach in the rest of the section.

4.1. Approaches for validating membership information

We discuss possible approaches that may be adopted for validating membership information:

- *Use of centralized authorities:* Centralized authorities can simplify validation of membership information by providing signed certificates that indicate that a member belongs to a group. The certificates may be distributed through the membership management mechanisms, and enable a participant to verify the membership information corresponds to a genuine member. However, the existence of central authorities cannot be assumed in many P2P deployments, and we only consider mechanisms that do not rely on their existence.
- *DHT-specific approaches:* It may be feasible to leverage the properties of DHTs, and design solutions tailored to them. For instance, one approach is to assign each node an ID dependent on its IP address, for instance

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

the hash of its IP address [26]. An attacker that attempts to provide fake membership information in response to a search query must then ensure that (i) the victim ID obeys the necessary relationship to the victim IP; and (ii) the victim ID is close to the target ID of the search. These dual constraints on the victim ID may be difficult to simultaneously satisfy, complicating the attack. Issues that need to be addressed with such an approach include the need to accommodate multiple participants behind the same NAT which share a common IP address, and the fact the victim ID is only loosely constrained by the target ID. While the approach has promise, we do not explore it further since our focus in this paper is on mechanisms that apply to both structured and unstructured approaches.

- *Corroboration from multiple sources:* Another approach to validating membership information is based on prior works on Byzantine-tolerant diffusion algorithms [27–30]. In this approach, a node will accept and communicate with a newly-learned peer only if it learns about the peer from multiple other nodes (say k). Such schemes are susceptible to attacks where the attacker has control over k or more nodes, because then he can make these malicious nodes lie about the same fake membership information, and defeat the corroboration. Such attacks may be particularly easy to conduct in conjunction with a Sybil attack. Another concern with the approach is that from a performance perspective, the larger the k value, the longer it may take to receive responses from all k nodes, which can slow down convergence and performance [31].

4.2. Validating peers through active probing

In this paper, we explore the potential of an active probing approach for membership validation. We focus on such an approach because (i) it does not rely on centralized authorities; and (ii) the technique applies to both unstructured approaches as well as structured DHT-based approaches. Thus, the technique has potential to be widely applicable to a range of existing P2P deployments.

In the approach, when a node receives a *membership message*, it probes any member included that it did not know before, and does not send further messages to it or propagate it unless it receives a response. We use the term *membership message* very broadly to refer to any protocol message that contains information about other participants in the group. This includes, for example, search replies and file publish messages in Kad, Overnet and Gnutella and gossip messages in ESM. It also includes messages where a node may directly announce itself to other nodes.

It is possible that a probe sent to the victim could trigger a spurious response from some other application running on the port under attack. To prevent this, the probe request and response should contain: (i) a predefined byte pattern unique to the application and distinct patterns for the request and the response and (ii) a sequence number in the request which will be incremented in the response.

While probing-based validation has potential, the key issues are ensuring the probes themselves do not become a source of DDoS, and dealing with benign validation failures that may occur due to packet loss, membership churn, and the presence of hosts behind Network Address Translators (NATs). We discuss heuristics to handle these issues in the rest of the section.

4.3. Preventing DDoS exploiting validation probes

To prevent validation packets from being a source of DDoS attacks, two broad approaches may be adopted. A first approach involves coordination across members in the system, so as to ensure only a subset of members conduct the validation. Such coordination mechanisms are themselves subject to attack when malicious nodes are involved, given that members may not be truthful in sharing information about validation failures. While reputation mechanisms such as [32] may be used to address these concerns, these mechanisms are often themselves vulnerable to attacks such as whitewashing (see [33] for a survey of schemes and attacks). Further, many schemes (for example [32]) rely on centralized authorities or pretrusted nodes, and we are interested in solutions not depending on their existence. Thus we focus on mechanisms that only rely on locally observable events. More specifically we employ two schemes as described below:

4.3.1. Source-throttling

This scheme seeks to limit the attack traffic that a single membership message from a malicious node can induce. This prevents attacks like the *Search hijack* in Section 2.2, which enabled an attacker to achieve significant attack amplification. In the scheme, when a node receives a membership message, rather than try to validate all the members included in the message at the same time, validations are performed to at most m members initially, where m is a parameter. A new validation to an additional member is conducted only when a previous validation is successful. This mechanism ensures that a single message from an attacker can trigger at most m validation messages to the victim under attack. Combined with the rest of the validation framework, this limits the total attack amplification achievable by the attacker to m, as we discuss in Section 5.1.

While small values of m are desirable to keep attack amplification small, the main concern is the potential impact on performance. In particular, validation failures may occur for benign reasons resulting in unnecessary throttling. Consequently genuine information in a received membership message may be ignored. Further, the validation process may incur some delay, possibly resulting in higher latencies or convergence times for the application. In Section 7, we evaluate the feasibility of employing small m values for real applications.

4.3.2. Destination-throttling

The *source-throttling* scheme by itself could prove highly effective in thwarting the attacker in many circumstances since it bounds the amplification the attacker can achieve. We augment this scheme with a simple mechanism we term *destination-throttling*, that can limit the number of packets each innocent participant sends to the

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

victim. We note that even with *destination-throttling* the victim can receive $O(N)$ attack packets, where N is the total number of participants in the system. However, since the amplification is bounded, it would require the combined set of all attacker machines to send $O(N)$ messages to innocent nodes to redirect them to the victim.

An obvious solution to limiting packets that each innocent node sends to the victim is to black-list sources that cause repeated validation failures, and ignore further membership messages from them. This heuristic by itself does not suffice however, since there are potentially many sources, and further the malicious node may mount a Sybil attack. Instead, with *destination-throttling*, repeated validation failures to a destination are used as an indication that it is under attack, and future validations are not sent in such a case.

A destination could refer to an ⟨IP, port⟩, an IP, or an entire network. The network to which an IP belongs may be determined by a longest prefix match on a database of prefixes and netmask information extracted from BGP routing table snapshots [34,35]. Such a database could be obtained by a client in an out-of-band fashion at the start of the session, and the information is unlikely to change over the duration of a typical session. Alternately, though less accurately, an IP could be assumed to belong to a /24 network. In the rest of the paper, we use the terms "prefix" and "network" interchangeably.

Every client maintains the total number of validation packets, and the number of failed validations to each ⟨IP, port⟩, IP, and prefix. Each validation failure is associated with a time-stamp indicating when the failure occurred, and only failures that occurred in a recent time window T are considered. A validation packet is suppressed if either the ⟨IP, port⟩, IP address, or prefix is suspected under attack. An ⟨IP, port⟩ is suspected under attack if more than F_{ipport} failures have been observed to it. A prefix (IP) is suspected under attack if it has seen at least $F_{prefix}(F_{ip})$ failures that involve $D_{prefix}(D_{ip})$ distinct IPs (⟨IP, port⟩ pairs). The set of parameters must be chosen so that the likelihood of destinations being falsely suspected under attack due to benign validation failures is small. We discuss this further in Section 7.

With the scheme as above the total validation failures to a prefix before it is suspected under attack could be as low as F_{prefix}, and as high as $F_{ipport} * D_{ip} * D_{prefix}$. To better contain the magnitude of a potential attack, once F_{prefix} failures have been seen to the prefix, validations are permitted only if there has been no prior failure to the IP. With this modification, at most $F_{prefix} + D_{prefix}$ validation failures are allowed to the prefix. A similar heuristic is applied to failures to individual IPs.

The *destination-throttling* scheme could potentially be exploited by malicious nodes to create attacks on the performance of the P2P system itself. We analyze the potential for such attacks in Section 5.2. A variant of the *destination-throttling* scheme that could be used to raise the bar against such attacks is to consider a prefix under attack if the percentage of failed validations to a prefix exceeds a threshold. While we believe such a variant could be easily integrated with our solution, we focus on a solution based on the total validation failures to each prefix to ensure the total number of packets sent by each innocent node to a victim prefix can be bounded under DDoS attacks.

4.4. Avoiding validation failures with NATs

Many P2P systems employ *NAT-Agnostic* membership management operations. In these systems, when member B propagates information about member X to member A, there is no indication as to whether X is behind a NAT or not. This can result in benign failures with probing-based validation. In particular, if X were behind a NAT, a validation packet sent by A to X will not successfully reach X unless X had previously contacted A.

