Modeling and Analysis of Self-stopping BT Worms Using Dynamic Hit List in P2P Networks

Jiaqing Luo, Bin Xiao, Guobin Liu, Qingjun Xiao
Department of Computing
The Hong Kong Polytechnic University
Hong Kong
{csjluo, csbxiao, csgliu, and csqjxiao}@comp.polyu.edu.hk

Shijie Zhou
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, P.R. China
sjzhou@uestc.edu.cn

Abstract

Worm propagation analysis, including exploring mechanisms of worm propagation and formulating effects of network/worm parameters, has great importance for worm containment and host protection in P2P networks. Previous work only focuses on topological worm propagation where worms search a hosts neighbor-list to find new victims. In BitTorrent (BT) networks, the information from servers or trackers, however, could be fully exploited to design effective worms. In this paper, we propose a new approach for worm propagation in BT-like P2P networks. The worm, called Dynamic Hit-List (DHL) worm, locates new victims and propagates itself by requesting a tracker to build a dynamic hit list, which is a self-stopping BT worm to be stealthy. We construct an analytical model to study the propagation of such a worm: breadth-first propagation and depth-first propagation. The analytical results provide insights of the worm design into choosing parameters that enable the worm to stop itself after compromising a large fraction of vulnerable peers in a P2P network. We finally evaluate the performance of DHL worm through simulations. The simulation results verify the correctness of our model and show the effectiveness of the worm by comparing it with the topological worm.

1. Introduction

Worms can propagate quickly and stealthily in P2P networks and pose a serious threat to popular P2P applications where worms have the potential to compromise hosts in a large scale. The particularly designed P2P worms neither need to probe random IP addresses nor generate high rates of failed connections. As a result, they could cause less abnormal behaviors which lead to more stealthy in detection. In recent years, P2P worms have already emerged in P2P file sharing systems like BearShare, KaZaa and Gnutella. In order to counter the threat from P2P worms [2], [4], [13], we need to investigate their propagation mechanisms and used parameters. The propagation mechanism of a worm shows how the worm finds new victims and which strategy it uses to duplicate itself. The effect of a parameter in the worm setting indicates why and how much the parameter change will affect the worm propagation. Thus, the study of worm propagation, including mechanism and parameters, has become an important issue for worm analysis and containment.

To date, little research has been conducted into exploring potential mechanisms of P2P worm propagation. The possible reason is that, at its early stage of study, P2P worm was only viewed as a form of topological worm which searches local information to find next victims [13]. However, the high penetration of P2P protocols and applications has provided the worm writers with a potential means of locating new victims in various ways. For example, in BitTorrent (BT) networks, the information from servers or trackers can be utilized for a worm design, which is a new threat that we have not yet prepared for.

To precisely characterize P2P worm propagation remains a challenging problem. Recent studies [5], [9], [10], [12] adopt the epidemic model to study P2P worm propagation.

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

However, some network parameters used in the model have not been fully understood previously and need to be further investigated. Network parameters, which can greatly affect the worm propagation performance, include network size, average node degree, churning rate, etc. Moreover, the worm parameters, which may also affect the worm propagation, have been largely ignored in previous studies.

In this paper, we focus on the designing of P2P worms in BT networks. We demonstrate a novel mechanism for P2P worm propagation. The worm, called Dynamic Hit-List (DHL) worm, is different from both topological and static hit-list worms in that it finds new victims and propagates itself by requesting a tracker to build a dynamic hit list. The dynamic hit list contains the most recent active peers returned from the tracker. The DHL worm poses an imminent threat to BT networks. First, it can spread at a fast speed but is not seriously impacted by network parameters, since its propagation does not rely on peer's neighbor information. Second, it can be easily implemented without waiting to generate a static IP list of all peers in a P2P network. To understand the damage of DHL worm, we formulate a mathematical model that precisely characterizes the propagation of the worm, to be either breadth-first or depth-first propagation. The model provides an insight into DHL worm design to enable the worm to stop itself after compromising a large fraction of vulnerable peers, which makes itself more stealthy and hard in detection [6]. Without formerly known network parameters, our model can closely display worm behaviors. In simulations, we investigate the impact of each worm parameter (e.g. TTL, number of returned peers from a tracker, etc.) on the DHL worm propagation. The simulation results verify the correctness of our model and show the effectiveness of the worm by comparing it with the topological worm.

