Modeling and Defending against Adaptive BitTorrent Worms in Peer-to-Peer Networks

JIAQING LUO, University of Electronic Science and Technology of China
BIN XIAO, QINGJUN XIAO, and JIANNONG CAO, Hong Kong Polytechnic University
MINYI GUO, Shanghai Jiao Tong University

BitTorrent (BT) is one of the most common Peer-to-Peer (P2P) file sharing protocols. Rather than downloading a file from a single source, the protocol allows users to join a *swarm* of peers to download and upload behavior investigation, (2) worm propagation modeling, and (3) worm detection and containment.

Worm behavior investigation needs to analyze strategies adopted by worms for target finding and detection evasion. Traditionally, P2P worms can be passive worms [Chen and Gray 2006], waiting for connections to propagate themselves. Though the abnormality of passive worms is hard to detect, they propagate at a relative low speed, and thus cause limited damage to the Internet [Tang et al. 2009]. Active topological P2P worms can search local information (i.e., peer's neighbor list) to find next victims [Engle and Khan 2006; Khiat et al. 2006; Zhou et al. 2005]. These worm can be very efficient, because they do not need to probe random IP addresses, which generates high rates of failed connections. However, it could be technically hard to obtain local information in various P2P configurations and implementations. The software diversity may prevent the neighbor list reading from different peers. This might be one of reasons why such a worm has not become widespread. To be stealthy, self-stopping worms [Ma et al. 2005] can stop themselves after compromising a large fraction of vulnerable peers. Yet, the largely compromised peers may be outdated in a highly dynamic P2P environment. Nowadays, the high penetration of P2P protocols and implementations has provided the worm writers chances to design some powerful worms that are previously unknown. The study of worm behavior remains to be an important aspect of the worm analysis.

Worm propagation modeling is to study how network/worm parameters affect the worm propagation. The epidemic model is widely used to characterize the growth of peers infected by P2P worms [Briesemeister et al. 2003; Ma et al. 2006; Thommes and Coates 2006; Yao et al. 2006]. The fluid model can depict the arrival and departure behavior of peers for the BT performance analysis [Guo et al. 2005; Qiu and Srikant 2004]. P2P worms behave differently in different P2P networks, because network configurations and settings (e.g., network size, uploading/downloading bandwidth, peer arrival/departure rate, etc.) can greatly affect how fast and how large a worm duplicates itself. Thus, there is no completely universal model for all P2P worms.

Worm detection and containment aim to find worm activities and minimize their damage to a P2P network. The self-defense infrastructure approaches [Douglas et al. 2008; Freitas et al. 2007; Zhou et al. 2005, 2006] try to build secured P2P architectures to prevent or isolate P2P worms. However, they may jeopardize the fairness of uploading/downloading policy in P2P networks. The disadvantages of worm-anti-worm [Yao et al. 2008] lies in the fact that it consumes a lot of bandwidth and raises some illegal issues. Though the feedback control [Wu and Feng 2006] can efficiently detect random scanning worms, it is difficult to handle P2P worms which can blend worm traffic into the normal traffic in a P2P network.

In this article, we present a powerful BT worm, called *Adaptive BitTorrent worm* (A-BT worm), to identify potential security problems in P2P networks. Such a worm finds new victims by sending forged requests to a tracker. To reduce its abnormal behavior, it sends out only one forged request within a defined time window. By adjusting the time window size, it can adaptively control its propagation speed. The A-BT worm is a potential threat to BT-like networks for it has the following properties.

—*Timeliness*. it has the ability to locate the most recent active peers even in a highly dynamic BT network.
—*Feasibility*. it is easy to be implemented by taking advantages of the high penetration of the BT protocol.
—*Adaptivity*. it can significantly reduce its abnormal behavior by adaptively adjusting its propagation speed.

We build a hybrid model by combining the fluid model with the epidemic model to estimate the damage caused by the worm. The fluid model evaluates the number of peers that could be infected in a swarm. The epidemic model measures the number of peers that have been infected. Our model shows the effect of network parameters on the worm propagation.

