On the Effectiveness of Structural Detection and Defense Against P2P-based Botnets

Duc T. Ha [†] Guanhua Yan [‡] Stephan Eidenbenz [‡] Hung Q. Ngo [†]

[†]Dept of Computer Science and Engineering
University at Buffalo
Buffalo, New York 14200
{ducha,hungngo}@buffalo.edu

[‡] Information Sciences (CCS-3)
Los Alamos National Laboratory*
Los Alamos, NM 87545
{ghyan, eidenben}@lanl.gov

Abstract

Recently, peer-to-peer (P2P) networks have emerged as a covert communication platform for malicious programs known as bots. As popular distributed systems, they allow bots to communicate easily while protecting the botmaster from being discovered. Existing work on P2P-based botnets mainly focuses on measurement-based studies of botnet behaviors. In this work, through simulation, we study extensively the structure of P2P networks running Kademlia, one of a few widely used P2P protocols in practice. Our simulation testbed not only incorporates the actual code of a real Kademlia client software to achieve high realism, but also applies distributed event-driven simulation techniques to achieve high scalability. Using this testbed, we analyze the scaling, clustering, reachability, and various centrality properties of P2P-based botnets from a graph-theoretical perspective. We further demonstrate experimentally and theoretically that monitoring bot activities in a P2P network is difficult, suggesting that the P2P mechanism indeed helps botnets hide their communication effectively. Finally, we evaluate the effectiveness of some potential mitigation techniques, such as content poisoning, sybil-based and eclipse-based mitigation. Conclusions drawn from this work shed light on the structure of P2P botnets, how to monitor bot activities in P2P networks, and how to mitigate botnet operations effectively.
Keywords: *Botnets, Kademlia, structural analysis, monitoring, mitigation*

1 Introduction

Botnets, which are networks of compromised machines controlled by one or a group of attackers, have emerged as one of the most serious security threats on the Internet. With an army of bots at the scale of tens of thousands or more, the computational power of botnets can easily be leveraged to launch large-scale DDoS (Distributed Denial of Service) attacks, or send spamming emails. For instance, it has been reported that six botnets contributed 85% of all spamming emails seen on the Internet [23], and botnets were used to launch DDoS attacks against DNS service [11].

As detection and mitigation techniques against botnets have been stepped up in recent years, attackers are also constantly improving their strategies to operate their botnets. The first generation of botnets typically employ IRC (Internet Relay Chat) channels as their command and control (C&C) centers . Though simple and easy to deploy, the centralized C&C mechanism of such botnets has made them prone to being detected and disrupted. Against this backdrop, peer-to-peer (P2P) based botnets [1] have emerged as a new generation of botnets which can conceal their C&C communication. Without a centralized C&C server, a P2P-based botnet does not suffer from a single point of failure, and its traffic, when buried in the enormous normal P2P traffic in the Internet, is extremely hard to detect.

Current work on structured P2P(Kademlia)-based botnets mainly focuses on measuring existing ones, such as the highly publicized Storm botnet. In this paper, we take one step further to investigate the structural characteristics of Kademlia-based botnets, explore the challenges of monitoring bot activities inside a P2P network, and evaluate the effectiveness of several techniques to disrupt botnet operations. Achieving all these goals calls for an experimental testbed with high flexibility and controllability. Towards this end, we build a botnet simulation testbed, which uses the actual implementation code of a Kademlia client software for high realism, and distributed event-driven simulation techniques for high scalability. Using this testbed, we analyze the structures of Kademlia-based botnets and evaluate several monitoring

*Los Alamos National Laboratory Publication No. LA-UR 08-08095

[1] In fact, they are mostly based on Kademlia, a widely used structured P2P protocol

and mitigation strategies against them.

The key contributions we make from this work can be summarized as follows: *First*, we analyze the scaling, clustering, reachability, and common centrality properties of Kademlia networks from a graph-theoretical perspective. *Second*, we demonstrate, experimentally and theoretically, that monitoring bot activities in a P2P network is difficult, suggesting that the P2P mechanism indeed helps botnets hide their C&C communication effectively. *Third*, we evaluate the effectiveness of some attacks against P2P networks, now used for botnet mitigation, such as content poisoning, sybil-based mitigation, and eclipse-based mitigation. Through this work, we hope to shed some light on the structure of P2P botnets, as well as on the fundamental problems of monitoring and disrupting bot activities.

The remainder of the paper is organized as follows. Section 2 first presents the related work. Section 3 then provides some background on the Kademlia protocol and its variant Kad, which is used in our simulation-based study. Section 4 gives an overview of the design of our simulation testbed. Next, Section 5 analyzes P2P(Kademlia)-based botnets from a graph-theoretic perspective, including their scaling, clustering, reachability, and centrality properties. Section 6 discusses the challenges of monitoring bot activities in a P2P network, both experimentally and theoretically. We further evaluate the effectiveness of three different mitigation techniques against botnet operations in Section 7. Finally, Section 8 concludes our paper.

2 Related Work

Existing studies on botnets mainly fall under the following three main categories: (i) Perform case-study of botnet behaviors and structures; (ii) Model botnets to gain insights on their dynamics; (iii) Propose techniques for botnet detection and disruption. In the *first* category, Freiling et al. [12] infiltrated a real botnet to identify C&Cs and study bot commands. Rajab et al. [21] employed a multifaceted and distributed infrastructure to study botnet behaviors. Some other work focused on measuring botnet sizes by various techniques such as DNS redirection [9] or DNS cache snooping [20]. In the *second* category, Dagon et al. created a diurnal propagation model to capture the diurnal pattern of the bot propagation [9]. A stochastic activity network model has recently been proposed to characterize peer-to-peer botnets in the Möbius tool [22]. Our work in this paper relies on a simulation testbed, in which a P2P-based botnet is modeled using discrete-event simulation techniques. Unlike previous work, this testbed uses the actual implementation code of a popular P2P client software to achieve high realism. In the *third* category, several techniques have been employed to detect botnet existence: anomaly detection [4], traffic or network activity statistics analysis [15]. In our work, we explore the difficulty of monitoring bot activities in a P2P network from both experimental and theoretical perspectives. We also evaluate the effectiveness of existing mitigation techniques against botnet operations in a quantitative fashion. Conclusions drawn from our work are complementary to previous results in botnet detection and mitigation.

