

Simulating Non-Scanning Worms on Peer-to-Peer Networks

Guanling Chen
University of Massachusetts Lowell
Lowell, MA
glchen@cs.uml.edu

Robert S. Gray
BAE Systems
Arlington, VA
robert.s.gray@baesystems.com

Abstract

Millions of Internet users are using large-scale peer-to-peer (P2P) networks to share content files today. Many other mission-critical applications, such as Internet telephony and Domain Name System (DNS), have also found P2P networks appealing due to their scalability and reliability properties. These P2P networks, however, could be leveraged by automatic-propagating Internet worms to quickly infect a large vulnerable population and inflict tremendous damages to information infrastructure and end systems.

While much work has been done to study random-scanning worms, such as CodeRed and Slammer, we have less understanding of non-scanning worms that are potentially stealthy. In this paper, we identify three strategies a non-scanning worm could use to propagate through P2P systems. To understand their behaviors, we provide a workload-driven simulation framework to characterize these worms and identify the parameters influencing their propagations. The non-scanning nature allows P2P worms to evade many of today's detection methods aimed at random-scanning worms. We propose and evaluate an online detection algorithm against these P2P worms using statistical detection of change-points in streaming sensor data.

1 Introduction

Millions of Internet users are using large-scale peer-to-peer (P2P) networks to share content files today. Listed on the website of Kazaa, a popular P2P software, there have been almost 39 millions downloads in total and more than 0.8 millions downloads in a single week (November 14, 2005). During the one-month period of November 2001, Staniford et al. observed 9 million distinct remote IP addresses engaged in successful Kazaa connections with hosts in a single university (about 5,800 Kazaa hosts) [41]. Using a fast and accurate crawler, Stutzbach et al. recently discovered more

than one million online Gnutella nodes in just a few minutes [44]. Due to the high scalability and reliability properties of the decentralized and self-organizing P2P networks, other mission-critical applications have also started to use peer-to-peer protocols. For example, Skype is a popular Voice over Internet application that use Gnutella-like protocols. There are also proposals to build Domain Name System (DNS) with P2P technologies [7].

The widely-deployed P2P systems used by end users, however, have strong security implications. First, the users may have downloaded files embedded with malicious code. Second, the P2P client software may contain vulnerabilities that could be exploited by attackers. In particular, P2P systems often has homogeneous client implementation. For example, a recent study found that more than 75% Gnutella clients run the same software (LimeWire) [44]. A single implementation weakness of a commonly used P2P client thus results in a large vulnerable population. This situation is attractive for adversaries to exploit the P2P networks using Internet worms, which can automatically propagate through the network using a single vulnerability without human intervention. The compromised P2P nodes may be used to capture end users' sensitive information or be used for further attacks, such as Distributed Denial of Service (DDoS) [26].

Internet worms have already demonstrated their capability to inflict large-scale disruption of and damage to information infrastructure. In 2001, CodeRed infected more than 35,900 hosts within 14 hours and cost about \$2.6 billion [26]. In 2003, Slammer infected more than 90 percent of vulnerable hosts in less than 10 minutes, precluding any human-based incident response [25]. These notorious worms employ a *random scanning* strategy to find the potential victims. Namely, they randomly select targets from the IP space. The selected addresses, however, may not be in use or may be used by honeypots [28, 31]. Thus it is possible to detect random-scanning worms by capturing their scans directly [41, 32]. Random-scanning worms may also

cause abnormal network activities, which could be passively observed for detection [42, 12, 54, 19].

On the other hand, we have less understanding of *non-scanning* worms, which do not actively probe randomly selected addresses. Instead, these worms propagate using legitimate network activities or network topology information [41]. The first known computer worm, the Morris worm, was non-scanning and infected 5 to 10 percent of the whole Internet in 1988 [10, 40]. Recently we saw the re-appearance of such non-scanning worms, now leveraging peer-to-peer (P2P) networks such as Kazaa [38], which provide an attractive platform for worm attacks because of their large user populations and homogeneous software clients [41]. The Symantec virus and worm repository returns about 458 instances when queried with the keyword “Kazaa” (as of November 14, 2005).

We identify three types of non-scanning worms that could leverage P2P networks: *passive* worms that hide themselves in malicious files and trick users into downloading and opening them; *reactive* worms that only propagate with legitimate network activities; and *proactive* worms that automatically connect to and infect known peers using topological information. Note that the reactive and proactive worms are analogous to *contagion* and *topological* worms [41]. Some researchers define the passive worms, which attach to files and propagate with user activations, as *viruses* [47]. We do not make such a distinction for the purpose of this paper.

Existing work on P2P worms has focused on proactive worms that propagate using network topologies [5, 52, 53], given the empirical evidence that the P2P topologies approximate power-law distributions [33, 37]. Studying worm propagation using the aggregated properties of P2P networks typically assumes a static topology, in which a node stores the addresses of all neighbors with which it had communicated. The lack of detailed peer interactions makes topology-driven models unsuitable to simulate worm propagation, for instance, if a node can only cache the last N communicating peers. It is also difficult, if not impossible, to use these models to study passive and reactive P2P worms.

P2P networks are complex systems and it may not be feasible to use an analytical approach to model worm propagations without making overly simplified assumptions. Instead we present a unified simulation framework, driven by a P2P file-sharing workload model, to study the non-scanning P2P worms. Unlike previous work, our approach models detailed peer-to-peer file-sharing interactions. Our model captures file requests and downloads, which lead to network activities and topologies; thus it can be used to study the propagations of all passive, reactive, and proactive worms. The purpose of our study is to understand non-scanning P2P

worms with different infection strategies and evaluate possible detection methods. Our contributions are three-fold:

- we provide a P2P worm simulation framework, driven by a realistic P2P workload model;
- we study and compare propagation behaviors of three non-scanning P2P worms and identify the impact of various parameters;
- and we describe an online change-point based algorithm to detect these worms and evaluate it using our framework.

In our simulation framework we need to model the immunization process during worm outbreak. Unfortunately there is no empirical data for us to derive such a model. Some researchers simply use a constant-rate immunization approach [50]. Instead we use a modified sigmoid function given the intuition that immunization will likely increase and decrease as worm infection does. We need to point out, however, that this model may not be completely realistic though it is more intuitive than a flat-rate model. It is known that P2P systems are dynamic, though again there is no empirical model on how nodes join and leave the network. To simplify our framework, we assume a network with a constant size in this paper. However we do explicitly model the file download failures and connection failures that may be caused by network dynamics.