The rest of the paper is organized as follows. Section 2 gives an overview of the BT protocol and Section 3 describes the design principle of the DHL worm. In Section 4, we construct an analytical model to study the DHL worm propagation. Section 5 presents both analytical and simulation results for the DHL worm, and makes comparisons with the topological worm. Section 6 presents important related work and Section 7 concludes the paper.

2. Bittorrent Overview

In this section, we explain how a peer successfully gets IP addresses from a tracker, which is used to build the dynamic hit list in BT (BitTorrent) networks. We first give a brief introduction of the BT protocol and then focus on peer-to-tracker communication, showing some details of the protocol traced from a real-world BT network.

2.1. Brief Introduction of BitTorrent

A BT network is commonly composed of *web servers*, *trackers*, *seeds* and *leechers*. A web server is typically used for publishing or downloading .torrent files. To manage connections, a tracker keeps track of who is downloading and/or uploading a particular file, and makes this information available to others who want to find the file. Peers in the network are either *seeds* which have a complete file and are willing to serve it to others, or *leeches* which are still downloading the file but are willing to serve the blocks that they have already have to others. Before joining a *swarm*, which is a peer group sharing a particular file, a peer firstly downloads a .torrent file from a web server. The .torrent file contains a URL list of trackers and other information related the sharing file. Then, it makes an *HTTP GET* request to the tracker on the URL list. Upon receiving the request, the tracker returns a random subset of peers sharing the same file. The peer then attempts to initiate TCP connections with the returned peers, which then become its neighbors. If the number of neighbors are below 20. it contacts to the tracker again to collect a list of additional peers it could connect to.

2.2. Peer-to-Tracker Communication

To explore BT protocol for worm design, we have a special interest in the communication between peer and tracker. The *HTTP GET* request sent by a peer contains a tracker's URL and some parameters. The following 3 parameters are required: *info_hash* is the hash value of info field in .torrent file. The tracker classifies peers into different swarms depending to this field. *peer_id* is peer's ID. *port* is the peer's port used for file sharing. Another useful parameter, *numwant*, is the number of peers to return. Its default value is 50, but it could be much higher than 50. For example, the value of *numwant* is 200 in BitComet, a popular downloading tool based on BT protocol. If the swarm size is smaller than *numwant*, all peers in the torrent will be returned for a request.

After receiving a request, the tracker sends a response which includes: a list of active peers (*peer_id*, *IP* and *port*), number of seeds, number of leechers, and interval between two requests. The sum of seeds and leechers is the size of the swarm, which is the total amount of peers sharing a certain file. The time interval between two requests is 20 minutes. If a peer doesn't send a request within 30 minutes, the tracker assumes that the peer has left the swarm and deletes the corresponding record.

An interesting observation is that the tracker distinguishes different peers depending on their *port* and *IP*. In other words, peers with a same *IP* and *port* are viewed as one, even though they have different *peer_id*. A new record

is added only if the tracker receives a request containing a new pair of *IP* and *port*.

3. Dynamic Hit-List Worm Design

In this section, we first describe how DHL (Dynamic Hit-List) worm propagates from one peer to another. We then propose two strategies for the worm spread: breath-first propagation and depth-first propagation.

3.1. Propagation Process

A DHL worm firstly pretends to be a normal client for obtaining targets' IP addresses from a tracker as the dynamic hit-list, and then spreads itself through vulnerabilities in a network to infect those targets. A compromised peer with the DHL worm can repeat this process to infect other more peers. We do not address how a DHL worm compromises a peer at the network layer, but focus on how it propagates at the overlay layer.

Peers in the BT network can be classified into different swarms according to the files they are sharing. In Figure 1, peers P_a, and P_b sharing the same file 1 are in swarm A. Peer P_c sharing file 2 is in swarm B and peer P_d sharing file 3 is in swarm C. A DHL worm does not intent to infect all swarms at a time, but only one of them. The attacking peer, who initiates the DHL worm, called an *initiator*, repeats attacks till all swarms are infected by the worm.

A DHL worm works as illustrated in Figure 1. Suppose P_a and P_b are vulnerable peers and a DHL worm aims to infect all peers in swarm A. The propagation process can be decomposed in four steps:

1 The initiator downloads a .torrent file from a web server, e.g., a .torrent file related to file 1. This torrent file contains a URL list of trackers.