While alleviating communication issues with NATs is an ongoing area of work [36,37], these techniques are not always effective with symmetric NATs [38], which is the most restrictive type of NATs, and accounts for close to 30% of NATs [36]. With symmetric NATs, different connections initiated by the same internal node generate different external IP and port mappings. Incoming packets are only allowed from the public nodes to which packets have been sent. Two nodes both behind symmetric NATs cannot communicate with each other.

To handle NATs (including symmetric NATs) and firewalls, we require that membership management operations are *NAT-Aware*. In particular, membership information about nodes behind NAT are propagated with a flag indicating they are behind NAT. When a node A learns about X, it probes X only if it is not behind a NAT, thereby avoiding benign validation failures. We discuss potential attacks on the scheme where a malicious member may falsify information regarding whether another member is behind a NAT in Section 5.

4.5. Illustrating the scheme

Probing-based validation mechanisms may be easily integrated with various P2P systems, to protect the systems from the exploits described in Section 2.2. As a concrete example, Fig. 4 illustrates the steps to prevent attacks exploiting buddy mechanisms in Kad and Overnet with our scheme: (1) Node N is behind a NAT, and sends an advertisement to node I indicating that a public node B is its buddy; (2) I validates B, and accepts information

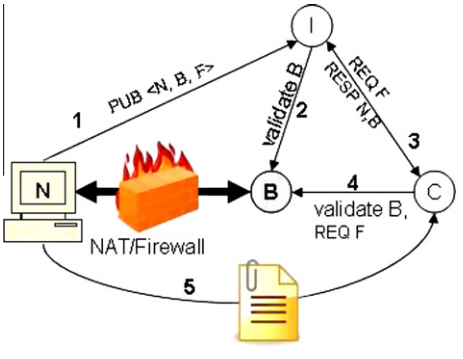


Fig. 4. Complete sequence of steps for normal buddy operations with probing-based validation mechanisms.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

only on successful validation; (3) C obtains information that N has the file and B is the buddy; (4) C validates B, and then sends it a request; (5) this is relayed to N which then sends C the file. Note that steps 2 and 4 are the validation messages that are triggered with our framework. The other exploits described in Section 2.2 are similarly handled, and we omit the details.

5. Analysis

In this section, we analyze the effectiveness of probing-based validation mechanisms in thwarting DDoS attacks. We also analyze the vulnerability of the mechanisms to new attacks that may impact application performance, and suggest refinements to minimize the impact.

5.1. DDoS attacks

We first consider DDoS attacks on hosts not participating in the P2P system, which is the primary focus of our paper, and what we view as a more critical threat. We then present possible refinements for handling attacks on participating nodes.

5.1.1. DDoS attacks on hosts not in the P2P system

We discuss the key measures of interest:

- *Message amplification*: this is the ratio of the number of messages received by the victim to the total number of messages sent by all malicious nodes. We observe that with source-throttling, the *message amplification* is at most m, independent of the number of malicious nodes. To see this, consider that a DDoS attack on a victim not in the P2P system consists entirely of validation messages. Each validation message must be triggered by a membership message directly received from a malicious node. This is because innocent members only propagate membership information they can validate, and hence do not propagate the victim. Thus the *message amplification* achievable is bounded by the maximum number of validations that may be sent to the victim due to a single membership message from an attacker node. This is bounded by m, by the throttling heuristics.
- *Bandwidth amplification*: this is the ratio of attack traffic received by the victim to the total traffic sent by all attacker nodes. More, precisely, the *bandwidth amplification* can be expressed as: $m * \frac{validation\_message\_size}{membership\_message\_size}$. Given that a validation message is small in general and smaller or at most comparable to the size of the membership message, the *bandwidth amplification* is also bounded by m.
- *Attack magnitudes*: The destination-based throttling scheme ensures that the total number of packets sent by each participating node to a victim is bounded. In particular, at most $F_{prefix} + D_{prefix}$ packets are sent to any victim prefix over a period of time T, where T is the time for which a failed validation is considered. Further, since this scheme is entirely based on the destination to which validation failures are observed, and does not depend on the source which triggered the validation, the bound holds irrespective of the number of attackers or under Sybil attacks. We note that the victim can still receive $O(N)$ attack packets, where N is the total number of participants in the system. However, we believe this is not a significant concern because the amplification is bounded by m, and the attacker must send $O(N)$ messages to innocent nodes to redirect them to the victim. While it may be possible to design mechanisms that can ensure not all innocent participants send attack packets to the victim, such mechanisms are likely to involve coordination across the nodes. Such coordination mechanisms are not only complex, but also are themselves subject to possible attack when malicious nodes are involved. We made a deliberate decision to avoid such mechanisms given the feasibility of bounding attack amplification without them.

Man-in-the-middle attacks: The above analysis assumes an attacker model where malicious nodes can only join the P2P system as regular participants. A more sophisticated attacker could potentially conduct a man-in-the-middle attack, perhaps by compromising routers. In particular, an attacker could intercept and respond to validation packets sent by innocent participants to the victim, tricking these participants into thinking the validation is successful. This is a potential concern because the tricked participants, which we term *delegates*, could propagate information about the victim, and consequently the amplification bounds above do not hold.

We note that to be effective, that attacker must strategically intercept validation packets from a moderate number of delegates, and this may not be trivial. If validations are intercepted from too many delegates (for instance, the man-in-the-middle is located close to the victim), the attacker is likely to see all validations to the victim; if the validations are intercepted from too few delegates, the total traffic induced at the victim due to the delegates is small.

The message amplification under man-in-the-middle attacks may be expressed as $\frac{m*M_{redirm}+M_{del}}{M_{redirn}+M_{MIM}}$. Here M_{redirn} is the number of redirection messages sent by the malicious nodes, and M_{MIM} is the number of validations to the victim that are intercepted by the attacker to conduct the man-in-the-middle attack. M_{del} is the number of validation messages received at the victim induced by membership information propagated by the delegates. Let us assume that the maximum number of innocent participants to which each delegate could spread information about the victim is K. This bound could be achieved by simply having a delegate conduct periodic revalidation, and limiting the rate at which it spreads information about innocent participants, or by having the delegate conduct a revalidation each time it spreads membership information to K other participants. Then, M_{del} is bounded by $K * M_{MIM}$, and the overall message amplification is $\frac{m*M_{redirn}+K*M_{MIM}}{M_{redirn}+M_{MIM}}$. It is easily verified that this quantity is between m and K. Finally, the amplification is likely to be even smaller because the analysis does not consider that the man-in-the-middle will not only see traffic due to validations from the delegates, but also normal protocol traffic packets from the delegates.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

5.1.2. DDoS attacks on participating nodes

We next discuss the case when a participating node is the victim of the attack. A straightforward way to extend the probing-based validation mechanisms is to require the victim to explicitly deny validation requests if it receives them at an excessive rate. On receipt of such a denial, an innocent participant considers the validation to have failed. While this can help limit the attack, this is not enough because innocent participants that were previously able to successfully validate the victim, (which we again refer to as *delegates*), could still continue to propagate the membership messages.

The validation traffic seen at the victim due to membership information spread by the delegates depends on (i) the number of delegates; and (ii) the rate at which each delegate spreads the victim to other innocent participants. The second factor is under the control of the delegates and is easily controlled. However, it becomes important to ensure the total number of delegates does not grow in unbounded fashion.

A possible heuristic to limit the number of delegates is to have each node A restrict the number of other nodes that may successfully validate it over any window of time, and deny all other requests. In addition, as described above, delegates are required to periodically revalidate A, and if a revalidation fails, they are required to stop sending further packets to A, and stop propagating A to others.

An attacker could exploit this heuristic by sending a large number of validation messages to the victim, thereby causing the victim to deny validations from innocent peers. We term such an attack a *Disconnection Attack*, and note that the attack targets the performance of the P2P system itself. It is unclear how attractive such an attack is since it would require the malicious node to send $I * U$ messages each revalidation period, where I is the delegate limit per node, and U is the number of nodes to be disconnected. While it is perhaps not hard to disconnect a single node, disconnecting a significant fraction of nodes in the system involves a large number of messages given Kad and Gnutella have over a million users. For instance, if $I = 10{,}000$, $U = 1000$ (representing a disconnection of 0.1% of all participants), and assuming a revalidation is conducted every minute, a malicious node would need to send about 160,000 messages per second. On the other hand, this revalidation traffic would not be a significant burden for innocent nodes-considering that typically each node has a few hundred neighbors, the revalidation traffic required of a typical node would be about 2 messages per second. Finally, we note that if the attacker only controlled a small number of IP addresses or prefixes, such disconnection attacks could be further prevented by black-listing peers that repeatedly send validation messages.