The rest of our paper is organized as follows. We discuss the design of our simulation framework in Section 2. We present the three non-scanning P2P worms and the simulation results in Sections 3 to 5. In Section 6 we evaluate the change-point detection algorithm for these worms. Finally we discuss related work and conclude in Sections 7 and 8.

2 Simulation Design

In this paper, we focus on studying *unstructured file-sharing* P2P networks, such as Kazaa and BitTorrent. Most existing P2P worms target these kinds of P2P systems. Some P2P systems are not designed for file sharing, such as RON (to support reliable routing) [2] and Skype (to support voice over IP applications) [4]. Some P2P networks are structured, such as Pastry [34] and Chord [43], which could also be used to support file sharing but have not been widely deployed. These P2P systems may have quite different structural properties and workload traffic than unstructured file-sharing ones, and we plan to study them as future work.

2.1 P2P workload

The main purpose of existing P2P systems is to share files among users, who oftentimes use the system to exchange (legal or not) popular music, movie, and software files. The frequency of a file being downloaded reflect its popularity among users. Recently Gummadi et al. studied a 200-day trace of Kazaa P2P traffic collected at the University of Washington, and found that the file popularity follows the Zipf(α) distribution [14]. Namely, the frequency of these files being downloaded is proportional to $x^{-\alpha}$, where x is the rank of these files according to their download frequency. They also found that the workload in Kazaa, unlike Web traffic, was mostly driven by request-at-most-once downloads. Once a file has been downloaded, users tend not to download it again.

We construct a P2P workload model using some of their findings. We denote N as the number of nodes in a P2P network and S as the number of unique files shared by all P2P users. The popularity of these files follow the Zipf(α) distribution. Once a node has requested a file, it will never download it again. Instead, it will request a file from among those it has not downloaded, according to a re-normalized Zipf distribution (so the probabilities of requesting those files still sum to be one).

New files are constantly introduced into P2P systems. We use A to denote the rate of new files being made available, in addition to the initial S files. The rank of a new file is again determined by the Zipf distribution. The original file at that rank will be pushed one slot down, and the Zipf distribution is re-normalized.

P2P systems are highly dynamic as nodes join/depart from the network [35]. Here the node join/depart could simply be the user opening/closing their P2P client software. A download from a remote peer may fail due to various reasons. For instance, the requested file has been removed, the network has failed, or the remote node has left during the downloading process. To increase reliability and speed up the download, like BitTorrent does, a file may be downloaded simultaneously from multiple peers who have that file, which are called *seeds*. We capture the dynamic factors by modeling the probability of a successful download. The probability P increases, but never exceeds one, with the number of seeds being used. We take a modified sigmoid function to model success probability, which has two parameters that we can control how fast it approaches the limit (one):

$$P_x = \frac{\gamma}{1 + e^{2-\beta x}}. \quad (1)$$

Here x is the number of seeds being used, and we adjust γ and β in simulations to study their impact (how fast approaching one). To limit resource usage, some-

times there is a limit on the maximum number of seeds from which a file can be simultaneously downloaded. We call this the *seeding limit* and denote it D .

We assume that the popularity of the nodes also follows a Zipf distribution. For instance, a better-connected node tends to be used more often than others. To choose the seeds from all available peers that can serve the file being requested, we choose (at most) D seeds according to their popularity distribution. Existing studies show that nodes in P2P systems evolve into power-law topologies [33, 37]. Namely, the connectivity degree of P2P nodes approximates a power-law distribution, which has a direct connection to Zipf [1]. Validation tests show that our workload model indeed generates power-law like topologies, using the simulation framework presented in Section 2.3.

2.2 Worm propagation

We consider three types of non-scanning worms that may propagate on P2P networks. Passive worms disguise themselves in files with enticing names to trick users into downloading and opening these malicious files. Reactive worms ride along with legitimate network activities, instead of attaching to files. Passive and reactive worms tend to spread relatively slowly, but they generate fewer network anomalies than random scanning worms. Proactive worms, however, achieve much faster propagation speed by directly connecting to other peers, whose addresses were cached after previous file requests and downloads. All of these worms can be released by the attacker on multiple nodes, called a *hit list* [41], to bootstrap infections. We use H to denote the size of the hit list.

Since passive and reactive worms propagate rather slowly due to relatively infrequent file requests, as we show in Section 3 and 4, it is likely for these worms to be discovered before reaching their full potential. For instance, the unexpected behavior after opening downloaded files may cause suspicion, and the worm spread onto a honeypot node could be easily analyzed [32]. The Benjamin worm, for example, was discovered and its signatures and vaccine were developed only one day after its initial release [38]. We thus need to model the user reaction in terms of applying immunization procedures to remove the worm or prevent infection.

Security patches, however, are generally applied slowly, either due to users' ignorance of worm threats, or concerns of the patches' compatibility with their existing systems. We decided not to use a constant immunization rate, since it does not capture various user reactions during different worm outbreak stages, as pointed out in [50]. Unfortunately we do not have empirical evidence of the immunization probability density function.

Our intuition is that the probability approaches but never exceeds one, as more nodes are infected that may bring more awareness to the general public. Again we take the sigmoid function (Equation 1) to determine the immunization probability. Here x is the ratio of the infected nodes to all nodes, also called the worm *prevalence*. We adjust γ and β in simulations to study the impact of immunization on worm propagation. To model the initial delay and user ignorance, the node does not apply the immunization when the prevalence is smaller than 1%. A node can no longer be infected, whether already infected or not, once it has been immunized. This corresponds to a Susceptible-Infectious-Removal (SIR) model, commonly used when analyzing infectious disease [17].

2.3 Simulation framework

We implemented the previous P2P workload and various worm propagations in a simulation framework. The simulator first initializes various components, such as nodes and files. All the nodes are initialized to be vulnerable. During each simulation time unit, immunization is applied if necessary and A new files are added into the system. Each node will request a file, which it has not downloaded before, according to the Zipf popularity distribution. Assuming the files downloaded by a node are not deleted, a requested file may be downloaded from one or more of the peers that have downloaded it before. At most D seeds are selected according to node popularity, following a separate Zipf distribution. The probabilities of a successful download and a node being immunized are determined by Equation 1, with different parameter values. We trace the worm propagation by plotting the worm prevalence against the timeline in all the experiments.