3 Primer on Kademlia and Kad

Kademlia, a peer-to-peer (P2P) protocol, was proposed by Maymounkov and Mazieres in [18]. Based on Distributed Hash Table (DHT), it provides a structured approach to P2P applications, where file storing and lookup operations can be efficiently performed with some resource-to-location mapping functions. In Kademlia networks, each node or resource (e.g., file) is associated with a 160-bit identifier in a circular ID space of size 2^{160}. These IDs are generated in a pseudo-random fashion (usually with a cryptographic hashing function), so that they can be deemed as uniformly distributed in the ID space. The distance d between two IDs X and Y is defined as the integral result of the bitwise-XOR operation between X and Y, namely $X \oplus Y$. Each resource is stored on nodes whose IDs are closest to the resource's ID based on this distance metric.

Routing in Kademlia is done in an iterative process based on distances. A node, when searching for an ID (either a resource ID or a node ID), queries its neighbors for new nodes whose IDs are closest to the target ID. Upon receiving the answers, it continues to query those that are closer to the target. This process repeats until no closer node IDs can be obtained from the answer set.

For routing purposes, each node maintains a routing table containing information about its neighbors, such as their IDs, IP addresses, contact ports, etc. Kademlia prefers existing nodes to newly joined ones when updating the routing tables. Further details regarding how routing tables are constructed can be found in [18].

Kad is a variation of the Kademlia protocol that has been adopted by the P2P community on several major P2P networks, including Overnet[2] and eMule [1]. Besides using 128-bit IDs, Kad supports more types of messages than Kademlia, and also handles the routing process in a slightly different manner. More specifically, the search process in Kad consists of two phases:

- **Routing phase**: Similar to the original Kademlia routing protocol, the searching node asks its neighbors for nodes closest to the key ID in an iterative fashion. To accelerate the process, each peer in Kad simultaneously asks for the three closest peers so far in each round. The messages used in this phase are KADEMLIAREQUEST and KADEMLIARESPONSE, in the parlance of Kad.

[2]Overnet was shut down due to copyright violations in late 2006.

- **Item querying phase**: After a certain amount of time since the query starts, the searching node selects some nodes that have responded and then queries for the key. The messages used in this phase are KADEMLIA_SEARCH_REQ and KADEMLIA_PUBLISH_REQ.

Each unit of information stored in a Kad network is associated with a unique key (i.e. an ID). There are three main types of keys: (1) *Source Key*: A source key identifies the content of a file and associates with some information about a source node that is storing the file. Each instance of a file is associated with a unique source key (there can be multiple copies, thus multiple source keys for one file); (2) *Keyword Key*: A keyword key identifies a textual keyword and associates with source keys of files that are related to the keyword; (3) *Note Key*: A note key identifies a comment related to a specific file, and associates with information about that file (through one of its source keys). Keys are periodically republished: every 5 hours for a source key and 24 hours for a keyword or note key. If not republished within these time frames, they will then be removed by the storing peers.

We shall use the terms Kad and Kademlia interchangeably in this paper, since Kad is the only prevalent implementation of Kademlia at present.

4 P2P-Based Botnet Simulation Testbed

Current work on P2P-based botnets mainly focuses on monitoring, either passively or actively, behaviors of some existing P2P-based botnets, such as the Storm botnet [16,17]. Although these bodies of work offer insights on how those botnets operate underground in reality, they have the following disadvantages. *First*, botnet monitoring usually takes place from a single or a few vantage points, thus cannot provide a full and consistent picture of the entire network. *Second*, researchers who attempt to actively measure an existing botnet may interfere with each other, potentially rendering information collected highly biased. For instance, the Storm botnet may have been overestimated due to interference from researchers performing their poisoning mitigation scheme on the botnet [17]. *Last but not least*, performing research on a real botnet may evoke ethical and legal issues that are often neglected by cyber-security researchers [6].

It is thus necessary to have a testbed with flexibility and controllability that can be used to study various aspects of botnet operations, and also to evaluate potential mitigation schemes without the aforementioned restrictions. For this purpose, we develop a simulation-based virtual environment in which we can investigate P2P-based botnets extensively. In contrast to some existing P2P simulators which often model P2P protocols at an abstract level, we use the actual code of aMule [2], a real and popular P2P client software for eMule [1]. The aMule P2P client implements the Kad protocol as described in Section 3. The original aMule code, however, can not work directly because it uses real time, while time in the simulation enviroment is virtual and simulated. To address this problem, we intercept all time-related system calls and redirect them to use the simulation API time system calls. The rest of the client code remains as is. All operations of a Kad network (discovering new neighbors, removing dead nodes, etc.) are handled by the aMule instances as they are done in the real Emule network.