In structured DHT-based overlays, a concern with limiting delegates is that a node may deny a validation request from a peer with an ID close to its own, thereby impacting the DHT structure. While this may not be a concern ordinarily if the delegate limit is chosen conservatively, it may be possible for an attacker to create such a situation by redirecting a lot of innocent nodes (whose IDs are far from the victim's ID) to the victim. To defend against this type of attack, the scheme may be modified for structured overlays to have nodes probabilistically accept a peer as a delegate, based on the ID distance between the two. The closer the peer, the higher the probability. The scheme is effective given that a node will not have too many peers with an ID close to it, and an attacker has no control over the ID of an innocent node. We defer a more detailed investigation of these issues to future work.

5.2. Attacks on destination-throttling

In this section, we discuss possible attacks on the destination-throttling scheme. We note these attacks do not apply to the source-throttling scheme, which was the primary mechanism in limiting attack amplification. We identify two variants of the attack:

- *Attacks on destination-throttling:* A malicious node M may flood another node A with several fake membership entries corresponding to a victim prefix. The resulting validation failures could force A to throttle the prefix, resulting in A being disconnected from valid participants in that prefix.
  A similar attack could be performed to cause A to be disconnected from a single node rather than a prefix. A malicious node M may do so by flooding A with incorrect port information about the victim node. However, this attack is less attractive for an attacker than the previous one, since it only disconnects A from a single node. While our analysis below focuses on attacks to prefixes, we believe that similar arguments will hold for attacks to a single node.
- *Attacks on NAT-Aware mechanisms:* When a malicious node M propagates membership information to node A, it may falsely indicate that a participating node N has a public IP address, even though N is behind a NAT. This could induce validation failures from A to N, potentially resulting in A throttling the prefix to which N belongs.

We observe that a malicious node must send at least $d = \frac{F_{prefix}+D_{prefix}}{m}$ messages to disconnect a node from participants in one prefix. This is true since at least $F_{prefix} + D_{prefix}$ validation failures are required to a prefix before future validations to it are suppressed, and at most m validation failures may be induced by a single membership message due to the source-throttling mechanisms. Based on our parameterization results in Section 7, we expect d to be of the order of 10 messages.

While such attacks are theoretically feasible, we believe they are not attractive for an attacker in practice, since a large number of messages must be sent to cause a noticeable degradation in application performance. We note the messages involved to disconnect a node depends on the number of prefixes spanned by participating nodes, which is of the order of tens of thousands, and at least an order of magnitude larger than the attacks in Section 5.1.2. Further, in a system like Kad, to limit the ability of n participants to access a file, an attacker must disconnect the participants from all prefixes with nodes that have the file. Further, popular files are likely to be distributed across many prefixes. Disconnecting n participants from p prefixes requires at least $d * n * p$ messages. This can be high, considering that the system can have millions of clients distrib-

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

uted over tens of thousands of prefixes. In addition, note that any such disconnection is temporary, since validation failures are timed out after time T, and sustaining the disconnection requires continued messages from the attacker.

It is possible to limit the extent of disconnection attacks if the malicious nodes are localized to a small number of prefixes. In particular, once a prefix Z is suspected of being under a DDoS attack, source IP prefixes that have triggered a validation failure to Z are black-listed. Further validations to Z are sent only if they are triggered by source IP prefixes that have not been black-listed. We note however that the fact malicious nodes are localized to a small number of prefixes may not be known apriori. Thus, it is desirable to dynamically tune the scheme to ensure the extent of disconnection attacks is limited if the number of prefixes spanned by malicious nodes is small, however ensure the scheme is not susceptible to DDoS attacks if the number of attacker prefixes is large. To handle this, once prefix Z is suspected of being under a DDoS attack, a validation triggered by a source prefix that is not black-listed is sent only if the source prefix belongs to a randomly selected fraction f of all possible prefixes. f is initialized to 1, and dynamically reduced, for example by a factor of 2 on each validation failure after the number of black-listed source prefixes exceeds a threshold. Intuitively, we expect the scheme to stabilize at f values which are inversely proportional to P, the number prefixes spanned by attackers. Further, the number of additional validation messages to the victim is at most $\log P$. We can bound this quantity even more, by not permitting any further validations once f goes below a threshold.

Finally, the *destination-throttling* scheme as presented in the paper considers a prefix under attack if the number of validation failures to a prefix exceeds a threshold. A potential approach to raise the bar against disconnection attacks is to consider a prefix under attack if the percentage of validation failures to the prefix exceeds a threshold. We have focused on the former variant in this paper to bound the total number of packets sent by each innocent node to a victim prefix under DDoS attacks. However, the latter variant could be easily integrated if the need to prevent disconnection attacks is viewed more critical.

6. Evaluation methodology

The primary question driving our evaluations is how effective the validation framework is in preventing DDoS attacks without sacrificing application performance. To understand this, we have integrated our framework into a file sharing application (Kad) and a video broadcasting application (ESM), two mature and contrasting P2P applications, with very different membership management designs and performance requirements. In the rest of the section, we present our evaluation goals, metrics and our experimental methodology.

6.1. Evaluation goals

We have the following goals:

- *Performance under normal conditions:* We study the impact that the validation framework has on application performance under normal conditions when there are no attacks. This enables us to determine the performance of the throttling heuristics under benign validation failures, arising due to packet loss, churn, and NATs.
- *Impact of throttling parameters:* As our analysis in Section 5 indicates, the attack amplification, and attack magnitudes achievable with the validation framework are directly dependent on the choice of the m parameter for the source-throttling scheme and the various F and D parameters for the destination-throttling scheme. On the one hand, it is desirable to keep these parameters small to limit DDoS attacks. On the other hand, the smaller the values, the greater the potential impact on performance. Thus our evaluations explore how various parameters of the throttling mechanisms impact application performance, and seek to identify operating ranges where the impact is small.
- *Impact on contrasting applications:* The performance with the schemes is dependent on the application itself. Hence, we explore the issues in the context of two very different applications, DHT-based structured file-sharing Kad and unstructured video broadcasting ESM.
- *Performance under attacks:* Finally, we evaluate the benefits of the validation framework both in limiting DDoS attacks, and in preventing degradation in application performance in the presence of malicious nodes.

6.2. Performance metrics

In our evaluations with file distribution applications (Kad), we consider the fraction of successful searches, and the time a successful search takes to locate an index node. For video broadcast (ESM), we consider the fraction of the streaming video rate received by participating nodes and the join time of nodes to the multicast tree. In addition, in evaluating the *destination-throttling* scheme, we measure the number of prefixes (as well as IPs, and $\langle$IP, port$\rangle$s) blocked by the scheme, in the absence of attackers.

6.3. Methodology

Our experimental methodology employs experiments both on the live Kad network, and on Planetlab.

- *Live Kad experiments:* The performance of both throttling schemes is sensitive to the extent to which benign validation failures are seen in realistic application deployment settings. This in turn depends on realistic churn rates, packet loss rates, and the fraction of participating NAT hosts. In addition, the performance of the destination-throttling scheme is sensitive to the number of participating nodes that share an IP prefix, and the number of participating nodes that share an IP address. To evaluate the performance of the schemes under such realistic conditions, several of our experiments are conducted on the live Kad network. We do this by implementing our validation framework in a Kad client, and having it join the live Kad network. We compare the performance of an unmodified Kad client on the live network, with multiple instances of the modified Kad client running different parameter settings.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

- *Planetlab experiments:* One limitation of evaluations on live Kad is that the throttling schemes are implemented only on the clients we have control over. To evaluate the throttling schemes in settings where all participating clients implement the validation framework, we evaluate Kad on Planetlab. In addition, Planetlab evaluations enable us to compare the performance of the schemes in the absence of attacks, and under attack scenarios.

Since there are no long-running live ESM broadcasts, experiments on live ESM deployments are not feasible, and our ESM experiments are conducted on Planetlab. To ensure realism, our experiments with ESM leverage traces from real broadcast events [22] to model the group dynamics patterns, and bandwidth-resource constraints of nodes. Since most nodes on Planetlab have public IPs, we emulate NAT connectivity restrictions by implementing packet filtering to ensure that two Planetlab incarnations that are behind a NAT cannot communicate with each other. Note that our emulations model *Symmetric NATs*, the properties of which are elaborated in Section 4.4.