We propose a statistical method to automatically detect the worm from the tracker by estimating the variance of the time intervals of requests. A normal peer usually requests at a stable interval, while an infected peer may request at a variable interval. To slow down the spread of the worm, we design a strategy in which the tracker returns a biased list of peers (e.g., more secured peers) when receives a request.

Our simulation results show that our hybrid model precisely depicts the worm damage (number of infected peers in a swarm). The A-BT worm can propagate as fast as the topological P2P worm, more importantly, it is unaffected by the P2P topology. Our statistical method is effective to detect the worm, though the worm can minimize its abnormal behavior.

The rest of the article is organized as follows. Section 2 gives a brief introduction to the BT protocol and shows the details of the peer-to-tracker communication. Section 3 describes how the A-BT worm propagates itself and adaptively adjusts its propagation speed. In Section 4, we builds a hybrid model to measure the worm damage. Section 5 proposes a statistical method and a safe strategy to detect and contain the worm. Section 6 validates the hybrid model through simulations, and compares the A-BT worm with the topological P2P worm. Section 7 reviews the work related to P2P worms. Finally, Section 8 concludes this article.

2. PROTOCOL OVERVIEW AND DETAILS

In this section, we first give a brief introduction of the BT protocol and then show some details of the peer-to-tracker communication.

2.1. Overview of the BT Protocol

BitTorrent (BT) is one of the most common Peer-to-Peer (P2P) file sharing protocols, and it has been estimated that it accounts for more than roughly 27%–55% of all Internet traffic as of February 2009 [Ernesto 2009]. Programmer Bram Cohen designed the BT protocol in April 2001 and released a first implementation on July 2, 2001. The protocol distributes a large file without the heavy load on the source computer and network. Rather than downloading a file from a single source, the protocol allows users to join a *swarm* of peers to download and upload from each other simultaneously. The protocol can also work as an alternative method to distribute data and can work over systems with low bandwidth so even small computers, like mobile phones, are able to distribute files to many recipients. There have been numerous BT clients available for a variety of computing platforms.

A BT network commonly consists of trackers, seeders and leechers. A *tracker* keeps track of peers who are participating in the process of downloading and/or uploading a particular file, and makes this information available to others who want to download the file. Peers in the BT network are either *seeders* who have a complete file and are willing to serve it to others, or *leechers* who are still downloading the file and are willing to serve the pieces that they already have to others. Before joining a *swarm* which is a set of peers sharing a particular file, a peer firstly downloads a .torrent file from a Web server, which contains a URL list of trackers and other information related to the sharing file. After that, the peer makes an HTTP GET request to a tracker known from the URL list. Upon receiving the request, the tracker randomly returns a subset of peers sharing the file. Finally, the peer attempts to initiate TCP connections with the returned peers, which then become its *neighbors*. Files in the BT network consists

```
http://bandit.ukb-kvcd.com:5600/announce?info_hash=%08%E1Y%20%AB%16%9C%B1%
AC%90q%5F%2DL%D9mL%D6%D1f&peer_id=%2DBC0052%2D100000000099&port=
8099&natmapped=1&localip=202.125.210.212&port_type=wan&uploaded=0&downloaded=
0&left=756792781&numwant=10&compact=1&no_peer_id=1 &key=3156 &event=started
```

Fig. 1. An example of the HTTP GET request.

of pieces (32-256KB in size) which in turn consist of blocks (16KB in size). A block is a portion of data that a peer requests from neighbors.

2.2. Peer-to-Tracker Communication

To exploit the BT protocol for P2P worm design, we are interested in the details of the communication between a peer and a tracker. The *HTTP GET* request sent by a peer contains a tracker's URL and some CGI parameters. URL and parameters are separated by "?". Parameter and value are separated by "=". Different parameters are separated by "&". Figure 1 illustrates an example of a request:

Note that only the first three parameters are indispensable. *info_hash*, is the hashed value of the information field in a .torrent file. A tracker can classify peers into different swarms according to this parameter. *peer_id* is the ID of a peer and parameter *port* defines its port used for file sharing. Another useful parameter, *numwant*, indicates the number of peers to be returned from a tracker to a requesting peer. Its default value is 50, but it could be much larger. For example, it is 200 in BitComet, which is a popular BT client.