Embedding real implementation code in the testbed preserves most real-life details of the Kad protocol without any abstraction. However, this also introduces another problem: simulating botnets at a realistic scale (e.g., tens of thousands of bots like the Storm botnet) is so computationally prohibitive that it cannot be finished on a single commodity PC within a reasonable amount of time. We resort to PRIME SSF [19], a distributed simulation engine based on conservative synchronization techniques. To achieve better scalability, we simulate only layers 3 (IP) and above (TCP/UDP). We also do not simulate background traffic, and IP routing protocols. Although these abstractions reduce the realism of our simulation, their effect is negligible, as we are only concerned about (Kademlia) traffic at the application layer.

We run our simulator on a distributed platform consisting of 30 machines, each equipped with 2 Pentium III CPUs and 4Gb RAM. On top of this testbed, we first construct a Kademlia network of thousands of aMule instances. Nodes join the network by bootstrapping to a node in a set of special nodes designated as bootstrap nodes, similar to the actual eMule network. They later handle all Kademlia-specific operations (self-advertise, discover new neighbors and remove unresponsive ones, etc.) by themselves (i.e., by the aMule instance). After several days in simulated time, we record a snapshot of the whole network and use it as the starting point for later experiments (in Section 5,6,7).

We observe that the nature of distributed simulation relieves our Botnet simulator from the memory limit of each physical machine. The execution time of our simulator is heavily affected by the debugging output options [3] and the experiment setup. Although we have adopted several techniques to optimize the execution performance (e.g. preloading overlay routing tables from snapshots), we have not rigorously tested its scalability. Our experiments performed in this study, however, suggest that simulating a botnet of size tens of thousands can be done in the order of hours.

5 Structural Analysis

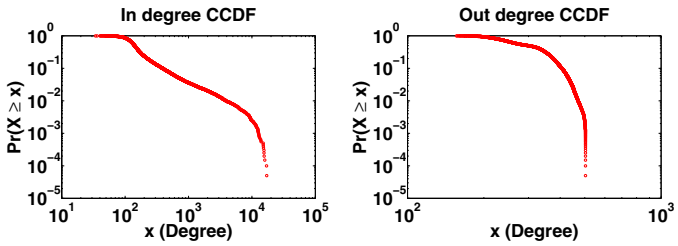
As the P2P protocol is the bedrock of a P2P-based botnet, operations of such a botnet and the corresponding defense schemes are inevitably implicated by its distinguishing P2P structure. In this section, we perform structural analy-

[3] We have five options in our simulator

sis on a Kademlia-based network with 20,000 nodes from a graph-theoretical perspective. Given the network, we form a *directed* graph $G(V, E)$ as follows: each node in the network is also a vertex in graph G and if a node b appears in node a's routing table, an edge from vertex a to b is added to the graph.

5.1 Network Properties

Scaling Property. Many real-world networks, such as the world wide web (WWW) and social networks, have been shown to be scale-free networks, whose degree distributions follow the power law. We are interested in whether a network built on the Kademlia protocol is also a scale-free network. In Figures 1(a) and 1(b), we depict the complement cumulative distribution of the in-degree and out-degree of graph $G(V, E)$, respectively. Visually, if the degree distribution follows a power law distribution, the curve should resemble a straight line. From the figures, it seems that both the indegree and outdegree of $G(V, E)$ do not follow the power law. To verify this observation rigorously, we apply

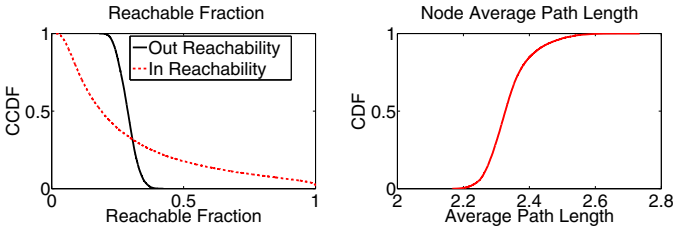


(a) In-Degrees (b) Out-Degrees

Figure 1. CCDF of the Degrees (log-log scale)

the statistical method developed by Clauset et al. [8], which is based on maximum likelihood methods and the Kolmogorov-Smirnov statistic. The resulting p-value, used to measure the goodness-of-fit, is 0 for both the in-degree and out-degree distributions. This further confirms that neither the indegrees nor outdegrees of $G(V, E)$ follow the power law.

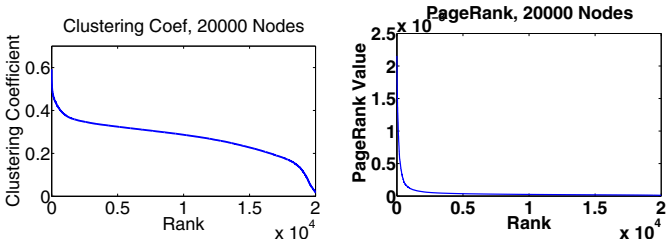
Reachability Property. Figure 2(a) shows the complement cumulative density function (CCDF) of the fraction of the reachable population from each node (out-reachability). Reachability between any pair of nodes is computed from the routing tables of all existing nodes in the network. As can be seen from the figure, 80% of the nodes can reach 25% or more of the node population, but only 10% of the nodes can reach more than 30% of the node population. Interestingly, none of the nodes can reach more than 40% of all possible destinations. This observation, however, does not hold for in-reachability, which shows the fraction of nodes that can reach a specific target node. The CCDF of in-reachability is also depicted in Figure 2(a). We note that about 10% of the nodes can be reached from more than 80% of the population and some nodes can even be reached by all the other nodes. This implies that the network can still function well if the resources are stored on these nodes with high in-reachability. Figure 2(b) shows the cumulative distribution of the average path length to reachable nodes. The average path length of the overall network is only 2.5 hops, suggesting that reachable nodes can be found quickly.