7. Parameterizing the validation framework

In this section, we present results evaluating the impact of the validation framework and its various parameters on the performance of Kad and ESM. Our goal is to identify the possible sweet-spot parameters that are small enough to minimize the DDoS attacks, yet large enough to tolerate most benign validation failures. We implemented the throttling schemes in real Kad and ESM clients. For comparison, in our experiments we also ran unmodified Kad and ESM clients, which we refer to as *Base-Kad* and *Base-ESM*. The experiments with Kad in this section were conducted in the live Kad network, while the experiments with ESM were conducted on Planetlab. All the experiments were conducted in the absence of attackers.

7.1. File sharing application

We parameterize the source-throttling scheme in Section 7.1.1 and the destination-throttling scheme in Section 7.1.2.

7.1.1. Source-throttling: impact of m

As described in Section 2.1, when a Kad node conducts a search for a keyword or a file, it issues queries, and receives replies containing membership entries. This enables the node to locate index nodes for the keyword or the file. The source-throttling scheme impacts the search performance because entries returned in a reply message may not be fully utilized, and the validation process may incur some extra delay, as we have explained in Section 4.3. A factor that may affect the results is the number of membership entries returned in a reply message. In Kad, this number is a client-specified parameter which can be set by the node, and is included in the query messages sent to peers so they will know how many entries to include in the replies. In the mainstream implementations such as eMule and aMule client software, the default values for this number are 2, 4 and 11, depending on the type of the search. While we set it at 11 for our experiments, we also study the sensitivity of our results to this parameter.

We first measure the impact of the m parameter on application performance. We let four versions of our modified Kad client (i.e., with source-throttling) run in parallel, with each version running a different value of m (i.e., one version with $m = 1$, one version with $m = 2$ and so on). In addition, we let an unmodified client (i.e., Base-Kad) run at the same time. A random set of 1000 keywords from the English dictionary were picked, and each client conducted one search for each of these keywords in sequence in a one hour period. For all the keywords for which the Base-Kad client returned at least one index node, we measured the time each client took to locate the first index node.

In all, the Base-Kad client returned at least one index node for 567 searches. The fraction of these searches for

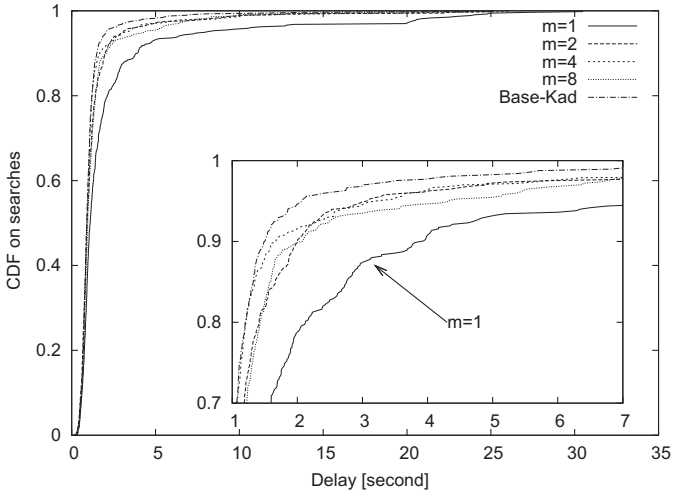


Fig. 5. Source-throttling: impact of m on search delay. Measured in Kad.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

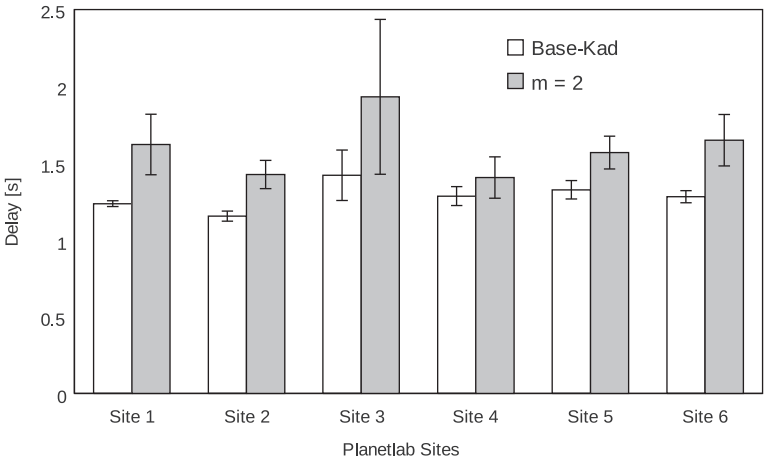


Fig. 6. Source-throttling: sensitivity to Planetlab Site. For each site, each bar is the average over five runs of the 90th percentile of the delay of successful searches. The error bars show the standard deviation. Measured in Kad.

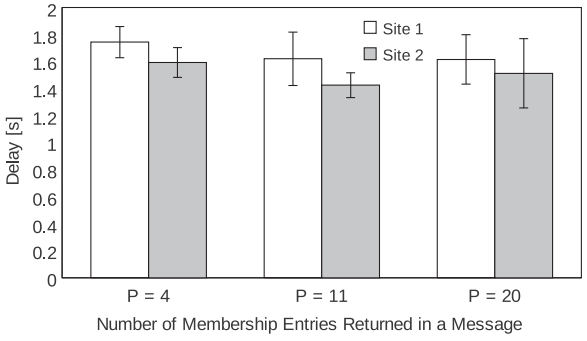


Fig. 7. Source-throttling: sensitivity to the number of membership entries returned in a reply message. For each site, each bar is the average over five runs of the 90th percentile of the delay of successful searches. The error bars show the standard deviation. Measured in Kad.

which the source-throttling scheme also returned one or more index node was 94.5% for $m = 1$, 98.8% for $m = 2$, 99.5% for $m = 4$, and 99.6% for $m = 8$. Thus, while the throttling scheme did impact some of the searches, the effect was minor for $m = 2$ and higher values.

Fig. 5 plots the CDF of the search delay for successful searches (i.e., the time taken for the first index node to be returned). The line on the top represents the Base-Kad scheme, and the line on the bottom represents the source-throttling scheme with m set at 1. The remaining lines are the source-throttling scheme with m set to 2, 4, and 8. These lines are very close to each other and practically indistinguishable. Overall, the results indicate that while extremely aggressive levels of throttling ($m = 1$) can result in noticeable degradation in application performance, the degradation is minimal for even slightly higher values of m, such as $m = 2$ or $m = 4$.

Sensitivity: The previous graph showed results for one experiment from one site. We now consider six distinct Planetlab sites and conduct five runs from each site. For all sites, we make two hosts join the Kad network, one host running Base-Kad and the other running source-throttling with $m = 2$. Fig. 6 shows the search delay for each Planetlab site. We show two bars per site, one for Base-Kad and the other for source-throttling with $m = 2$. For each run, the 90th percentile of the delay across successful searches is taken. Each bar is the average of the 90th percentile of the search delay over five runs. The error bars show the standard deviation. We observe that the results across sites are consistent with the results in Fig. 5.

Additionally, we study the sensitivity of the results to the number of membership entries that are returned in a reply (P) in Fig. 7. Here we only consider source-throttling with $m = 2$. Each group of bars correspond to a different value of P, with each bar corresponding to a different site. For each run, the 90th percentile of the delay across successful searches is taken. Each bar is the average of the 90th percentile of search delay over five runs. The error bars show the standard deviation. We notice that the performance of $m = 2$ is not sensitive to the change in the P setting. This further confirms the feasibility of using small m values in realistic settings. We repeated these experiments on additional Planetlab sites and results were similar.

7.1.2. Destination-throttling: impact of D and F

We next study the impact of the destination-throttling scheme on Kad performance. In doing so, the key metric is the extent to which destinations (prefixes, IPs, and ⟨IP,Port⟩s) are unnecessarily blocked due to benign validation failures. Note that in the destination-throttling scheme, only the validation failures in a recent window of time are considered in deciding whether a destination is to be blocked or not. Hence our evaluation focuses on understanding validation failures seen by typical clients over such a time window. We believe that a reasonable window length should be tens of minutes, and use one hour in our evaluation.[2]

[2] If the number of peers in the system is N, each peer could incur P validation failures before blocking a prefix, each validation packet is B bytes, and the validation failures are considered for a time T, then the expected validation traffic to a victim under a DDoS attack is $N * P * B/T$. For a population of 1 million, with $P = 20$, $B = 100$ bytes, and $T = 1$ h, this is about 4.4 Mbps, which we believe is reasonable.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

Table 2
Destination-throttling: percentage of contacted prefixes blocked with the particular search rate and for various combinations of (D_{prefix}, F_{prefix}). Measured in Kad.