2 The initiator contacts to the trackers known from the list by sending an *HTTP GET* request.

3 The tracker responds the request by returning a list of active peers who are sharing the file 1. For example, it returns P_a's *IP*, *port*, and *peer_id*.

4 After obtaining the information of P_a, the initiator forwards a worm copy to P_a. The payload of DHL worm contains an *HTTP GET* string including *URL*, *port*, and *peer_id*. Note that *peer_id* is randomly generated, but *port*is P_a's port. The reason is that the use of P_a's port can avoid adding a new record on the tracker.

After successfully compromising P_a, the worm on P_a can request the tracker, and infect P_b in a similar way.

3.2. Propagation Strategies

DHL worm propagates in a multi-hop fashion through the BT network. As can be seen in Figure 2, the initiator attempts to directly infect 5 peers. 2 of them are invulnerable peers which are patched or well protected. Thus, only 3 peers are infected at hop 1. Each newly infected peer then scans other 4 peers returned from a tracker. Since the return peers are randomly selected, it is possible that a newly infected peer sends a worm copy to the peer who has already been infected. Due to the overlapped scanning, only 9 distinguished peers are hit by the worm, 6 of them are infected at hop 2. The worm continues its propagation till TTL (Time to Live) decreases to 0. Depending to different requirements of parameters, we divide the propagation strategies of DHL worm into two categories: one is breath-first propagation, the other is depth-first propagation.

3.3. Breath-first propagation

Breath-first propagation tries to compromise a majority of peers in the swarm within a limited time interval. For that purpose, the initiator uses a small TTL value $h_{limited}$. If $h_{limited} = 1$, it has to learn IP addresses of all peers in the swarm, which indicates that the worm is a form of static hit-list worm. To cover a large fraction of peers within $h_{limited}$ hops, the initiator needs to infect enough peers at hop 1. Normally, if the swarm size is big, the worm needs a large number of infected peers at hop 1 for a given $h_{limited}$. Although increasing the number of infected peers at hop 1 could speed up the worm, the initiator is more likely to be detected due to its continuously requests. A model described in the next section will help us to choose the minimal number of infected peers at hop 1.

3.4. Depth-first propagation

Depth-first propagation aims to limit abnormal behaviors of the initiator. When requesting a tracker, the initiator acts like a normal peer who is trying to join the swarm. That means the number of infected peers at hop 1 does not exceed the default value of *numwant*. A small number of infected peers at hop 1 leads to a low worm speed, namely, the worm needs a large TTL. If TTL is too large, the worm will generate tens of hundreds of worm copies, which may cause an obvious abnormal network traffic. Thus, it is necessary to set an appropriate TTL to limit the abnormal traffic. In next section, we will give another model to compute the minimal TTL.

Choosing suitable parameters, the worm could stop itself after compromising an expected ratio of vulnerable peers in the swarm. In other words, we could precisely control the range of worm propagation by adjusting the parameters mentioned above.

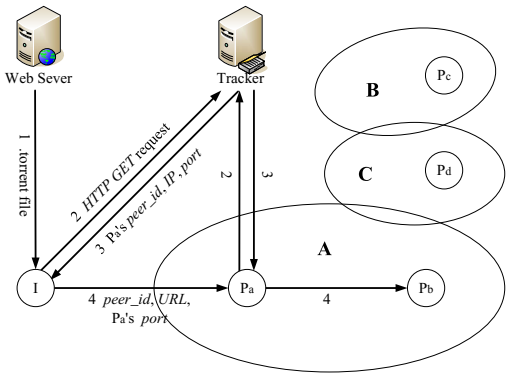

Figure 1: The propagation process of DHL worm

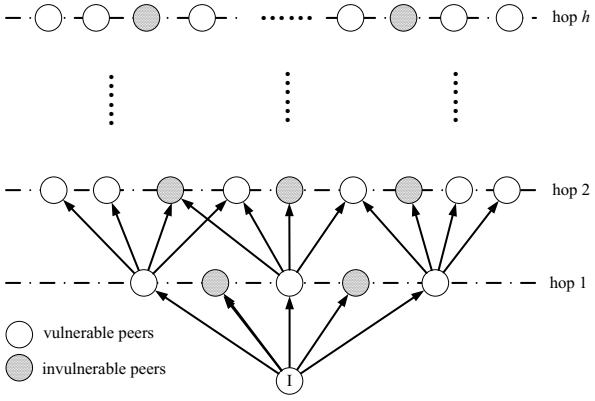

Figure 2: The propagation stategy of DHL worm

4. Modeling Dynamic Hit-List Worm

In this section, we first describe assumptions and notations used in the model. Then, we formulate a mathematical model to formulate the effects of parameters presented in the previous section.