(a) CCDF Reachable Fraction (b) CDF of Node Average Path Length

Figure 2. CDF and CCDF of the Path Length and Reachable Fraction

Clustering Property. The clustering coefficient of a node is defined as the ratio of the number of actual connections among its neighbors over all possible ones . The individual clustering coefficients in our network are shown in Figure 3(a). It can be seen that the Kademlia network is highly clustered: almost 85% of all nodes have clustering coefficient more than 0.2, and the average network clustering coefficient is 0.3. This is much higher than the theoretical value 0.027 in an Erdos-Renyi random graph with the same number of nodes and edges.



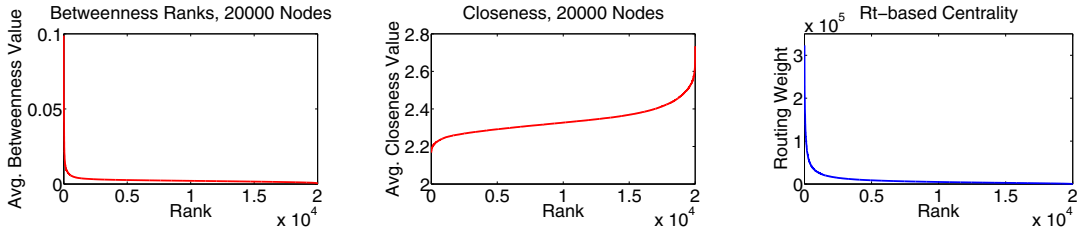
(a) Clustering Coefficients (b) PageRank

Figure 3. Clustering Coefficients and PageRanks

5.2 Centrality Measures

In this section, we study various measures to identify important nodes in a Kademlia-based network. The goal is to find good metrics to identify strategic points. From a practical perspective, they should be easy to use, yet able to give a fairly accurate picture of critical nodes. We consider common measures widely used in other domains.

Degree Centrality. We already showed the distribution of the in- and out-degrees in Figure 1. In many networks degree



(a) Betweenness (b) Closeness (c) Routing Weight

Figure 4. Betweenness, Closeness and Routing-based centrality measures

is an effective measure of the importance of a node. In the Internet, for example, nodes with large number of existing connections typically tend to receive more connections from new nodes.

Eigenvector Centrality. A more sophisticated measure of node importance is the eigenvector centrality. Eigenvector centrality acknowledges that not all connections are of equal importance: nodes connected to more significant nodes have more influence than those connected to less significant ones. This effect can be represented by defining the centrality of a node to be proportional to the average centrality of its neighbors. This approach has shown to be an effective measure in many situations, for example in studies of the Internet topologies [14], finding clusters in information retrieval context [3], or ranking Web pages [5].

We use the method employed for ranking Web pages [5] to compute centrality as our network is directed and is of large scale. The ranking method works as follow. Let $\mathcal{A}'$ be the directed adjacency matrix. For each node v_i define the out degree of v_i as $d_{out}(v_i) = |\{v_j : a_{ij} = 1\}|$. Now consider the stochastic matrix P where $p_{ij} = \frac{a_{ij}\alpha}{d_{out}(v_i)} + \frac{(1-\alpha)}{n}$. The eigenvector with eigenvalue 1 of P gives the ranks of the nodes. Note that since each of the nodes in our network has non zero outdegree, we have no dangling nodes, one important condition for the existence of this vector. Figure 3(b) shows the PageRank values of the nodes against their ranks. We note from the figure that the PageRank measure decays rapidly with the node rank, suggesting that a small fraction of nodes bear very high PageRank values compared to the remaining ones.

Betweenness Centrality. The betweenness centrality of vertex v_i is usually defined as the fraction of geodesic paths between other pairs of vertices on which v_i falls. That is, we find the shortest path (or paths) between every pair of vertices in the network and then derive the fraction of paths on which vertex i appears. In Kademlia-based networks, however, the geodesic paths do not reflect the actual routing paths, as explained in Section 3. To make betweenness a more reasonable centrality measure in our context, we first define a *query set* from a source to a destination as the set of nodes that are queried by the source node in order to get to the destination. The betweenness centrality is then defined as the fraction of query sets between other vertex pairs that contain v_i. Figure 4(a) plots the betweenness values against node ranks in the network. From the figure, we observe a similar Pareto pattern to the PageRank centrality: only a small number of nodes appear frequently in the query sets of all source-destination pairs. For instance, only 20 nodes appear more than 0.5% of the paths between all source-destination pairs. This seems intuitive as P2P protocols are designed to be well distributed so that each node in the network has equal importance.

Closeness Centrality. The closeness centrality of vertex v_i is defined as the mean distance from vertex v_i to every other reachable vertex. We have already showed the cumulative distribution of the average distance per node in Figure 2(a). Figure 4(b) shows these distances against their ranks. It can be seen that almost all nodes on average can reach their reachable destinations within less than 3 hops.

Routing-based Centrality. We define another metric called the *routing weight* based on the actual routes in the network. For each route of length L (excluding the source and target node), we define a *routing weight* for each node in the route as: $1 + \frac{i}{L}$. Here i is the order of the current node, starting from 0 with the node next to the target, and increases towards the nodes near the source. The intuition behind this scheme is to put more weight on nodes that appear close to the source, yet still taking into account the number of routes they are on. Figure 4(c) shows these weights against their ranks. Again, a small number of nodes appear much more frequently and sooner in the routing paths than the rest of the nodes.

Correlations Between Centrality Measures. Some of the above centrality measures can be computed easily, while others require significant efforts. From a practical perspective, if there are strong connections between two measures, one can estimate the measure that is hard to compute based on the more simple measure. We compute the correlation coefficients between these centrality measures with significance level 5% and show the results in Table 1. Two interesting observations can be made from this table: first, although there is a common Pareto pattern shared among page rank, betweenness and routing weight centralities, only the last two appear to have high correlation; second, routing weight correlates even more to in-degree than to betweenness.