# of searches	(D_{prefix}, F_{prefix}) (%)				
	(5,5)	(5,10)	(10,10)	(10,15)	(15,15)
100	0.07	0	0	0	0
300	0.55	0	0	0	0
1000	4.53	0.44	0.29	0.07	0.05

A key factor that impacts the results is the aggressiveness with which clients conduct searches, as this impacts the number of validations conducted and failed validations seen. Thus we have done a sensitivity study to the rate at which our clients conduct searches. We choose the rate to be 100, 300 and 1000 per hour. According to a study of client query patterns [39], even 100 per hour is higher than the search rate of a typical client. Three-hundred per hour is comparable to the most aggressive clients, while 1000 per hour is significantly beyond even most aggressive clients and stresses our scheme. Adopting a similar methodology as in Section 7.1.1, we instrument multiple versions of a client to join the live Kad network in parallel, each conducting an appropriate number of keyword searches over a one hour period.

Table 2 shows the percentage of prefixes blocked for a client conducting a particular number of searches in one hour. Results are shown for various combinations of D_{prefix} and F_{prefix}. A prefix is blocked if it has seen more than F_{prefix} failures involving D_{prefix} distinct IP addresses. To determine the prefix to which a client belongs, we use the Route Views dataset that helps map IP addresses to prefixes based on BGP data [34]. If any prefix is coarser than a /16 however, we simply consider the /16 as the prefix. The results show that barring extremely small values of the D_{prefix} and F_{prefix} parameters, the number of blocked prefixes is small. In particular, for 300 searches per hour, no prefixes are blocked if D_{prefix} is 5, and F_{prefix} is 10 or larger. Even with a search rate as high as 1000 per hour, the percentage of falsely blocked prefixes is less than 0.3% for a D_{prefix} of 10 or larger, and a F_{prefix} of 10 or larger. Overall, choosing F_{prefix} values in the 10–15 range, and D_{prefix} in the 5–10 range is effective.

We have conducted a similar evaluation to study the impact of the F_{ip}, D_{ip}, and F_{ipport} parameters. Our results show that with D_{ip} value of 3 or larger, and F_{ip} and F_{ipport} values of 5 or larger the scheme works well. With these settings, no destinations are blocked at the IP level even when the client conducts 1000 searches per hour. Further, as the client search rate varies from 100 to 1000 per hour, only 0.2–0.4% of the ⟨IP, Port⟩s contacted are blocked. Interestingly, most of the blocked destinations corresponded to port 53, which is used for DNS service. We believe this corresponds to a real attempt of DDoS attacks exploiting the wild Kad system, as indicated by Zhou et al. [20], and discussions on the eMule forum [19].

Sensitivity: We conducted sensitivity experiments of the results in Table 2 for *300* searches per hour and two (D_{prefix}, F_{prefix}) combinations of parameters, on six Planetlab sites. Fig. 8 presents our results. There are two bars per site, each for a different combination of (D_{prefix}, F_{prefix}). Each bar shows the average over five runs of the percentage of contacted prefixes that were blocked. Error bars show the standard deviation. We can see that the results are consistent across sites with the results in Table 2. In particular, notice that for $(D_{prefix}, F_{prefix}) = (10,10)$, five out of the six sites did not block any prefixes for the experiments conducted.

7.2. Video broadcasting application

Our ESM parameterization experiments were conducted on Planetlab. We leverage a trace from a real deployment [22] to emulate group dynamics and resource constraints in the system. We only show results for the source-throttling scheme. We were unable to parameterize the destination-throttling scheme since it requires realistic distributions of participants sharing a prefix or an IP address and this information was not available in the trace. Given this, we use the results from the Kad experiments to select the *F* and *D* parameters for ESM.

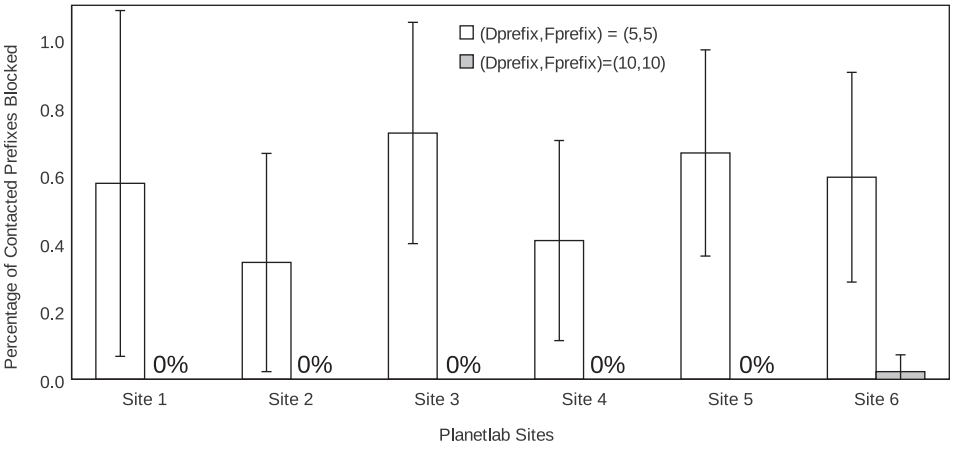


Fig. 8. Destination-throttling: sensitivity to Planetlab Site. For each site, each bar is the average over five runs of the percentage of contacted prefixes that are blocked. The error bars show the standard deviation. Note that for all but one site, when $(D_{prefix}, F_{prefix}) = (10,10)$, 0% of prefixes are blocked. Measured in Kad.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

The Planetlab experiments emulate a 20 min segment of the trace, with 148 joins, 173 leaves and 377 nodes in total. The streaming video rate employed is 450 Kbps, which represents typical media streaming rates in real settings [22]. In addition, we vary the fraction of NAT nodes in the system and turn off the NAT-aware heuristics, to increase the likelihood of benign failures and stress our scheme.

We observed that the average streaming rate received by nodes throughout the experiment is not affected by small values of m. More than 94% of the nodes received more than 90% of the streaming rate for $m = 1$, $m = 2$ and $m = 4$. To explain these results, consider that a key factor that may affect the performance of ESM is the number of entries the nodes have in their routing table. Having many entries ensures that nodes have enough potential parents to contact when they join the group or when a parent leaves. The source-throttling scheme can impact ESM by reducing the rate at which nodes learn about others, since membership information received may not be fully utilized. But this does not affect nodes in the long run, since over time they are still able to build a large routing table. Hence, the value of m does not affect the average streaming rate received by nodes.

To explore the potential impact of the source-throttling scheme on the performance of nodes in the initial phase, we consider the time it takes for a node to join the multicast tree. Fig. 9 shows the 90 percentile of the join time for various values of m and different NAT percentages. There is one bar for each value of m. Each bar is the average over five runs. Error bars show the standard deviation. The results show that while there is some increase in join time for small values of m, the impact is limited. For instance, the join time for $m = 2$ and $m = 4$ is only 2 s higher than Base-ESM, for settings with 65% NATs. Note that with NAT-aware heuristics in place, this difference would be even smaller.

7.3. Discussion

Our results indicate that the performance degradation with Kad, and ESM is minimal, even with small values of the throttling parameters. Since the amplification and magnitudes of potential DDoS attacks are directly determined by these parameters, these results indicate the promise of the validation framework in effectively controlling DDoS attacks without impacting performance of these applications.

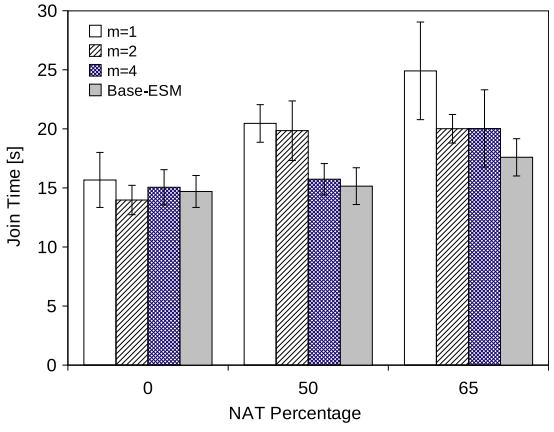


Fig. 9. Source-throttling: impact of m on the join time of nodes. Each bar is the average over five runs of the 90th percentile of the join time. The error bars show the standard deviation. Measured in ESM.

While the validation framework itself may be integrated easily in many P2P systems, the appropriate parameter choices are potentially dependent on the particular application. We now discuss the factors that might impact the parameter choice, and why we expect the parameters can potentially be kept small in general.