4.1. Assumptions and Notations in Modeling

With some reasonable assumptions made, we are able to deeply understand how DHL worm propagates through the network.

Assumption 1: There is only one initiator in the network. Before initiating an attack, it selects worm parameters according to the size of a swarm.

Assumption 2: An infected peer will not be effective any more, after it delivers worm copies to the peers returned from a tracker.

Assumption 3: A newly infected peer requests a default number of active peers from a tracker.

Assumption 4: There exists a time interval between hop $(h-1)$ and h. In other words, the infections at different hops are independent of each other.

Our assumptions could minimize the communication overhead between infected peer and tracker. Thus, the tracker will not receive too many requests within a very short time interval. This is very important for DHL worm design, because the tracker can easily detect abnormal behaviors and slow down the worm propagation by simply shutting down itself for a while or blacklisting the suspected peers.

The notations used in our models are listed in Table 1.

Table 1: Parameter Definition

N	total number of peers in a swarm
V	total number of vulnerable peers in a swarm
F_h	number of infected peers (vulnerable and infected) at hop h
INF	ratio of infected peers, $INF = \frac{\sum_{i=1}^{h} F_i}{V}$
R	number of active peers returned from a tracker in a request
p	probability of an uninfected peer is returned in a request
$\sim p$	probability of an uninfected peer is not returned in a request
P_h	probability of an uninfected peer is returned in all requests at hop h
$\sim P_h$	probability of an uninfected peer is not returned in all requests at hop h
h	number of hops

4.2. Propagation Model

In DHL worm, the ratio of infected peers is affected by different worm parameters. We suppose the initiator directly infects F_1 peers. Each infected peer one hop away from the initiator requests R active peers from the tracker and tries to infect them. Since the return peers are randomly selected, each peer, no matter it is infected or not, has an equal chance to be returned in a request. We have:

$$\sim p = 1 - \frac{R}{N} \tag{1}$$

Clearly, the requests sent by F_1 infected peers are independent. The probability that an uninfected peer is not returned in all requests at hop 1 is:

$$\sim P_1 = \sim p^{F_1} = (1 - \frac{R}{N})^{F_1} \tag{2}$$

By Equation (2), the number of infected peers at hop 2 can be written as:

$$F_2 = (1- \sim P_1)(V - F_1) \tag{3}$$

Similarly, the probability that an uninfected peer is not returned in all requests at hop $(h-1)$ can be written as:

$$\sim P_{h-1} = \sim p^{F_{h-1}} \tag{4}$$

The number of infected peers at hop h is given by:

$$F_h = (1- \sim P_{h-1})(V - \sum_{i=1}^{h-1} F_i) \tag{5}$$

4.3. Breath-First Propagation Model

We limit h to $h_{limited}$ and try to find the minimal F_1 for a given INF. Given INF, we can get the minimal F_1 from the following equation:

$$INF < \frac{\sum_{i=1}^{h-1}(1- \sim p^{F_{h-1}})(V - \sum_{i=1}^{h-1} F_i)}{V} \tag{6}$$

$$h = h_{limited} \tag{7}$$

4.4. Depth-First Propagation Model

We restrict F_1 to be no more than R and obtain the minimal h for a given INF. Given INF, the minimal h can be calculated by solving the following equations:

$$INF < \frac{\sum_{i=1}^{h-1}(1- \sim p^{F_{h-1}})(V - \sum_{i=1}^{h-1} F_i)}{V} \tag{8}$$

$$F_1 = \frac{V}{N}R \tag{9}$$

5. Evaluation

In this section, we set up simulations to verify the correctness of the analytical model and evaluate the performance of DHL worm. First, we initialize some simulation parameters based on the BT protocol. Then, we observe the effect of worm parameters on worm propagation. Finally, we compare the performance of DHL worm with that of topological worm.