Table 1. Correlation Coefficients Between Centrality Measures

	Closeness	Betweenness	Routing	In-Degree	Out-Degree	PageRank
Closeness	1.0000	-0.1112	-0.3684	-0.2572	-0.6392	0.4261
Betweenness	-0.1112	1.0000	0.7744	0.6933	0.0722	-0.3127
Routing	-0.3684	0.7744	1.0000	0.8540	0.4383	-0.4751
In-Degree	-0.2572	0.6933	0.8540	1.0000	0.3606	-0.3217
Out-Degree	-0.6392	0.0722	0.4383	0.3606	1.0000	-0.5144
PageRank	0.4261	-0.3127	-0.4751	-0.3217	-0.5144	1.0000

6 Monitoring

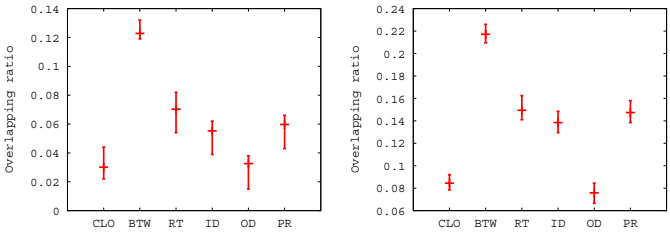
In this and the next section, we focus on the problem of monitoring bot traffic and disrupting bot communications. Effective (in terms of traffic) observation of malicious traffic is a necessary and vital requirement for any system analyst to detect and thwart botnets. Furthermore, given the sheer scale and traffic volume of P2P networks, efficiency (in terms of resources) is just as if not more important. One of the questions we attempt to answer here is: whether the centrality measures defined previously can help identify strategic points in the network, in order to monitor traffic effectively and efficiently.

Our experiments are setup as follows. First, from a snapshot of 20,000 nodes, we reconstruct a Kademlia network. Next, we select 10 random sets of 2000 nodes and use them to simulate bots. The bots are modeled after the real Storm(Peacomm) bots [16]: bots search for the malicious content camouflaged as a filename associated with a specific SourceKey. As Kademlia nodes do not republish the keys they receive from other nodes, the bots are setup to republish the malicious content (i.e., SourceKey) they found so that more bots can also find it. Finally, we use the traffic generated by the bots to benchmark the performance of each centrality measure. More specifically, the top m nodes that receive most traffic in each scenario is compared with the top m nodes ranked by each centrality measure. We define the ratio of the resulting number of common nodes over m as the *overlapping ratio*.

As described in Section 3, a Kad peer goes through two phases to publish (query) a key: (iterative) routing phase and item publishing (querying) phase. The set of nodes involved in each iterative step of the former phase, and those selected in the latter phase is usually very small compared to the total number of responded nodes. This feature, intended to prevent excessive routing traffic, can also limit bot operation. Bots therefore can try to increase the prevalence of keys or shorten the searching time by contacting more nodes during both phases. Consequenty, bot traffic is affected by the bots' decision. We consider three different strategies by the botmaster: in the publish/query phase, as well as each iterating step of the routing phase, a bot selects only a small set of responded nodes as in the standard protocol (*Strategy 1*), 50% (*Strategy 2*), or all of them(*Strategy 3*).

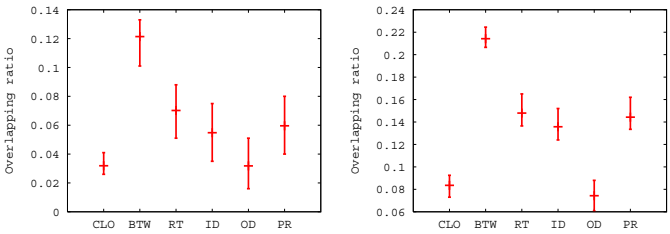
Note that we implicitly assume the bot traffic is recognizable (through the SourceKey). Our focus here is not on how to detect bot traffic from normal network traffic, but rather to measure the amount of observable traffic at some locations. In practice, having access to the traffic is only necessary but not sufficient to detect bot communications.

The performance of each centrality measure in terms of the overlapping ratio for $m = 1,000$ and 2,000 nodes is depicted in Figures 5, 6, and 7, under strategy 1,2 and 3, respectively. In these figures, CLO, BTW, RT, ID, OD, and PR stand for the Closeness, Betweenness, Routing-based, Indegree, Outdegree, and PageRank centrality measures, respectively. At each data point, we give the mean overlapping ratio and its observed minimum and maximum over the 10 sample runs.



(a) 1000 nodes (b) 2000 nodes

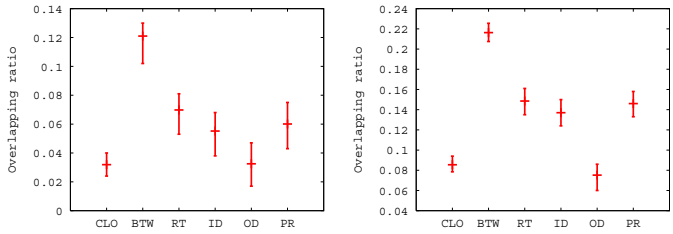
Figure 5. Overlapping ratios under Strategy 1



(a) 1000 nodes (b) 2000 nodes

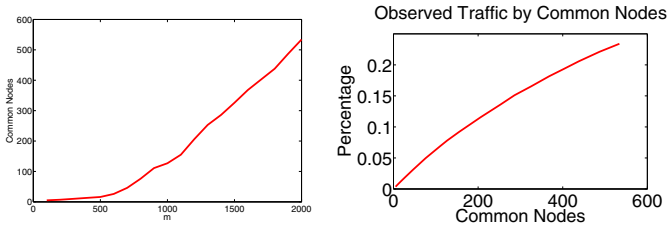
Figure 6. Overlapping ratios under Strategy 2

From these plots, it can be seen that none of the standard centrality measures provides good prediction of important nodes in terms of observed traffic, regardless of the strategy employed by the botmaster. In fact, their performance



(a) 1000 nodes (b) 2000 nodes

Figure 7. Overlapping ratios under Strategy 3



(a) Common set size (b) By common nodes

Figure 9. Observed Traffic

is close to the case when the nodes are randomly selected, as can be seen from the following lemma.