The primary concern with the source-throttling scheme is that when a membership message is received, benign validation failures incurred to some of the membership entries could prevent other potentially useful entries in that message from being validated (and hence utilized). To get more insight into this, consider that each membership message includes K entries, and assume the probability a probe will fail for benign reasons is p. With the source-throttling scheme, it may be easily verified that the expected number of membership entries that are probed is $\frac{m}{p}$, and the number of potentially useful entries probed is $\frac{m}{p} - m$. Since the expected number of potentially useful entries included in the entire membership message is $(1 - p)K$, the fraction of potentially useful entries that our source-throttling scheme actually utilizes is $\frac{m}{pK}$. If the probability p of benign validation failures is kept low, for example by including NAT-aware heuristics, and by ensuring the system does a good job of minimizing stale membership information, then, small m values will have only minor performance impact. Even if the application cannot eliminate benign validation failures, it may have other mechanisms that can help limit the performance impact. For example, when the Kad system receives a membership message in response to a search request, it preferentially probes entries that are closer to the search target. This ensures that the most potentially useful entries are utilized first, even if some of the entries are not utilized.

In our work, we have assumed the parameters are set uniformly across all clients and in static fashion. One could envision the need for the parameters to be set dependent on the client characteristics (e.g. aggressiveness of search patterns), or with differing levels of thresholds for different destination networks, based on their number of participants or bandwidth capabilities. In our evaluations, we have found that setting parameters conservatively based on worst-case scenarios (e.g. based on extremely aggressive search patterns) is sufficient to keep parameters low. That said, there may be potential benefits in other applications and deployment scenarios to tuning the parameters to individual clients or prefixes. Self-tuning mechanisms to dynamically determine appropriate parameters for any application, and deployment scenario are an interesting direction of future work.

8. Evaluation under attack

In this section, we present results evaluating the effectiveness of our validation framework in minimizing DDoS

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

attacks while achieving good performance even under attack. We implemented the validation framework on a Kad client and an ESM client. We refer to our modified clients as *Resilient-Kad* and *Resilient-ESM*. Again, we refer to the unmodified Kad and ESM clients as *Base-Kad* and *Base-ESM*. We set various throttling parameters according to the results obtained in Section 7. In particular, we set $m = 2$, $D_{prefix} = 10$, $F_{prefix} = 10$, $D_{ip} = 3$, $F_{ip} = 5$ and $F_{ipport} = 5$ for both *Resilient-Kad* and *Resilient-ESM*. All the experiments in this section were conducted on Planetlab.

8.1. File sharing application

In our Kad experiments, the inter-arrival patterns of nodes, and the stay time duration follow a Weibull distribution, based on [40]. A mean stay time of 10 min is assumed. Each experiment lasts 30 min, and involves 680 clients in total, with peak group size around 270. Each client conducts a search for a random logical identifier every 60 s, which is intended to simulate a search for a keyword or a file. There are five attackers that stay through the entire experiment and there is a single victim. Each attacker conducts the *Search hijack* attack from Section 2.2, and employs the *Attraction* [6] and *Multifake* (Section 3.1) heuristics. In particular, when an attacker initially joins the network, it proactively pushes information about itself to about *100* innocent nodes, forcing them to add the attacker to their routing tables. This causes many nodes to send search queries to the attacker and be redirected to the victim. In addition, in every search response, the attacker includes the victim's contact information about *100* times. This causes even larger attack magnitudes since innocent nodes send several messages to the victim for every response from the attacker (see [6] for more details).

Fig. 10 shows the attack traffic generated at the victim as a function of time, from one experiment. With the *Base-Kad*, the traffic was as high as 10 Mbps throughout the run. Further, the traffic at each attacker was only about 250 Kbps, which while higher than what a normal user sees, is 40 times lower than the traffic seen by the victim. However, with *Resilient-Kad*, the attack magnitude was effectively reduced by a factor of 100,000 from 10 Mbps to 0.1 Kbps.

Fig. 11 compares the performance of the *Base-Kad* and the *Resilient-Kad* schemes. There are three sets of bars, each corresponding to a setting with a particular percentage of nodes behind NAT. Each set has four bars corresponding to the two schemes, with and without the presence of attackers. The graph shows the search delay, averaged over all searches conducted by all nodes throughout a run, then averaged over five runs. Error bars show the standard deviation. Here search delay is measured as the time taken to locate the node which is not behind NAT and which has an ID that is the closest to the target ID. Our goal is to emulate location of index nodes in the real Kad network, and we note that only public nodes can be index nodes. We make the following observations.

First, in the presence of attackers, the search delay with the Base-Kad degraded significantly (over eight seconds in all NAT percentage settings). This was because nodes kept getting redirected by the attackers to the victim when they conducted searches. However, with Resilient-Kad, in the presence of the attackers the degradation was small and the search delay was still under 3 s for all NAT percentage settings. This was because fake information provided about the victim was quickly throttled and not used as part of the searches.

Second, in the absence of attackers, Resilient-Kad performed slightly better than Base-Kad. There are mainly two reasons for this. First, Base-Kad does not involve NAT-aware membership management. Thus, its routing table could include nodes behind NAT, and many search messages could fail since nodes behind NAT are contacted, leading to longer search times. In contrast, Resilient-Kad contains NAT-aware membership management leading to a routing table with fewer useless entries. Second, the destination-throttling scheme implemented in Resilient-Kad has the side effect that it can help purge stale membership information. In particular, consider a scenario where node

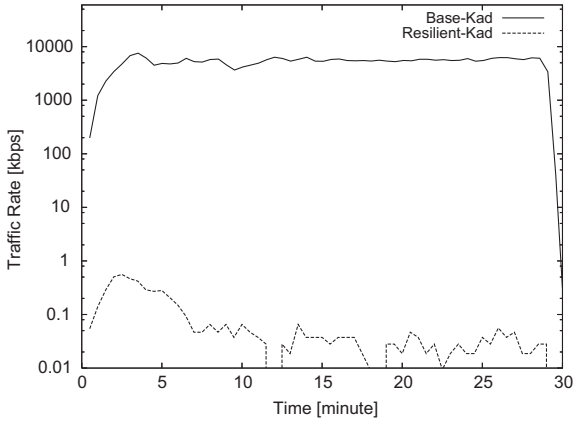


Fig. 10. Traffic seen at the victim as a function of time, with five attackers, and 50% of the nodes behind NAT. Measured in Kad.

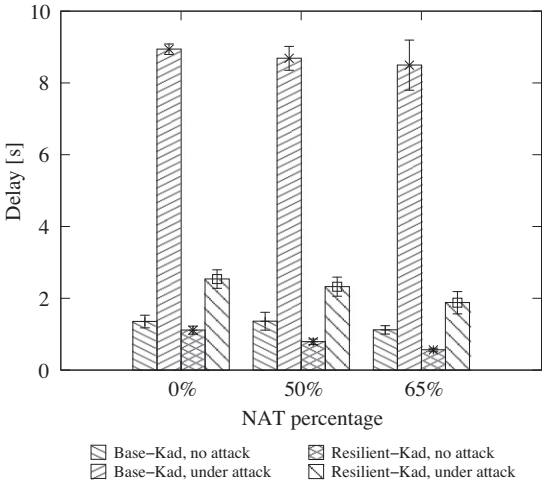


Fig. 11. Average time a search takes to locate the index node, with different NAT percentages. Each bar is the average over five runs. The error bars show the standard deviation. Measured in Kad.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

A has left the group. Node B learns stale membership information about node A from some other node. In Base-Kad this membership information is accepted, adding to useless entries in B's table for some time. Further, the stale information may also be propagated by *B* to others. In contrast, with Resilient-Kad this membership information is not accepted due to the validation process, consequently helping avoid useless entries.

Finally, the performance of all schemes get better when the NAT percentage increases. This is because, as the number of public nodes decreases, there are fewer hops to reach the index node of a given target ID. Resilient-Kad nodes see better improvements since they only maintain entries of public nodes while Base-Kad nodes maintain useless entries of peers behind NAT.

8.2. Video broadcasting application

We consider the effectiveness of *Resilient-ESM*. We assume 10% of the nodes are malicious and perform an attack as described in Section 2.2 and in [6]. Our results indicate that the *Resilient-ESM* scheme is effective in containing attacks, reducing the attack magnitude from 10 Mbps to 0.1 Kbps.