5.1. Simulation Settings

To increase reality, we select some parameters from a real-world BT network for simulation settings. As discussed in section 2.2, the default number of return peers is 50. We can easily learn the size of a swarm from the response of *HTTP GET* request. Usually, the swarm sizes range from 5,00 to 2,500, where peers share popular files.

We generate several topologies using BRITE [1], a topology generator. Their sizes are 2000 and 5000 respectively, and average degree, denoted as D, is 50. In our simulations, we set only one initiator. An infected peers will not be active any more after launching an attack. When an infected peer requests a tracker, the tracker randomly selects a subset of peers to return.

5.2 Simulation of Breadth-First Propagation

Using the breadth-first propagation model, we first compute a minimal number of infected peers at hop 1 for a given ratio of infected peers. Then, we test whether the given ratio is reached in a simulation by choosing such a number.

5.2.1 Effect of Swarm Size

We study the minimal number of infected peers at hop 1 by varying the swarm size. We set N ={2000, 4000, 6000, 8000}. Assume the ratio of vulnerable peers is 0.8, namely, V ={1600, 3200, 4800, 6400}. Further assume $R = 50$, TTL ={2, 3}, and $INF = 0.9$. In Figure 3, we present the numerical results to F_1 under a guaranteed INF. Clearly, F_1 increases along with N increases. We should notice that, for a given INF, when TTL increase from 2 to 3, F_1 drops dramatically from 359 to 10, for $N = 8000$. As can be seen in Figure 4, using F_1 gained through the model, INF will achieve the expected value in simulations.

5.2.2 Effect of Vulnerable Peer Ratio

We further show the impact of the ratio of vulnerable peers. To spread the worm fast, we set $TTL = 2$. We also set N ={1000, 2000, 3000, 5000, 10000}, $INF = 0.9$ and $R = 50$. The ratio of vulnerable peers ranges from 0.1 to 1.0. From Figure 5, we know that F_1 increases slightly when the ratio of vulnerable peers increases. A very exciting simulation result is that, if we select the lowest F_1, INF will only increase about 0.05 when the ratio of vulnerable peers changes from 0.1 to 1, which is shown in Figure 6. That means, for a small TTL, we can maintain a statable infection rate using almost unchanged parameters, even though the ratio of vulnerable peers is unknown or has dramatically changed. Although INF's change is minor, it can not be ignored when N is large. Thus, we are interested in keeping INF stable. Clearly, a stable F_1 may lead to a stable INF. We assume that F_1 will become more stable, if R is set higher. The reason is that, $F_1 = \log_{\sim p} \sim P_1$, F_1 will be less sensitive to P_1's change if $\sim p$ is smaller, for $0 <\sim p < 1$ and $0 <\sim P_1 < 1$. Since $\sim p = 1 - \frac{R}{N}$,

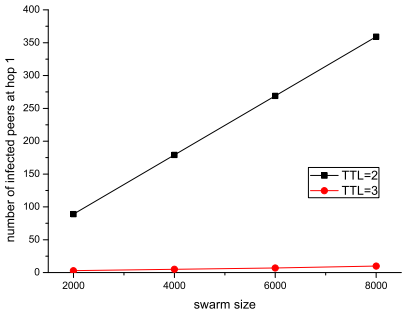

Figure 3: The number of infected peers at hop 1 vs the swarm size

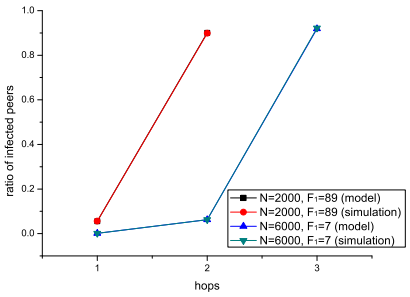

Figure 4: The ratio of infected peers vs the swarm size

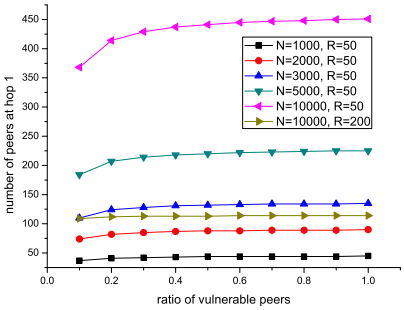

Figure 5: The number of infected peers at hop 1 vs the ratio of vulnerable peers

increasing R will leads to a smaller $\sim p$. As we might expect, both F_1 and INF show only a little change when R reaches 200.