Lemma 6.1. *Let X be a fixed set of size m in a space $N, |N| = n \geq m$. Let Y be another set of the same size selected randomly from N. The expected number of common elements between X, Y can be computed as:*

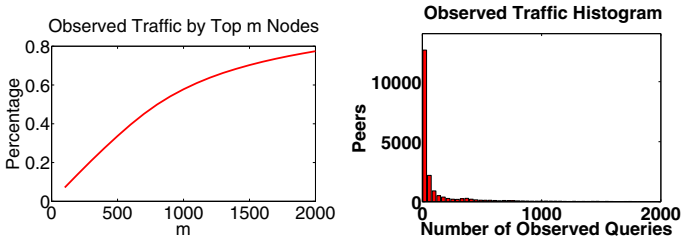
$$E[|X \cap Y|] = \frac{m^2}{n} \tag{1}$$

Proof. Denote $c = |X \cap Y|$. We have:

$$\begin{aligned} E[c] &= \sum_{k=0}^{m} k\mathrm{P}[c=k] = \sum_{k=1}^{m} \frac{k\binom{m}{k}\binom{n-m}{m-k}}{\binom{n}{m}} \\ &= \frac{m\binom{n-1}{m-1}}{\binom{n}{m}} = \frac{m^2}{n} \end{aligned}$$

□

Plugging $n = 20000$, $m = 1000$ and $m = 2000$, we have $E[|X \cap Y|] = 50$ and $E[|X \cap Y|] = 200$, corresponding to the overlapping ratio of 0.05 and 0.1, respectively. From these numbers, it can be seen that all centrality measures except Betweenness do not yield better results than a random approach.



(a) By top m nodes (b) Histogram

Figure 8. Observed Traffic

It is also possible that the traffic is well distributed over the whole network, rendering any ranking method ineffective. To verify this scenario, we look at the traffic observed by top m nodes with heaviest traffic in each sample run. Figure 8(a) shows this relation (averaged over 10 runs). It can be seen that the query traffic is disproportionally distributed. For example, nearly 80% of all traffic is observed (received) by 10% of the population (2,000 nodes out of 20,000 in our case). This is further confirmed in Figure 8(b), which shows the histogram of the number of observed queries by each node. The majority of the nodes only observe a few queries, while a small set of nodes claim the lion share.

Figure 9(a) depicts the number of common nodes shared among the top m across all 10 scenarios, while the fraction of traffic observed by these common nodes (when $m = 2000$) is shown in Figure 9(b). These figures show one important characteristic of Kad networks: the routing query traffic is very dynamic and is strongly affected by the locations of searching peers in the network. As a result, centrality measures discussed in Section 5 all perform poorly in predicting critical nodes. They can still capture some (common nodes), as seen from Table 2. This table shows the number of nodes with most traffic across all 10 botset scenarios that each centrality measure recognized. Unfortunately, even for an optimal measure that can identify all of the common nodes, the amount of observed traffic by these nodes is still very small as shown in Figure 9(b).

Fixed Bot Communication Monitoring. The above results show that the set of traffic-critical nodes and the traffic they observe strongly depend on the bots' locations on the Kademlia network. This guarantees that any static monitoring scheme will perform poorly for some bot distributions. However, it is still possible to observe most traffic when the bot locations are fixed, as suggested in Figure 8(a). In practice, we can collect a list of nodes that are suspicious. A natural question arises: if we know (or can estimate) the communication paths among these nodes (from the network routing tables), which nodes should we select so that communication traffic generated by these suspicious nodes can always be monitored? We formulate it as the following problem:

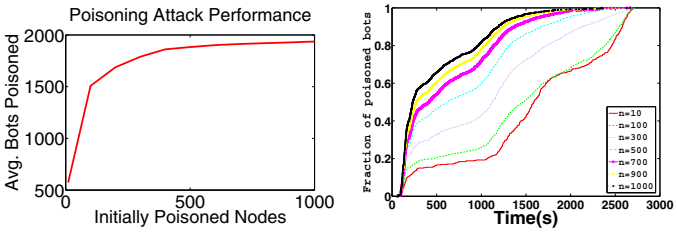
Problem 1. **(Bot Communication Monitoring–** BCM) Given a set of communication paths $P = \{p_1, p_2, \ldots, p_n\}$ in a directed graph $G = (V, E)$, where each path p_i is an ordered list $< v_{i1}, \cdots >$ of nodes in V. A subset $I \subseteq V$ intersects a path p_i if $\exists v_j \in I$ and $v_j \in p_i$. Find the minimum intersecting set I such that $\forall p_i \in P$, I intersects p_i.

Table 2. Number of common nodes among the top m recognized by the centrality measures

m	Total number of common nodes	Closeness	Betweenness	Routing	In-Degree	Out-Degree	PageRank
1000	127	2	46	29	23	12	26
2000	535	40	219	152	142	75	152

Note that this problem is not restricted to Kademlia networks, but is applicable to any structured P2P networks. Here, each path represents a communication path estimated between two suspicious nodes. One would hope to find the smallest set of nodes that lay on all of these paths, so that the monitoring cost can be minimized. Unfortunately, the problem is intractable, as shown by the following theorem, whose proof by reduction from the HITTING SET problem [13] is straightforward and thus skipped.