We do not explicitly model the distribution of file sizes and we assume that a file download completes (successfully or not) in one time unit. This is not an unreasonable assumption since we can define the time unit as several hours or longer (Gummadi et al. suggest half an day in their study [14]). Namely, for passive worms, a malicious file is downloaded and opened in one time unit, and the requesting node becomes infected if the download succeeds.

In the next sections, we study how the three non-scanning worms propagate under different situations. For each of the experiments, we ran the simulation ten times and took the average for the plots. We summarize common simulation parameters in Table 1, and we set the default values of some parameters as suggested by [14]. We present worm-specific parameters in their respective sections. All simulations use the default values except for the parameter being varied by individual

param	default	note
N	1,000	number of P2P nodes
S	40,000	number of initial files
A	10	new file arrival rate
D	1	seeding limit
α	1.0	param of Zipf distribution
β_1, γ_1	0.5, 1.0	params of download success prob.
H	1	size of hit list
β_2, γ_2	5, 0.1	params of immunization prob.

Table 1. Common simulation parameters.

experiment.

3 Passive Worms

In this section we study the passive P2P worms, which attach themselves to shared files and propagate as these files are downloaded and opened on other nodes. Such passive P2P worms have already emerged in reality. Targeting Kazaa networks, the Benjamin worm creates a local directory and changes the Kazaa settings to share this directory with remote nodes [38]. Then it makes copies of itself in that directory under many different names, such as popular music, movie, and software titles. Once these files are downloaded by other nodes and are opened, the previous process repeats and the worm propagates.

We assume that the worm always creates and shares the same K malicious files on the local host. To increase the chance of being downloaded, thus speeding up the propagation, these malicious files tend to have popular names, such as the titles of hit songs and movies. We assume that the worm author is able to use K names out of the top M most popular files. During a simulation time unit, each node requests a single file according to the file popularity distribution (Zipf). If the node has not been infected and immunized, the requested file is malicious (out of the K), the download is successful, then the status of that node becomes infected. In our simulations, the default values of K and M are 100 and 1000, respectively.

3.1 Simulation results

Intuitively, the more malicious files a worm can generate (large K) and the more popular their file names are (small M), the more likely other users will download these files and become infected. Figure 1 confirms that the worm spreads faster with larger K and reaches higher peak prevalence. The same observation holds as these files use names in a smaller top popularity range,

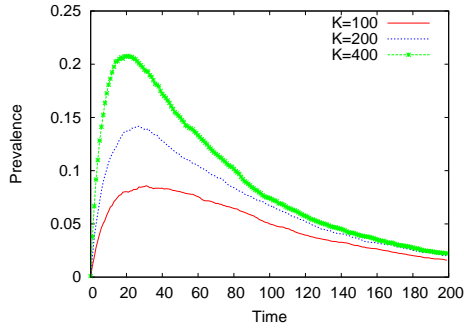


Figure 1. [Passive] Number of malicious files.

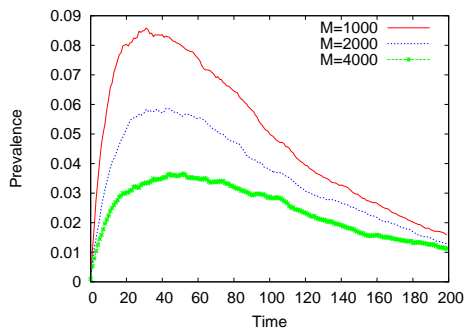


Figure 2. [Passive] Popularity of malicious files.

as shown in Figure 2. As a reality check, the Sanker worm that creates about 20 malicious files in Kazaa's shared directory had less than 50 infection reports by Symantec. On the other hand, the Benjamin worm that creates and shares about 2000 malicious files reportedly infected more than 1000 nodes.

To improve the worm effectiveness, the attacker could write a worm that creates a large number of files on a victim's machine. This, however, will consume significant system resources and may alarm the users. The attacker also could try to guess the most popular files to facilitate worm spread. It is difficult, however, if not impossible, for the attacker to do this given different interests of a large user population. Even worse, the continuous arrival of new files further reduces the worm spread, as shown in Figure 3. With 100 new files arriving each time unit, assuming half a day per unit, the worm can infect at most 3% nodes after about 2 weeks. The reason for such a large impact of the new files is the decreased popularity of the worm-generated malicious files. This suggests that it is a bad strategy for the worm to use a static list of file names. To write a worm that can gen-

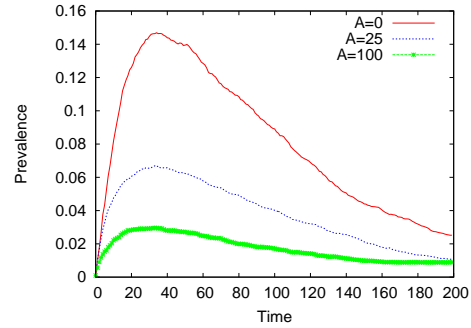


Figure 3. [Passive] New file arrival.

erate a dynamic list of names always having high popularity ranks, such as by passively monitoring download activity and learn the popularity, requires significant effort to put such intelligence into the worm.

We note that the worm prevalence increases to a certain peak and then starts to decline, due to the interplay of the file downloading and immunization processes. Recall that at any given time, a node is either vulnerable, infected, or immunized. The immunization process picks up as the worm spreads, and it reduces both the already infected nodes and the vulnerable populations. At some point the infection rate, determined by the probabilities of requesting a malicious file and successfully downloading it, starts to fall behind the immunization speed.

As expected, the immunization probability has a strong impact on worm spread. Figure 4 shows that the more likely a node to be immunized, the slower the worm spreads and the smaller the peak prevalence. Note that we adjusted the parameter γ_2 (Equation 1), and that the immunization probability increases as γ_2 increases. When there is no immunization process, the prevalence keeps growing until all nodes are infected. The infection rate decreases as both the vulnerable population and the popularity of malicious files decreases, leading to a slow worm propagation.