5.3 Simulation of Depth-First Propagation

Using the depth-first propagation model, we first obtain a minimal value of TTL for a given ratio of infected peers. We suppose that the initiator infects no more that 50 peers directly. Then, we test whether the given ratio is reached in a simulation by selecting such a value.

5.3.1 Effect of Swarm Size

We study the impact of swarm size on worm propagation. We set N ={500, 1000, 1500, 2000, 2500}, INF ={0.3, 0.6, 0.9}. Suppose the ratio of vulnerable peers is 0.8, and R is 50. Figure 7 indicates that, for $INF = 0.3$, TTL maintains when N increases. But, for $INF = 0.9$, TTL increases from 2 to 3 when N increases from 500 to 1000. To verify our model, we choose N ={1000, 2000} and $TTL = 3$, and then observe INF in the simulation. As can be seen in Figure 8, simulation and numerical results are almost overlapped. This Figure also shows that all vulnerable peers are infected within 3-hop propagation, though the expected INF is 0.9.

5.3.2 Effect of Vulnerable Peer Ratio

We then show the effect of vulnerable peer ratio on parameter settings. We set N ={1000, 2000, 3000, 5000, 10000}, $INF = 0.9$ and $R = 50$. The ratio of vulnerable peers ranges from 0.1 to 1.0. From Figure 9, we can see that TTL decreases when the ratio of vulnerable peers increases. For $N = 10000$, TTL drops to 4 when the ratio goes up to 0.2, and then drops to 3 when the ratio reaches 0.5. If we choose $TTL = 3$ regardless of the ratio change, the ratio of infected peers will drops dramatically from 1.0 to 0.13, which can be seen in Figure 10. The reason is that low ratio of vulnerable peers may result in a low number of infected peers at hop 1, which may have a great effect on the ratio of infected peers. For example, when $\frac{V}{N} = 1.0$, F_1 is 50, but when $\frac{V}{N} = 0.1$, F_1 is only 5.

5.4 Performance Comparison

We compare the performance of DHL worm with that of neighbor-list worm through simulations. Our simulations are based on PeerSim [3], a Java based simulation simulator. In our simulations, we do not set the "time interval", which is mentioned in the *Assumption 4*. That means, DHL worm spreads with its fastest speed.

5.4.1 Effect of Average Degree

We study the effect of average degree on the worm propagation. Since we know little about the degree distribution of BT networks, we generate 6 topologies with either power-law or random distribution. All of them have 2000 peers. Their average degree D is 50. Also, we set $R = 50$, $\frac{V}{N} = 0.8$. From Figure 11, we can see that DHL worm can spread faster than topological worm. Within a time unit, DHL worm, which is unaffected by topology, compromises 94% vulnerable peers, while topological worm infects 81% and 58% vulnerable peers in power-law and random topologies, respectively. We also observe the number of hits per peer. Clearly, the higher the number is, the more network anomalies are. If the network anomalies are obvious, the worm can be easily detected. Figure 12 indicates that, the numbers caused by topological worm are 66 and 39 in different topologies. On the other hand, the number caused by DHL worm is about 27. Though DHL worm is effective, the fast spread of that worm may cause a DDoS attack to the tracker. To avoid detection, we need to to control the