Theorem 6.2. *The general version of* BCM *is* NP*-hard.*



(a) Avg. num. of bots poisoned (b) Frac. of bots poisoned

Figure 10. Performance of Poisoning Schemes

7 Mitigation

Besides monitoring, system analysts may choose to employ different techniques to thwart botnet communications. In this section, we present the result of using our testbed to explore the effectiveness of several potential defense schemes against botnets. Specifically, we shall look at three schemes: (content) poisoning, sybil, and eclipse-based mitigation.

7.1 Poisoning-Based Mitigation

As bot command keys are used by bots to search for new commands in a P2P-based botnet, they are actually one Achilles' heel of such botnets: once these keys are known, we can inject benign contents with the same key into the network so that some bots may not receive the original bot commands. This technique is thus called *poisoning attack* and was first experimented in [16].

In our simulation, we publish the bot command key with benign contents on a set of nodes. Bots that make search requests to these special nodes are considered to be "poisoned" with the content we provided. This is because the poisoned nodes reply to any request for the bot command with the content they have (which we have provided). This method is reported to be successful in [16] with heuristic choices of key parameters. One specific important parameter of this method is the size of the initially poisoned set. To study the impact of this factor in a quantitative fashion, we vary the size of the initially poisoned set to be 10, 100, 300, 500, 700, 900, and 1000. The poisoned nodes are chosen from those with top betweenness values, since as shown in the previous section, the betweenness centrality measure seems to provide more nodes with high query traffic than other measures.

Figure 10(a) shows the average number of affected bots that received poisoned bot commands as the initially poisoned set size varies. It can be seen that although betweenness centrality measure is not efficient for traffic monitoring, nodes selected by this measure can still be effective in poisoning the bots. For instance, using only 10 initially poisoned nodes we can poison on average 25% of the total number of bots (more than 500 bots), regardless of their locations. With 100 nodes, we can poison more than half of the bots. However, it is more costly to poison a large portion of the bot population. After 400 nodes, the size of the initially poisoned set just makes little difference in the final number of poisoned bots, though the process can be expedited (shown in Figure 10(b)).

It seems counterintuitive that nodes with high betweenness values can perform well here while poorly in monitoring traffic. However, note that in monitoring, the amount of observed traffic matters, while in poisoning, bot coverage is more important.

7.2 Sybil-Based Mitigation

Table 3. Sybil-based Mitigation Scenarios

	Manipulated Nodes	IDs per Manipulated Nodes	Total No. of IDs
Scen. (1)	10	1000	10000
Scen. (2)	100	100	10000
Scen. (3)	1000	10	10000
Scen. (4)	100	1000	100000
Scen. (5)	1000	100	100000
Scen. (6)	1000	300	300000

Instead of passively monitoring traffic, the system analyst can introduce a set of node IDs into the network. These IDs are controlled by a small number of physical processing nodes, rather than full-fledged P2P clients. This way, the system analyst still has access to the query traffic routed to these IDs, without bearing the cost of maintaining the same number of regular IDs. More advanced systems can also generate dynamic IDs when queried, or ignore (not responding to) the

Table 4. Traffic Fraction Observed by Sybil Agent

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	2.6017e-06	0.0023	1.6122e-05	0.0142	0.021112

Table 5. Bot Coverage by Sybil Agent

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	6.7	1935.3	26.6	1998.8	1999

traffic they are not interested in. This is a form of *Sybil attack*, described in [10] as a potential vulneratility of P2P networks. It can be used in many more sophiscated attacks. At present, we are not aware of any effective solution against this threat in practice, despite many proposals in the literature.

In our simulation, we inject a varied number of IDs into the routing tables of a selected set of nodes in the network. Again, these nodes are chosen based on the betweenness centrality measure. All injected Sybil IDs are associated with one extra node designated as the central processing node called *Sybil Agent*. This agent is responsible for all requests directed to a Sybil ID. We only simulate a passive agent that only collects requests, and does not generate any dynamic ID. Using this setup, we study the performance of this approach with respect to some key parameters, such as the number of nodes to which the Sybil IDs will be introduced and the number of Sybil IDs per node.

We consider three different sets of Sybil IDs with size 10000, 100000 and 300000. For the first two sets, we consider several sub-scenarios when the set of nodes to be manipulated varies. Table 3 details the configuration of each of these scenarios. Table 4 shows the fraction of queries that were sent to the Sybil agent through these IDs. Two important observations can be made from the results: one, the traffic received by the Sybil Agent is very small even when the number of injected IDs is large; two, it is much more effective to spread out the IDs than to inject more of them into fewer manipulated nodes.

Another metric to assess the performance of Sybil-based mitigation is the number of bots that query at least one Sybil ID. Table 5 shows the average number of such bots, out of 2000 bots. Here we can see the Sybil Agent has very good coverage of the bot population when we send Sybil IDs to a large number of nodes. Again, given the same number of Sybil IDs, it is better to spread out the IDs. Note that this is not obvious because sending more IDs also increases the chance to catch more keys.

7.3 Eclipse-Based Mitigation

Eclipse attacks against P2P networks were first described by Castro et al [7]. In this type of attacks, a set of nodes collaborate to separate a node from the rest of the network by cutting off all information to and from of the node. This can only be done when all incoming and outgoing traffic is directed through the malicious nodes. To stop outgoing traffic, the victim's routing table needs to contain only information about malicious nodes. For incoming traffic, the malicious nodes must be able to interfere with all queries towards the victim. If any of these conditions fails, the victim will still be able to contact benign nodes, get updated with fresh information and break out of the quarantine.