Another way for an attacker to facilitate worm spread is to use a large hit list. Figure 5 shows that this approach, however, only increases initial prevalence (compare with Figure 1; note the maximum prevalence is about the same). Note that the initial prevalence for $H = 200$ was slightly less than 20%, due to some overlap of randomly generated values. We see that a large hit list does not help passive worms much since it does not influence how a node requests files. On the other hand, the prevalence declines quicker with a large hit list as the immunization correlates with the prevalence (Section 2.2). A large hit list does boost the number of copies of malicious files and increases the probability of successfully downloading these files. In general, how-

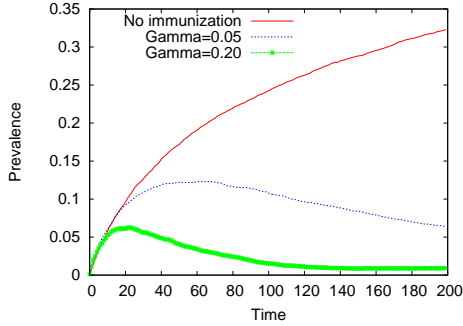


Figure 4. [Passive] Node immunization (γ_2).

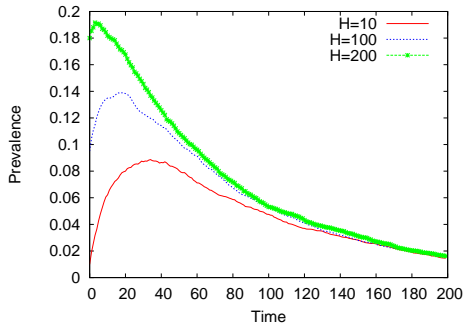


Figure 5. [Passive] Hit list.

ever, using a large hit list is not an effective strategy for passive worms.

Some other parameters of P2P systems also influence the spread of passive worms. For instance, D and β_1 (Table 1) change the probability of a successful file download. The larger the probability is, the faster the worm spreads and the higher the peak prevalence reaches. We omit the plots for these results, but all the curves have similar shapes and the prevalence reaches peaks between time 20 and 40.

4 Reactive Worms

Reactive worms, also called contagion worms, do not need to attach to files, but propagate with legitimate network activities to defeat common anomaly-based detection methods. A reactive worm on a victim host could learn the addresses of potential targets by simply waiting for other peers to request files from the victim or for the victim to request files from other peers. Triggered by the new connections, a reactive worm could then probe and infect other peers if they are vulnerable. Reactive worms typically require only one exploit of the dominant implementation of the P2P protocol, since the diversity of

P2P client software tends to be small. Yet the reactive worms still may have considerable propagation speed. For instance, Staniford et al. estimate, by analyzing an empirical trace collected at a single network location, that roughly 9 million Kazaa clients could be infected within one month [41]. Their estimate is conservative since they could not observe peer interactions not crossing their sensor. Our simulation framework, on the other hand, provides a holistic view of the worm propagation.

The simulation runs without worms for T time units to allow the files to distribute among the P2P network. Then the worm is randomly released on H nodes simultaneously. The file request and download operations continue as the worm spreads. We compare three worm infection directions: *source infection* if the worm can infect only the peers initiating the connections; *sink infection* if the worm can infect only the peers to which connections are made; and *mixed infection* if the worm can go in either direction. We assume that the infection always succeeds and is independent of whether the file download succeeds or not. The latter affects the file distribution in the system and thus indirectly influences the worm spread. Note that there is a subtle difference between source-infection reactive worms and passive worms. For passive worms, the host requesting a non-malicious file from another host containing malicious files will not be infected.

4.1 Simulation results

We found that the new file arrival rate has minimal influence on reactive worms, so we default it to zero in all experiments. We also found that T plays little role after it reaches a certain threshold, as the file distributions become relatively stable. So we use 100 time units as the default value of T , after which the worm is released.

We first examine the impact of worm infection direction, as shown in Figure 6. Source and sink worm propagations have a comparable spreading speed, while the source infection has a higher peak prevalence. This is a direct result of skewed distribution on the seeds from which to download files. The sink worm infection more likely will hit popular peers, while the source infection allows more nodes to be infected before immunization catches up. The mixed infection strategy, however, significantly boosts worm spread speed and prevalence. Within 10 time units, the worm infected almost 80% vulnerable population. Considering the large number of Kazaa users, this is much worse than the estimation made by Staniford et al. using observations at a single sensor [41]. The reason behind fast propagation of mixed worm infection is that the popular nodes are quickly infected by sink infections, and then all other nodes are infected by source infections when requesting

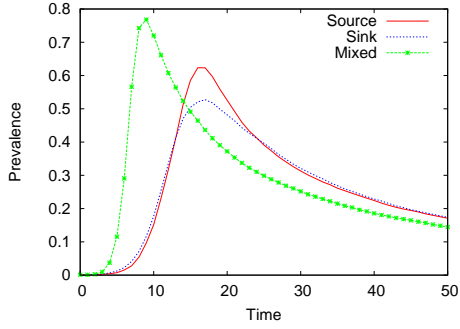


Figure 6. [Reactive] Infection direction.

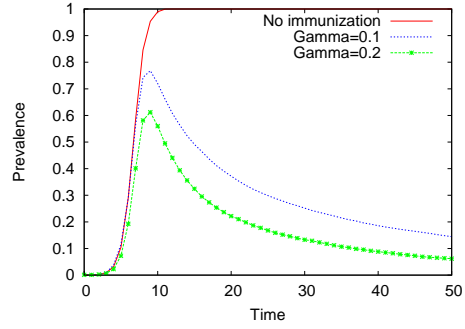


Figure 8. [Reactive] Node immunization (γ_2).

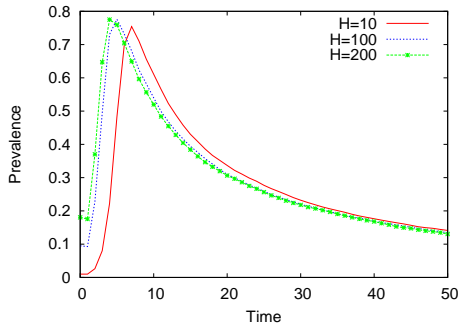


Figure 7. [Reactive] Hit list.