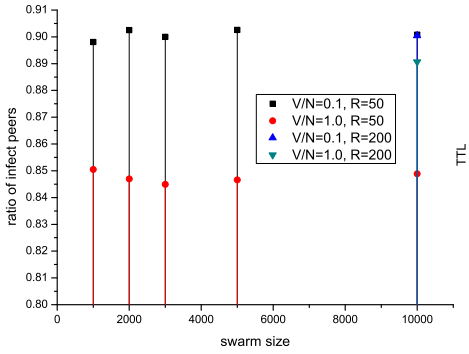

Figure 6: The ratio of infected peers vs the swarm size

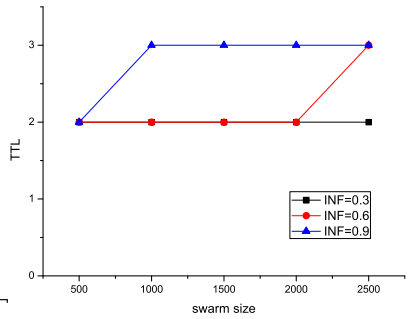

Figure 7: TTL vs the swarm size

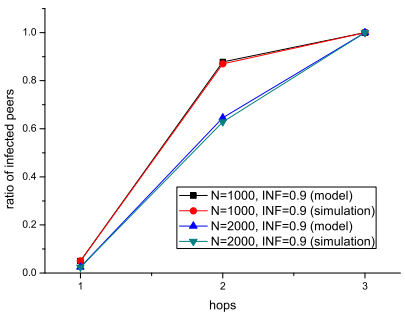

Figure 8: The ratio of infected peers vs hops for the swarm size is 2000

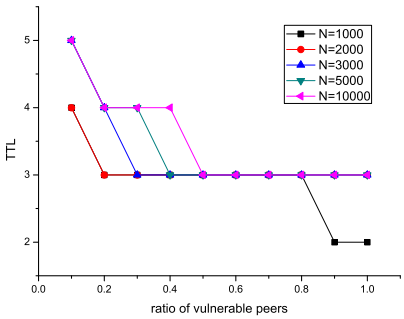

Figure 9: TTL vs the ratio of vulnerable peers

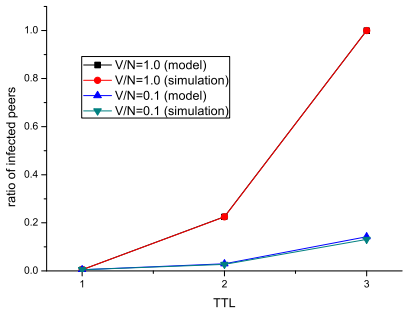

Figure 10: The ratio of infected peers vs hops for the swarm size is 10000

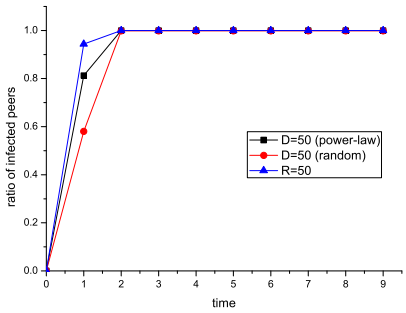

Figure 11: The ratio of infected peers vs time

worm propagation speed and set an appropriate TTL.

5.4.2 Effect of Vulnerable Peer Ratio

We then show the impact of vulnerable peer ratio. Again, we generate 2 topologies with a size of 5000. Their average degrees are 20. We set $R = 50$. The ratios of vulnerable peers are 0.1 and 0.9. As shown in Figure 13, the worm propagation slows down when the ratio of vulnerable peers decreases from 0.9 to 0.1. A very interesting observation is that topological worm cannot infect all the peers when the ratio of vulnerable peers drops to 0.1. The reason is that some vulnerable peers are isolated by invulnerable peers. For the same reason, we can use the software diversity of the client to contain topological worm. However, DHL worm does not have such a problem, because their potential victims are randomly selected.

6. Related Work

In this section, we provide a brief review of modeling work on P2P worms. Previous studies adopt epidemic model to study factors affecting P2P worm propagation, emphasizing on propagation process and environment.

Propagation process: To understand how a worm duplicates itself from one host to another, we could use host state transition to describe the propagation process of the worm. Two states, Susceptible and Infected, are considered in a simple passive P2P worm model [5]. Susceptible is the state that peers are vulnerable for worm; Infected is the state that peers have been infected and start to propagate. Two additional states, Exposed and Recovery, are added in SEI and SEIR models [11]. Exposed is the state that peers have downloaded one or more infected files, but have not executed them; Recovery stands for peers have installed patch and no longer infect other peers. TF-SEI model considers a Quarantine state which stands for the vulnerable peers are immunized by anti-worms and no longer susceptible to infection. Both on-line/off-line behaviors are discussed in [10]. The case that some peers become infected, but never detect the problem and not take any actions to remove the viruses is also considered in that model.

Propagation environment: P2P worm may behavior differently when it propagates in different P2P applications.