We only look at incoming traffic towards the victim, since poisoning the victim's whole routing table requires significant time, it may even be impossible if existing nodes in the table are still alive (due to Kad's preference of existing nodes). In our experiments, we use our Sybil IDs as the malicious nodes. Any bot that queries at least one of these IDs within 120 seconds are considered to be fooled by our nodes. Table 6 shows the number of bots among the 2,000 bots in the network that can be fooled. It shows that eclipse-based mitigation is not effective against botnets, even though Sybil-based mitigation can achieve very good coverage. This is because we require the malicious nodes to be contacted at a much earlier state of the querying process. Our simulation result also agrees with the outcome in [16], where the authors could not eclipse bot commands with a very large number of Sybil IDs.

8 Conclusions

In this work, we build a P2P-based botnet simulation testbed, which uses actual implementation code of the Kad P2P protocol to achieve high realism. This simulation testbed employs distributed event-driven simulation techniques for scalability. With this testbed, we analyze the structural characteristics of Kademlia-based botnets, explore the challenges of monitoring bot activities inside the network, and evaluate the effectiveness of mitigation techniques proposed in the literature. We hope our results drawn from this work can shed lights on the structure of P2P botnets, how to monitor bot activities in P2P networks, and also how to mitigate botnet operations effectively.

Table 6. Bot Coverage under Eclipse-Based Mitigation

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	0.1	23.6	1.2	36.4	39.9

Acknowledgment

Hung Ngo and Duc Ha were supported in part by NSF CAREER Award CCF-0347565. We would like to thank the anonymous referees for the valuable comments and Dr. Yves Roudier for his help on revising this paper.

References

[1] http://www.emule-project.net.

[2] http://www.amule.org.

[3] Y. AZAR, A. FIAT, A. KARLIN, F. MCSHERRY, AND J. SAIA, *Spectral analysis of data*, in STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, New York, NY, USA, 2001, ACM, pp. 619–626.

[4] J. R. BINKLEY AND S. SINGH, *An algorithm for anomaly-based botnet detection*, in SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, Berkeley, CA, USA, 2006, USENIX Association.

[5] S. BRIN AND L. PAGE, *The anatomy of a large-scale hypertextual web search engine*, vol. 30, Amsterdam, The Netherlands, The Netherlands, 1998, Elsevier Science Publishers B. V., pp. 107–117.

[6] A. J. BURSTEIN, *Conducting cybersecurity research legally and ethically*, in LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[7] M. CASTRO, P. DRUSCHEL, A. GANESH, A. ROWSTRON, AND D. S. WALLACH, *Secure routing for structured peer-to-peer overlay networks*, in Proceedings of OSDI'02, 2002.

[8] A. CLAUSET, C. R. SHALIZI, AND M. E. J. NEWMAN, *Power-law distributions in empirical data*. http://arxiv.org/abs/0706.1062.

[9] D. DAGON, C. ZOU, AND W. LEE, *Modeling botnet propagation using time zones*, in NDSS, The Internet Society, 2006.

[10] J. R. DOUCEUR, *The sybil attack*, in IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, London, UK, 2002, Springer-Verlag, pp. 251–260.

[11] `http://securitywatch.eweek.com/exploits_and_attacks/everydns_opendns_under_botnet_ddos_attack.html`.

[12] FREILING, HOLZ, AND WICHERSKI, *Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks*, in ESORICS: European Symposium on Research in Computer Security, LNCS, Springer-Verlag, 2005.

[13] M. R. GAREY AND D. S. JOHNSON, *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[14] C. GKANTSIDIS, M. MIHAIL, AND E. ZEGURA, *Spectral analysis of internet topologies*, vol. 1, March-3 April 2003, pp. 364–374 vol.1.

[15] G. GU, V. YEGNESWARAN, AND MARTIN, *Bothunter: Detecting malware infection through ids-driven dialog correlation*, pp. 167–182.

[16] T. HOLZ, M. STEINER, F. DAHL, E. BIERSACK, AND F. FREILING, *Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm*, in LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, Berkeley, CA, USA, 2008, USENIX Association, pp. 1–9.

[17] C. KANICH, K. LEVCHENKO, B. ENRIGHT, G. M. VOELKER, AND S. SAVAGE, *The heisenbot uncertainty problem: challenges in separating bots from chaff*, in LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[18] MAYMOUNKOV AND MAZIERES, *Kademlia: A peer-to-peer information system based on the XOR metric*, in International Workshop on Peer-to-Peer Systems (IPTPS), LNCS, vol. 1, 2002.

[19] `http://lynx.cis.fiu.edu:8000/twiki/bin/view/Public/PRIMEProject`.

[20] M. RAJAB, J. ZARFOXX, F. MONROSE, AND A. TERZIA, *My botnet is bigger than yours (maybe, better than yours)*, Proceedings 2007 USENIX First workshop on Hot Topics in Understanding Botnets, (2007).

[21] M. A. RAJAB, J. ZARFOSS, F. MONROSE, AND A. TERZIS, *A multifaceted approach to understanding the botnet phenomenon*, in Internet Measurement Conference, J. M. Almeida, V. A. F. Almeida, and P. Barford, eds., ACM, 2006, pp. 41–52.

[22] E. V. RUITENBEEK AND W. H. SANDERS, *Modeling peer-to-peer botnets*, in Proceedings of IEEE QEST'08, 2008.

[23] `http://www.theregister.co.uk/2008/02/29/botnet_spam_deluge`.