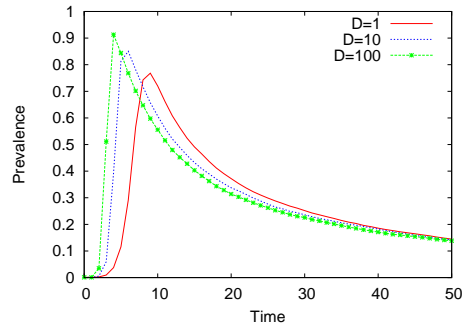


Figure 9. [Reactive] Seeding limit.

files from those popular peers. In all other experiments of this section, we use mixed infection as the worm’s default propagation strategy.

Figure 7 shows the impact of the hit list. A worm can spread faster with a larger hit list. The size of the hit list, however, does not significantly boost the peak prevalence. Recall that a large hit list for passive worms triggers quicker immunization and deters its prevalence (Figure 5). This is not the case for reactive worms; the prevalence continues to grow to a significant ratio because the immunization process could not catch up with the worm’s fast propagation speed. If there is no immunization applied, a reactive worm will quickly saturate the P2P network on time unit 12, as shown in Figure 8. As expected, a smaller probability for node immunization (larger γ_2) leads to faster worm spread and higher peak prevalence.

Unlike passive worms, the seeding limit has a non-trivial impact on the worm spread as shown in Figure 9. The more peers from which an infected node can download files, the faster the worm spreads with the sink infection strategy. In summary, reactive worms can penetrate a much higher percentage of nodes in a shorter time than passive worms. It generally can infect more than 70% nodes within 10 time units, while passive worms reach peak prevalence around 20% between time 20 to

40.

Note that the prevalence of both the passive and reactive worms drops quickly after it reaches certain peak value. This is due the correlated immunization with prevalence. If we use a constant immunization rate, the prevalence peak will reach higher and drop slower. Figure 10 shows the comparison of using these two different immunization process for both passive and reactive worms. We use default values for all parameters and the constant immunization rate is 1% in this case.

5 Proactive Worms

The addresses of the peers with which a node has communicated may be cached deliberately or unintentionally on that node. A proactive worm on an infected node then could simply choose these peers in the cache as targets and infect them. In our simulation, we model a FIFO cache with C entries on every node that keeps the addresses of the latest C peers with which the node had communicated. If we set C to be N , the cache is effectively unbounded. Given that a proactive worm voluntarily initiates connections to other peers, it may cause obvious network anomalies. To evade detection methods

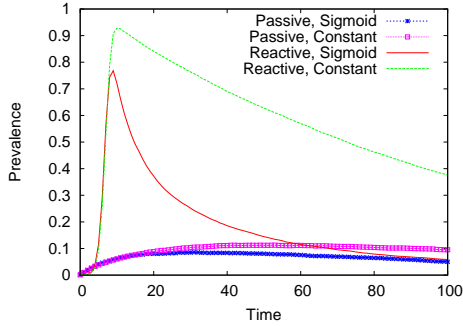


Figure 10. [Passive/Reactive] Comparison of constant immunization rate.

such as virus throttles [49], we assume that the worm only probes one peer selected from the cache every time unit. For instance, the time unit could be 1 second to evade 1 new connections per second detection threshold used by a virus throttle. The scale of time unit here typically is much smaller than the one used by passive and reactive worms.

Many P2P nodes are highly dynamic and they may not be available when probed by the worm [35]. We keep an *age* for each peer in the cache, which is defined as the time it has stayed in the cache. Note that the age is refreshed if the corresponding peer communicated with this node again. Intuitively, the likelihood for a peer to be available reduces as it ages in the cache. So we define the probability for a successful infection as the inverse of the peer’s age: $1.0/(K * age^\delta)$, where δ has default value of 2 and K is a normalization constant. Though we do not have empirical evidence the infection probability density function does follow this model, we could adjust δ to study its impact. We study three strategies for the worm to select peers from the cache: selection with increasing ages, selection with decreasing ages, or random selection (when the age information is not available). We assume the worm knows whether the infection succeeds or not, so it will avoid probing the nodes already infected by the probing worm instance.

We first let the simulation run with normal file operations for 100 time units to allow the caches to be filled. Then we release the worm on H nodes. Since the proactive worms propagate much faster than passive and reactive worms, on the order of seconds, we do not run file operations and immunization processes once the worm is released. The worm propagates using the existing state of all caches to select and probe one peer from the cache every time unit, which is conceivably much shorter (i.e. one second) than the time unit used by passive and reactive worms (i.e. half a day).

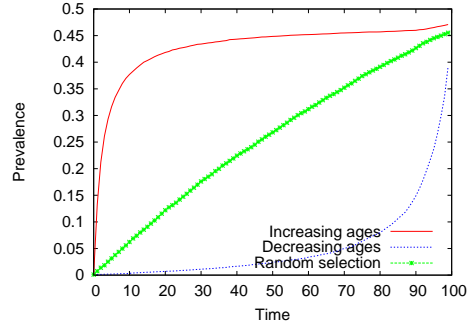


Figure 11. [Proactive] Peer selection.

5.1 Simulation results

In Figure 11, we show that how the worm selects cached peers has a great impact on its propagation (with C defaulting to 100). Probing the new entries in the cache first allows the worm to spread much faster at early stage than probing the old entries first, since the peers with small age are more likely to be available and thus have greater probability to be infected. On the other hand, probing old entries in the cache first allows the worm to become more effective at later stage when it comes to the head of the cache. If the worm has to resort to random selection, such as when the aging information is not available, it spreads with a rough linear speed between the other two methods. In the rest of the experiments, we assume the proactive worm employs a random selection strategy.

Intuitively the worm would spread faster with a larger cache size C . It turns out that C has minimal impact on worms who select cached peers with increasing ages. A small number of new peers in the cache guaranteed a fast spread. Figure 12, however, shows that the worm actually spreads slower with a larger C for random peer selection. Two factors cause this counter-intuitive behavior: the worm tries to evade virus throttle by probing one peer a time, and the average probability for a successful infection becomes smaller with a large aging cache. A small cache (10) enables a proactive worm to infect almost 90% population at time 50.

Even with a relatively large cache, a proactive worm could achieve higher prevalence if the infection success probability drops slower with the age. Figure 13 shows the impact of infection probability when a worm probes a cached peer. Given the same age, larger δ leads to smaller probability of successful infection. With a small δ (1.5) and a large cache (100), a proactive worm can infect more than 50% population at time 100.