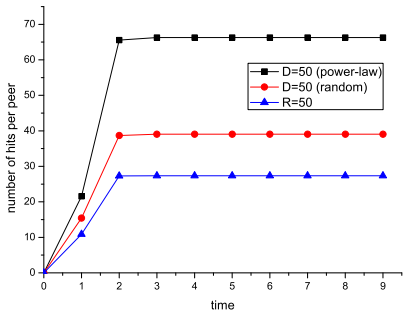

Figure 12: The number of hits per peer vs time

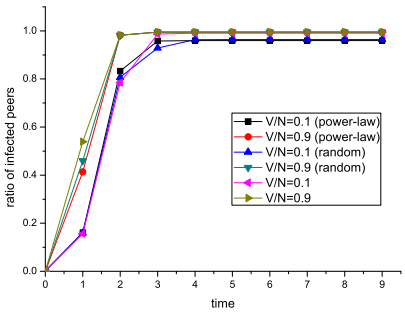

Figure 13: The ratio of infected peers vs time

In [9], the authors argue that the assumption [10] that a vulnerable peer can be infected by any of the infected ones is not true in Gnutella like networks since the potential victims for an infected peer are limited to TTL hops away from it and not the entire P2P network. To address the problem, the authors quantify the average number of peers within TTL hops from any given peer and incorporate the neighborhood information into the final model for malware spread.

7. Conclusion

In this paper, we first design a novel worm, called Dynamic Hit-List (DHL) worm, in BT networks. The worm finds new victims and propagates itself by requesting a tracker to build a dynamic hit list. Then, we formulate a mathematical model to investigate the impact of each worm parameter (e.g. TTL, number of returned peers from a tracker, etc.) on the DHL worm propagation, to be either breadth-first or depth-first propagation. The analytical results provide insights of the worm design in choosing parameters that enable the worm to stop itself after compromising a large fraction of vulnerable peers in the network. We finally evaluate the performance of DHL worm through simulations. The simulation results verify the correctness of our model and show the effectiveness of the worm.

References

[1] I. M. A. Medina, A. Lakhina and J. Byers. *BRITE*. Boston University, 2.0 edition.

[2] M. Engle and J. I. Khan. Vulnerabilities of p2p systems and a critical look at their solutions. Technical report, Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, 2006.

[3] G. P. Jesi. *PeerSim HOWTO: Build a topology generator for PeerSim 1.0*, 1.0 edition, 2006.

[4] N. Khiat, Y. Carlinet, and N. Agoulmine. The emerging threat of peer-to-peer worms. In *MonAM'06: Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation*, 2006.

[5] J. Ma, X. Chen, and G. Xiang. Modeling passive worm propagation in peer-to-peer system. In *CIS'06: Proceedings of the International Conference on Computational Intelligence and Security*, pages 1129–1132, 2006.

[6] J. Ma, G. M. Voelker, and S. Savage. Self-stopping worms. In *WORM '05: Proceedings of the 2005 ACM workshop on Rapid malcode*, pages 12–21, 2005.

[7] P. K. Manna, S. Chen, and S. Ranka. Exact modeling of propagation for permutation-scanning worms. In *INFOCOM'08: Proceedings of the IEEE International Conference on Computer Communications*, 2008.

[8] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM'04: Proceedings of the Conference of the Special Interest Group on Data Communications*, 2004.

[9] K. Ramachandran and B. Sikdar. Modeling malware propagation in gnutella type peer-to-peer networks. In *IPDPS06: Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, page 447, 2006.

[10] R. Thommes and M. Coates. Epidemiological modeling of peer-to-peer viruses and pollution. In *INFOCOM06: Proceedings of the IEEE International Conference on Computer Communications*, 2006.

[11] Y. Yao, X. Luo, and S. Ai. Research of a potential worm propagation model based on pure p2p principle. In *ICCT 06: Proceedings of the International Conference on Communication Technology*, pages 1–4, 2006.

[12] W. Yu, C. Boyer, and D. Xuan. Peer-to-peer system-based active worm attacks: Modeling and analysis. In *ICC05: Proceedings of the IEEE International Conference on Communications*, 2005.

[13] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, and S. Chien. A first look at peer-to-peer worms: Threats and defences. In *IPTPS'05: Proceedings of the International Workshop on Peer-To-Peer systems*, pages 24–35, 2005.