Next, we consider application performance with *Base-ESM* and *Resilient-ESM*. Fig. 12 shows the fraction of nodes that see more than 90% of the source rate, for both schemes, with and without attackers. Each bar is the average over five runs. Error bars show the standard deviation. We observe that *Base-ESM* shows significant degradation in performance in the presence of attackers. For instance, less than 70% of the nodes received more than 90% of the source rate, in settings with 65% NATs. This is because under attack, much of the membership information in the routing table of nodes is fake. This reduces the number of potential parents to contact when nodes join the group or when a parent leaves. Furthermore, we notice that the performance degradation on *Base-ESM* becomes more significant as the fraction of nodes behind NAT increases. This is because in these regimes there are fewer potential parents in the system, so the impact of having fake entries in the routing table of nodes is even higher. In contrast, with *Resilient-ESM*, invalid membership information was not used, leading to fast convergence and high performance. In particular, 95% of the nodes received more than 90% of the source rate, for all NAT percentages, with and without attackers.

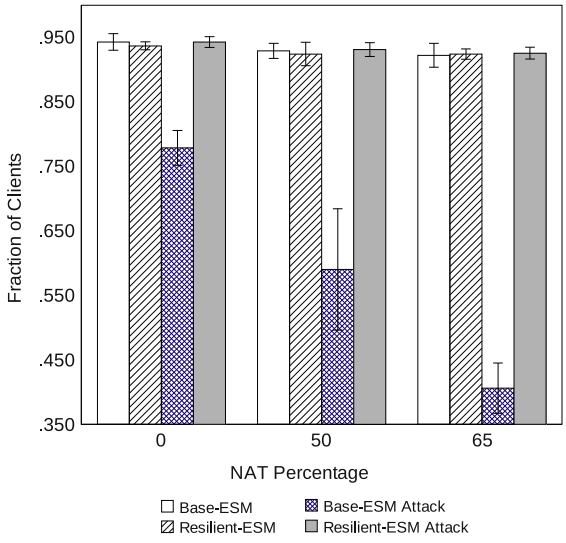


Fig. 12. Fraction of nodes that see more than 90% of the source rate, with 10% of the nodes as attackers and different NAT percentages. Each bar is the average over five runs. The error bars show the standard deviation. Measured in ESM.

9. Interactions with P2P developers

We have initiated discussions with P2P developers alerting them to the potential for DDoS attacks exploiting their systems, and encouraging them to make changes to their system to address the vulnerabilities. When we contacted the developers of eMule, they indicated they were already aware of our workshop paper [6], which had shown the feasibility of exploiting Kad (part of the eMule software) to cause DDoS attacks, and had implemented a number of changes to address the vulnerabilities. We identified some limitations of the changes, and the developers indicated they would implement additional mechanisms to address these in a future release. The combined set of changes limit the total number of entries in search response messages, and limit each response to have only one IP associated with an ID, and have at most 10 IPs for each /24 prefix. Similar restrictions are placed on the routing table entries. These changes are primarily intended to defend against attack heuristics such as *Multifake*, in order to bound attack amplification. These fixes are easily implementable and help solve some of the immediate problems. However, the fixes still suffer from a few limitations: (i) the amplification on /24 prefixes could be as high as 10; (ii) attacks on prefixes coarser than /24 are not prevented and the amplification of attacks on such coarser prefixes is not bounded; and (iii) each innocent participant could send an unbounded number of packets to the victim. We are in ongoing discussions with the developers to get our throttling mechanisms integrated, which could address these limitations. Our overall interactions show that P2P developers recognize the potential for DDoS attacks exploiting their systems, and the value of designing systematic solutions to counter the threat.

10. Related work

In Section 2.2, we have described most of previous research, which focus on exploiting individual P2P systems for DDoS attacks. In contrast, we have classified known DDoS attacks by amplification source, and identified principles to prevent amplification. In addition, ours is the first work aimed at enhancing the resilience of P2P systems to DDoS attacks.

The idea of employing probing-based mechanisms to validate membership information is briefly discussed in [2]. Athanasopoulos et al. [3] discusses a solution specific to Gnutella which requires a node to complete a "hand-

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

shake" with a peer prior to any file request message being sent to the peer, which could be viewed as a form of validation. In Kad, nodes employ a form of probing-based validation mechanism when nodes learn about members in search response messages (but nodes do not validate members learnt through publish messages). However, none of these consider the possibility that validation packets could be exploited to cause DDoS attacks. In fact, this was exploited in [6] to achieve large attack amplification. Further, we have also considered issues such as benign validation failures, DDoS attacks on entire network prefixes, and disconnection attacks. In addition, we have presented analysis and comprehensive performance evaluations of our framework. Finally, we have shown that our mechanisms are general and can defend against attacks on a diverse range of systems. Freedman et al. [41] have used probing-based mechanisms to validate membership information to improve DHT lookup performance. In contrast, our focus is on DDoS detection and the interoperability of validation mechanisms with sources of benign failures such as NATs.

In our earlier paper [31], we showed the limitations of three other techniques in enhancing the resilience of P2P systems. A first technique was to limit DDoS attacks by using pull-based membership management rather than push. However, while this reduces the susceptibility to attacks in some contexts, pull-based protocols are still vulnerable as our attacks on Kad show [6]. A second technique was to corroborate membership information from multiple sources. However, this technique is highly susceptible to Sybil attacks, and can incur significant performance degradation. A third technique is to limit the number of IDs associated with the same IP address, and the number of IPs sharing the same prefix that a node accepts. The limits are applied even if only genuine participants are involved, greatly limiting communication between participants. In contrast, this paper takes a different approach that requires nodes to be validated, and bounds the number of validation failures to each IP address or prefix, which in turn results in much fewer false positives. The focus on validations, throttling schemes, and analysis of their performance and security distinguishes our current work.

Researchers (e.g. [42]) have looked at exploiting unstructured file systems to launch DDoS attacks on P2P systems by introducing unnecessary queries, and having them flooded by the system. In contrast to these attacks, our focus is on DDoS attacks on external servers, caused by introducing fake membership management information.

Several works [11–15] focus on how malicious nodes in a peer-to-peer system may disrupt the normal functioning, and performance of the overlay itself. Many of these works, and most notably [13,14], focus on attacks on structured DHT-based overlays, and rely on trusted authorities to assign node IDs to principals in a certified manner. Our work differs in several ways. First, we focus on DDoS attacks on the external Internet environment, i.e., on nodes not participating in the overlay. Second, we focus on mechanisms that may be easily integrated into both structured DHT-based and unstructured non DHT-based systems. Third, we do not rely on a centralized authority to certify membership information. We believe these considerations are important to meet the threats for many existing extensively deployed systems. However, it may be interesting to investigate whether stronger security guarantees can be provided by exploiting DHT properties and using centralized authorities.

Several works have looked at the design of Byzantine resilient gossip protocols in the traditional distributed systems community in order to validate information (for example, [27,30]). While there is much to learn from these efforts, 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. Our focus in this paper is on exploiting P2P systems to launch DDoS attacks. In contrast, other works have explored attacks caused by DNS and web-server reflectors, and misuse of web browsers and botnets [17,43,44]. A recent work [45] builds a DDoS attack model in the application layer, and proposes a defense mechanism against Layer-7 attacks by combining detection and currency technologies.

11. Conclusions

In this paper, we have made two contributions:

- First, we have shown that the feasibility of exploiting P2P systems to launch high-amplification DDoS attacks on web and Internet servers stems from a violation of three key principles essential for robust P2P design. These principles are: (i) membership information must be validated before use; (ii) innocent participants must only propagate validated information; and (iii) the system must protect against multiple references to the victim. While these principles are almost obvious in retrospect, the failure to follow the guidelines in a wide range of deployed systems, and the resulting repercussions are striking.
- Second, we have shown the effectiveness of an active probing approach to validating membership information in thwarting such DDoS attacks. We have focused on such an approach given that it does not rely on centralized authorities for membership verification, and is applicable to both structured and unstructured P2P systems. Despite the simplicity of the approach, it can keep attack amplification low (to a factor of 2), while having a modest impact on performance. For a video broadcast application with stringent performance requirements, and for $m = 2$, the average source rate seen by nodes is practically unaffected, and when the 90 percentile of client join time is considered, the increase is less than 12%. With Kad and for $m = 2$, the search time increases by less than 0.3 s on average.

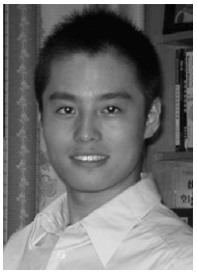
While we have taken a key step towards enhancing the resilience of peer-to-peer systems to DDoS attacks, we are extending our work in several directions. From a security perspective, we are investigating mechanisms that can bound amplification when DDoS attacks are conducted

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

on nodes actually participating in the system. From a performance stand-point, we are investigating self-tuning mechanisms to dynamically determine appropriate parameter choices for any application and deployment scenario.