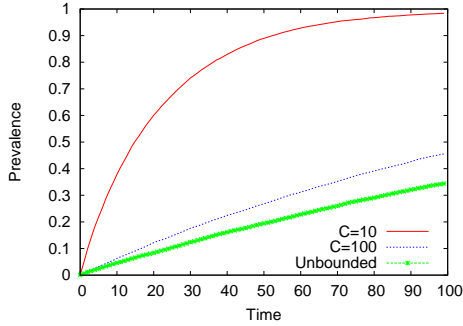


Figure 12. [Proactive] Cache size.

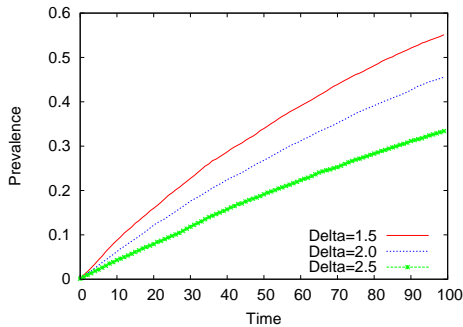


Figure 13. [Proactive] Infection probability.

6 Worm Detection

It is somewhat controversial whether we should detect worm propagations and further develop protection methods for P2P systems, which are used mostly to exchange illegal files nowadays. Putting philosophical debates aside, in this section we only focus on the technical feasibility of detecting non-scanning P2P worms discussed in previous sections.

Existing worm detection methods target random-scanning worms, by capturing the scans spread into unused IP space [28], by detecting exponential scan increases and probe failures [12, 54], or by hypothesis testing on fast port scans [19]. These methods are not effective given P2P worms' non-scanning behaviors. Host-based detection methods, such as Tripwire [20], have potential to detect P2P worms, but it is difficult to deploy them onto wide-area P2P nodes. Thus we seek network-based detection methods that leverage application-level knowledge to identify anomalies.

Passive worms will create and share popular files on the victim hosts, thus distorting the popularity of those nodes. Namely, the victim node may see an increased number of file requests by other peers. Similarly the reactive worms that employ source or mixed infection

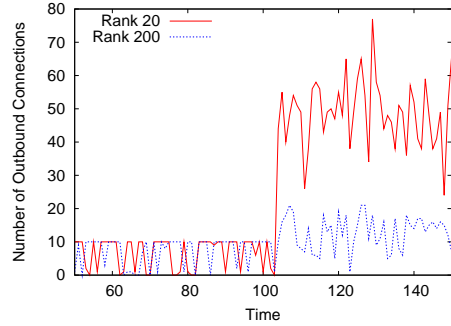


Figure 14. Detectability of reactive worms.

strategies will connect to and infect any peer that contacts the victim node requesting files. Thus an infected popular node being contacted for files by many peers will generate increased outbound connections. Proactive worms try to evade virus throttles by probing one peer per time unit. One could look for increased outbound connections (probes) by aggregating several time units. This approach, however, might not be effective since the cache on each node could be small or the worm only probes a few latest entries (still effective as we show in Section 5). We note that the peers in the caches, however, tend to be popular nodes and thus will receive a disproportional number of inbound probes.

For all these scenarios, we could monitor the rate of either inbound or outbound connections for P2P nodes. A sudden rate increase indicates potential worm activities. Due to limited space, we only focus on the detection of reactive worms and show its representative results in this section. We plan to present detailed detection results on other worms and how these results can be used for worm defense methods in a future publication.

Figure 14 shows the rate changes of outbound connections for two nodes with different popularity ranks, with D set to be 10 and a reactive worm with mixed infection released at time 100 (assuming no immunization applied). Note that both curves show abrupt changes after the worm release, and the less popular node has less dramatic changes (it has less chance to be contacted for files). We also note that there are some delays before the worm hit these nodes and caused the rate changes. If the sensor can monitor the outbound connections of multiple nodes, such as an organizational egress sensor, the aggregated rate may exhibit more obvious changes with worms. We say that the sensor *covers* these n nodes and we show the impact of n later in this section.

To detect abrupt changes in streaming sensor data, we use the algorithm proposed by Guralnik and Srivastava [15]. We define a likelihood estimation criteria L to be the sum of residual squares, after a linear model

$n \setminus \delta$	1.0	2.0	3.0	4.0
1	5.8, 25.2	6.6, 20.3	9.2, 16.7	10.9, 13.6
10	15.3, 2.2	13.3, 1.0	9.1, 0.7	9.3, 0.5
50	5.9, 0.1	5.5, 0.0	6.6, 0.0	7.4, 0.0
100	4.6, 0.0	5.2, 0.0	6.0, 0.0	5.8, 0.0

Table 2. Worm detection results.

is fitted to the time series data. Now given an array of accumulated data points collected from time t_i to t_j , we need to determine whether there is a change-point t_k in the range. We use the criteria that a change-point has been detected if and only if

$$(L(i, j) - L_{\min}(i, j)) / L(i, j) > \delta,$$

where δ is a user defined threshold, and

$$L_{\min}(i, j) = \min_{i+p-1 \leq k \leq j-p+1} \{L(i, k) + L(k+1, j)\}.$$

Here p is the minimum number of data points needed to fit a linear model, and we choose 4 as its default value.

Note that if a change point t_k is detected at time t_j , t_k is always earlier than t_j and we call the difference the *detection delay*. On the other hand, we call the difference of t_j and the actual worm release time the *detection error*. Besides these two metrics, we also compute the number of false positives and false negatives to evaluate the worm detector. Table 2 shows the detection results with different threshold δ and coverage n . The pair of values for each table entry are detection delay and number of false positives, averaged over 100 random runs. We found the detector generates only 1.5% false negatives, and the detection error is always within 2 time unit.

When only one node is monitored, we see the threshold δ controls the tradeoff between detection delay and accuracy. As the threshold increases, detection has a longer delay but fewer false positives. If the sensor covers 10 nodes, the aggregated output makes the jump in time series data less significant (some unpopular nodes are selected). This leads to longer detection delay, but much fewer false positives because the variance in aggregated data is more stable. If the sensor could cover even more nodes, the abrupt jump will be boosted by the popular peers and lead to easier detection (both smaller delay and more accurate). We could have further reduced the detection delay by using a smaller p , such as 2 instead of 4, with a tradeoff of increased false positives.

To put the detection results in context, Figure 6 shows that the reactive worms have a slow start phase until time 5 or 6. If we can detect the worm in its slow start, there is a good chance to contain it before the significant outbreak. Such a detection delay can be achieved by

monitoring either a few popular nodes or a large number of randomly selected nodes (Table 2). To cover a large number of nodes, the sensor could be deployed at the network borders, or be integrated with P2P clients that coordinate to compute aggregated connections by exchanging self-certifying alerts [6]. Montresor et al. propose an aggregation algorithm on P2P networks that could be used for this purpose [24].

While this detection algorithm works well in our simulation framework, we note it is not straightforward to apply it in real-world traffic. A modern P2P system typically also carry non-file sharing traffic, such as node discovery, route maintenance, and online advertisements. Other types of background traffic may also cause false positives to the worm detectors. As future work, we plan to correlate connection rate with other network features to improve the detection accuracy and reduce detection delay.

7 Related Work

Recent efforts to understand widely-deployed P2P systems by analyzing empirically collected data have provided foundation for our work and others [14, 44]. In particular, Gummadi et al. found that the popularity of P2P files follows a Zipf distribution and users fetch files at most once [14]. Other studies also confirmed that the topology of P2P systems approximates a power-law distribution [33, 37], and that the P2P systems are highly dynamic [35]. We have taken all these findings directly into our workload model to drive worm simulation.

Zhou et al. study worm propagation with a simulation framework that implements several P2P protocols such as Gnutella, Gia, and Pastry [53]. They have focused on how to contain proactive worms. They assume the existence of guardian nodes that are immune and capable of detecting worms and issuing non-reputable alerts [6]. It will be interesting to study how fast the worm can be contained using our detection method with their alert subsystem. Yu et al. give analytical results on how different parameters influence worm propagation on both unstructured and structured P2P systems [52]. Several other works [29, 8, 5], like these two, focus on proactive worms and assume that the worm blasts neighbors within a static P2P topology, without considering the evasion of existing worm defense methods such as virus throttles [49].

The non-scanning worms may also propagate through interactions of other Internet applications, such as Email [55], Instant Messaging [23], and SSH [36]. Like previous P2P worm studies, however, they only focused on proactive worms since their topology-driven models lack detailed peer interactions.

Note that accurately simulating worm propagation on a large-scale network is not a trivial task. Most of previous worm studies leverage a customized simulation at abstract level [41, 54, 27, 9]. Liljenstam et al. used a mixed model where part of network is simulated at macroscopic level while the rest is simulated at packet level to achieve scalability [22]. Perumalla and Sundaragopalan performed a full packet-level worm simulation using massive parallelization to study the detailed worm dynamics [30]. Weaver et al. emulated worm propagation on a testbed with scaled-down version of Internet [46]. Unlike these detailed studies, our work still relies on aggregated abstract models though we construct a realistic workload to drive these models. It will be interesting to apply our framework in a packet-level P2P simulator, such as GnutellaSim [16].

There has been a considerable amount of work on detecting random-scanning worms. Telescope-like monitors or honeypots can directly capture random scans [41, 3, 32, 13, 18, 51]. Others identify random-scanning worms by looking for specific network anomalies, such as tree-like propagations [42, 11], exponential scans and ICMP failures [12, 54], fast port scans [19, 45], common payloads [21, 39], and DNS anomalies [48]. Unfortunately these methods are not effective against P2P worms due to their non-scanning nature. Host-based detectors, such as Tripwire [20], have potential to detect non-scanning worms, but they are hard to deploy onto wide-area P2P nodes.

We observe that the files and P2P nodes follow certain popularity distributions and the three types of P2P worms distort those distributions one way or another. This leads to a real-time detection algorithm that can identify the sudden changes in connection rates caused by the worms. In some sense, our approach is similar to virus throttles that contain random-scanning worms by limiting the rate of new connections [49]. Their alert threshold, however, needs to be tuned for individual nodes or applications to reduce false positives [50]. Our approach does not require baselining individual nodes, even though they have different popularities. It also can detect changes in a longer time period, such as in days, which may be invisible to a virus throttle. The simulation results show that our detector is effective against reactive worms, one of the stealthiest worms [41]. We plan to perform further evaluation for other P2P worms with realistic background traffic.

8 Conclusion

In summary, we believe that passive P2P worms are unlikely to be a major threat due to the difficulties of generating a large number of popular names in a dynamic environment. In reality, none of the passive

worms we know have caused major outbreaks since the first passive worm was discovered more than three years ago. Reactive worms, on the other hand, are stealthy and can penetrate a much higher percentage of nodes in a shorter time period than passive worms. The infection strategy, whether source-based, sink-based, or mixed, has great impact on the worm propagation. The proactive worms are the worst among the three P2P worms since they aggressively spread using historically accumulated peer addresses. How the worm selects a peer from the cache is a critical parameter for its propagation speed. The proactive worms, however, are likely to generate more network anomalies such as an increased number of failed connections (the peers are not available when probed).

We have evaluated a detection algorithm for these worms by real-time identification of change-points in the continuously sampled connection rates, using a statistical time-series analysis method. The intuition is based on the observation that the worms distort node popularity, reflected as changes in connection rates. We show that the approach is effective against reactive worms if the sensor monitors a few popular nodes or a large number of randomly selected nodes. As future work, we plan to further enhance this algorithm and evaluate it with realistic background traffic. We are also interested in studying the effectiveness of worm containment based on our detection method.

Acknowledgment

We thank Institute of Security Technology Studies at Dartmouth College for hosting the fellowship of the first author when working on this project. We thank David Kotz for discussions of the early draft of this paper. This work is supported by the Institute for Information Infrastructure Protection at Dartmouth College under award #2003-TK-TX-0003 from the U.S. Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security.