References

[1] S. Bellovin, Security Aspects of Napster and Gnutella, invited talk at USENIX Annual Technical Conference, 2001.
[2] N. Naoumov, K. Ross, Exploiting P2P systems for DDoS attacks, in: International Workshop on Peer-to-Peer Information Management, 2006.
[3] E. Athanasopoulos, K.G. Anagnostakis, E. Markatos, Misusing unstructured P2P systems to perform DoS attacks: the network that never forgets, in: Proceedings of the ACNS, 2006.
[4] K.E. Defrawy, M. Gjoka, A. Markopoulou, BotTorrent: misusing BitTorrent to launch DDoS attacks, in: Proceedings of the Usenix SRUTI, 2007.
[5] K.C. Sia, DDoS Vulnerability Analysis of BitTorrent Protocol, Technical Report, UCLA, 2007.
[6] X. Sun, R. Torres, S. Rao, DDoS attacks by subverting membership management in P2P systems, in: Proceedings of the NPSec, 2007.
[7] Prolexic, <http://www.prolexic.com/content/moduleId/tPjJLKRF/article/aRQNVcBH.html>.
[8] Windows peer-to-peer networking, <http://www.microsoft.com/p2p>.
[9] S. Ali, A. Mathur, H. Zhang, Measurement of commercial peer-to-peer live video streaming, in: Proceedings of the WRAIPS, 2006.
[10] X. Hei, C. Liang, J. Liang, Y. Liu, K. Ross, Insights into PPLive: a measurement study of a large-scale P2P IPTV system, in: Proceedings of the Workshop on Internet Protocol TV (IPTV) Services Over World Wide Web in conjunction with WWW2006, 2006.
[11] E. Sit, R.Morris, Security considerations for peer-to-peer distributed hash tables, in: Proceedings of the IPTPS, 2002.
[12] J. Douceur, The Sybil attack, in: Proceedings of the IPTPS, 2002.
[13] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D.S.Wallach, Security for structured peer-to-peer overlay networks, in: Proceeding of the OSDI, 2002.
[14] A. Singh, T.-W. Ngan, D.P.D. Wallach, Eclipse attacks on overlays: threats and defenses, in: Proceedings of the INFOCOM, 2006.
[15] D. Wallach, A Survey of peer-to-peer security issues, in: International Symposium on Software Security, 2002.
[16] J. Yu, Z. Li, X. Chen, Misusing Kademlia protocol to perform DDoS attacks, in: Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2008.
[17] V. Paxson, An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks, 2001.
[18] D.D.J. Mirkovic, S. Dietrich, P. Reiher, Internet Denial of Service: Attack and Defense Mechanisms.
[19] eMule Online Community, <http://forum.emule-project.net/index.php?showtopic=133799>.
[20] M. Zhou, Y. Dai, X. Li, A measurement study of the structured overlay network in P2P file-sharing applications, in: Proceedings of IEEE ISM, 2006.
[21] EMule, <http://www.emule-project.net>.
[22] Y. Chu, A. Ganjam, T.S.E. Ng, S.G. Rao, K. Sripanidkulchai, J. Zhan, H. Zhang, Early experience with an internet broadcast system based on overlay multicast, in: Proceedings of the USENIX, 2004.
[23] J. Harrington, C. Kuwanoe, C. Zou, A BitTorrent-driven distributed denial-of-service attack, in: Proceedings of the 3rd International Conference of Security and Privacy in Communication Networks (SecureComm), 2007.
[24] eDonkey, <http://en.wikipedia.org/wiki/EDonkey2000>.
[25] P. Maymounkov, D. Mazieres, Kademlia: a peer-to-peer information system based on the XOR metric, in: Proceedings of the IPTPS, 2002.
[26] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, in: Proceedings of the ACM SIGCOMM, 2001.
[27] D. Malkhi, Y. Mansour, M. Reiter, Diffusing without false rumors: on propagating updates in a byzantine environment, Theoretical Computer Science 299 (1–3) (2003) 289–306.
[28] D. Malkhi, E. Pavlov, Y. Sella, Optimal unconditional information diffusion, in: 15th International Symposium on DIStributed Computing (DISC 2001), 2001.
[29] D. Malkhi, M. Reiter, O. Rodeh, Y. Sella, Efficient update diffusion in byzantine environments, in: 20th Symposium on Reliable Distributed Systems (SRDS 2001), 2001.
[30] Y. Minsky, F. Schneider, Tolerating malicious gossip, Distributed Computing 16 (1) (2003) 49–68.
[31] X. Sun, R. Torres, S. Rao, On the feasibility of exploiting P2P systems to launch DDoS attacks, Journal of Peer-to-Peer Networking and Applications, vol. 3, Springer, New York, 2010.
[32] S. Kamvar, M. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in: Proceedings of the WWW, 2003.
[33] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Computing Survey 42 (1) (2010).
[34] Route Views Project, <http://www.routeviews.org>.
[35] B. Krishnamurthy, J. Wang, On network-aware clustering of web clients, in: Proceedings of the ACM SIGCOMM, 2000.
[36] S. Guha, P. Francis, Characterization and measurement of TCP traversal through NATs and firewalls, in: Proceedings of the ACM SIGCOMM IMC, 2005.
[37] B. Ford, P. Srisuresh, D. Kegel, Peer-to-peer communication across network address translators, in: Proceedings of the USENIX, 2005.
[38] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, Simple Traversal of User Datagram Protocol Through Network Address Translation (STUN), RFC-3489, 2003.
[39] A. Klemm, C. Lindemann, M.K. Vernon, O.P. Waldhorst, Characterizing the query behavior in peer-to-peer file sharing systems, in: Proceedings of the ACM SIGCOMM IMC, 2004.
[40] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the ACM SIGCOMM IMC, 2006.
[41] M.J. Freedman, K. Lakshminarayanan, S. Rhea, I. Stoica, Non-transitive connectivity and DHTs, in: Proceedings of the WORLDS, 2005.
[42] Y. Liu, X. Liu, C. Wang, L. Xiao, Defending P2Ps from overlay flooding-based DDoS, in: Proceedings of the International Conference on Parallel Processing, 2007.
[43] V. Lam, S. Antonatos, P. Akritidis, K.G. Anagnostakis, Puppetnets: misusing web browsers as a distributed attack infrastructure, in: Proceedings of ACM CCS, 2006.
[44] DNS Amplification Attacks, <http://www.isotf.org/news/DNS-Amplification-Attacks.pdf>.
[45] J. Yu, Z. Li, H. Chen, X. Chen, A detection and offense mechanism to defend against application layer DDoS attacks, in: Proceedings of the Third International Conference on Networking and Services (ICNS), 2007.



Xin Sun is a PhD student in the School of Electrial and Computer Engineering at Purdue University. He works with Prof. Sanjay Rao. He received his B.E. degree from University of Science and Technology of China in 2005. His research interests are in overlay networks with an emphasis on security, and enterprise network management.

Ruben Torres is a Ph.D. student in the School of Electrical and Computer Engineering at Purdue University and a research assistant in the Internet Systems Lab. Ruben received his MS in electrical and computer engineering from Purdue University in 2006 and his BS in electronics and communications engineering from the University of Panama in 2002. His research interests include distributed systems and security, with an emphasis on peer-to-peer networks.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021

Sanjay G. Rao received the Bachelor's degree in Computer Science and Engineering from the Indian Institute of Technology, Madras in 1997 and the Ph.D. from the School of Computer Science, Carnegie Mellon University in 2004.

He is an Assistant Professor in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, where he leads the Internet Systems Laboratory. He was a visiting researcher in the Network Measurement and Management group at AT&T Research, Florham Park, New Jersey in Summer 2006. He wrote the first paper to show the viability of an overlay approach to multicast, and led the design of an overlay broadcasting system based on a peer-to-peer architecture. His research interests are in Networking, and Distributed Systems. His current projects are in Peer-to-Peer systems, and Network Management. Prof. Rao has served on the Technical Program Committees of several workshops and conferences including ACM SIGCOMM, IEEE Infocom, and ACM CoNEXT.

Please cite this article in press as: X. Sun et al., Preventing DDoS attacks on internet servers exploiting P2P systems, Comput. Netw. (2010), doi:10.1016/j.comnet.2010.05.021