References

- [1] L. A. Adamic. Zipf, Power-laws, and Pareto - A ranking tutorial, Oct. 2000.
- [2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating System Principles*, Banff, Canada, Oct. 2001.
- [3] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor - A Distributed Blackhole Monitoring System. In *Proceedings of the*

- 12th Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2005.
- [4] S. A. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. Technical Report CUCS-039-04, Columbia University, September 2004.
- [5] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. In *Proceedings of the 1st ACM workshop on Rapid Malcode*, Washington, DC, Oct. 2003.
- [6] M. Costa, J. Crowcroft, M. Castro, and A. Rowstron. Can we contain Internet worms? In *Proceedings of the Third Workshop on Hot Topics in Networks*, San Diego, CA, Nov. 2004.
- [7] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS using a peer-to-peer lookup service. In *Proceedings of the First International Workshop on Peer-to-Peer Systems*, pages 155–165, Mar. 2002.
- [8] Z. Dezső and A.-L. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65, 2002.
- [9] B. Ediger. Simulating network worms. <http://www.users.qwest.net/~eballen1/nws/>. Last accessed: November 2005.
- [10] M. Eichin and J. Rochlis. With microscope and tweezers: An analysis of the Internet virus of November 1988. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1989.
- [11] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia. A behavioral approach to worm detection. In *Proceedings of the 2nd ACM workshop on Rapid Malcode*, Washington, DC, Oct. 2004.
- [12] R. S. Gray and V. Berk. Rapid detection of worms using ICMP-T3 analysis. In *Proceedings of the SPIE 2004 Symposium on Defense and Security (formerly AeroSense)*, Orlando, Florida, Apr. 2004.
- [13] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley. Worm Detection, Early Warning and Response Based on Local Victim Information. In *Proceedings of the 20th Annual Computer Security Applications Conference*, Tucson, AZ, Dec. 2004.
- [14] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, Bolton Landing, NY, Oct. 2003.
- [15] V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of the Fifth ACM International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, Aug. 1999.
- [16] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto. Mapping peer behavior to packet-level details: A framework for packet-level simulation of peer-to-peer systems. In *Proceedings of the 11th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, Orlando, FL, Oct. 2003.
- [17] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4), Oct. 2000.
- [18] X. Jiang and D. Xu. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.
- [19] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2004.
- [20] G. H. Kim and E. H. Spafford. The design and implementation of tripwire: a file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, Fairfax, VA, Oct. 1994.
- [21] H.-A. Kim and C. M. U. B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.
- [22] M. Liljenstam, D. M. Nicol, V. H. Berk, and R. S. Gray. Simulating realistic network worm traffic for worm warning system design and testing. In *Proceedings of the 1st ACM workshop on Rapid Malcode*, pages 24–33, Washington, DC, Oct. 2003.
- [23] M. Mannan and P. C. van Oorschot. Instant messaging worms, analysis and countermeasures. In *Proceedings of the 3rd ACM workshop on Rapid Malcode*, Fairfax, VA, Nov. 2005.
- [24] A. Montresor, M. Jelasity, and O. Babaoglu. Robust Aggregation Protocols for Large-Scale Overlay Networks. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, Florence, Italy, June 2004.
- [25] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4), July-August 2003.
- [26] D. Moore, C. Shannon, and k claffy. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the Second ACM Internet Measurement Workshop*, 2002.
- [27] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: requirements for containing self-propagating code. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1901–1910, San Francisco, CA, Mar. 2003.
- [28] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Network telescopes: Technical report. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), July 2004.
- [29] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86, 2001.
- [30] K. S. Perumalla and S. Sundaragopalan. High-Fidelity Modeling of Computer Network Worms. In *Proceedings of the 20th Annual Computer Security Applications Conference*, pages 126–135, Tucson, AZ, Dec. 2004.
- [31] H. Project. Know your enemy: Honeynets, Nov. 2003.
- [32] N. Provos. A Virtual Honeypot Framework. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.
- [33] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, Linköping, Sweden, Aug. 2001.

- [34] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 2001 International Middleware Conference*, Heidelberg, Germany, Nov. 2001.
- [35] S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems*, 9(2), 2003.
- [36] S. E. Schechter, J. Jung, W. Stockwell, and C. McLain. Inoculating SSH against address-harvesting worms. <http://nms.csail.mit.edu/projects/ssh/>, May 2005.
- [37] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transactions on Networking*, 12(2), Apr. 2004.
- [38] M. Singer. Benjamin worm plagues KaZaA. *Internet-news.com*, May 2002.
- [39] S. Singh, C. Estan, and G. V. S. Savage. Automated Worm Fingerprinting. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, San Francisco, CA, Dec. 2004.
- [40] E. H. Spafford. The Internet worm: Crisis and aftermath. *Communications of the ACM*, 32(6), June 1989.
- [41] S. Staniford, V. Paxson, and N. Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, Aug. 2002.
- [42] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dillger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – a graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.
- [43] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, CA, Aug. 2001.
- [44] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Proceedings of the Fifth ACM Internet Measurement Conference*, pages 49–62, Berkeley, CA, Oct. 2005.
- [45] S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum. New Streaming Algorithms for Fast Detection of Superspreaders. In *Proceedings of the 12th Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2005.
- [46] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary results using scale-down to explore worm dynamics. In *Proceedings of the 2nd ACM workshop on Rapid Malcode*, pages 65–72, Washington, DC, Oct. 2004.
- [47] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *Proceedings of the 1st ACM workshop on Rapid Malcode*, Washington, DC, Oct. 2003.
- [48] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based Detection of Scanning Worms in an Enterprise Network. In *Proceedings of the 12th Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2005.
- [49] M. M. Williamson. Throttling Viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of the 18th Annual Computer Security Applications Conference*, San Diego, CA, Dec. 2002.
- [50] C. Wong, C. Wang, D. Song, S. Bielski, and G. R. Ganger. Dynamic Quarantine of Internet Worms. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, Florence, Italy, June 2004.
- [51] V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of the 11th Network and Distributed System Security Symposium*, San Diego, CA, Feb. 2004.
- [52] W. Yu, C. Boyer, and D. Xuan. Analyzing impacts of peer-to-peer systems on propagation of active worm attacks. Technical report, Department of Computer Science and Engineering, The Ohio-State University, Mar. 2004.
- [53] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, and S. Chien. A first look at peer-to-peer worms: Threats and defenses. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, Ithaca, NY, Feb. 2005.
- [54] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington, DC, Oct. 2003.
- [55] C. C. Zou, D. Towsley, and W. Gong. Email worm modeling and defense. In *Proceedings of the 13th International Conference on Computer Communications and Networks*, Chicago, IL, Oct. 2004.