After receiving a request, the tracker sends back a response that includes: a list of active peers (*peer_id*, *IP* and *port*), number of seeders, number of leechers, and time interval between two requests. The sum of seeders and leechers is the size of a swarm, showing the total number of peers sharing a particular file. The time interval between two requests is 20 minutes. If the tracker doesn't receive a request from a peer within 30 minutes, it will assume that the peer has left the swarm, and delete the corresponding record.

An interesting observation is that the tracker distinguishes different peers depending on their *IP* and *port* rather than *peer_id*. In other words, a tracker will add a new record only if it receives a request containing a new pair of *IP* and *port*.

3. A-BT WORM DESIGN

In this section, we describe the strategies used by the A-BT worm to find next targets and reduce abnormal behavior.

3.1. Target Finding Strategy

An A-BT worm finds potential victims in a BT network by mimicking a normal client requesting a tracker. Figure 2 shows that peers P_a, and P_b sharing file 1 are in swarm A. Peers P_c and P_d sharing file 2 and 3 are in swarm B and C, respectively. Assume that P_a and P_b are vulnerable. Figure 2 illustrates the propagation of the A-BT worm in swarm A, which contains the following steps.

(1) The peer who initiates the worm propagation, called *initiator*, downloads a .torrent file from a web server (e.g., a .torrent file related to file 1), which contains a URL list of trackers.
(2) The initiator contacts with a tracker known from the list by sending an *HTTP GET* request.
(3) The tracker responds to the request by returning a list of active peers who are sharing file 1. For example, it returns a list containing information of P_a, such as *IP*, *port*, and *peer_id*.

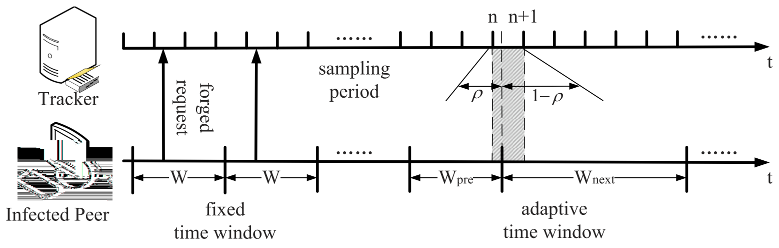


Fig. 3. The adaptive speed control for the A-BT worm.

decreases the time window size linearly. Otherwise, it increases the time window size exponentially. Thus, we have:

$$W_{next} = W_{pre} - \phi T_{life}, \qquad if\ A_{pre} < A_{th} \tag{1}$$

$$W_{next} = \varphi W_{pre}, \qquad if\ A_{pre} \geq A_{th}, \tag{2}$$

where T_{life} is the swarm lifespan [Guo et al. 2005], which can be viewed as a constant for a particular swarm. ϕT_{life} represents the decreasing amount of the time window size for $0 \leq \phi \leq \frac{W_{pre}}{T_{life}}$. φ is an increasing ratio of the time window size for $\varphi \geq 1$. Note that there is no requirement for the synchronization between infected peers and the tracker.

From Figure 3, we can see that the n-th sampling period is separated into two parts ρ and $(1 - \rho)$ for $0 \leq \rho \leq 1$. The probability that the infected peer sends a forged request within the n-th sampling period β is $\frac{\rho}{W_{pre}} + \frac{1-\rho}{W_{next}}$. During the interval $[n + 1, n + W_{next} - 1]$, we have $\beta = \frac{1}{W_{next}}$.

The A-BT worm is different from the R-BT worm in that the probability β for A-BT worm is determined by Equations (1) and (2), rather than a constant. In fact, the R-BT worm can be regarded as a special case of the A-BT worm with $\phi = 0$ and $\varphi = 1$.

4. A-BT WORM MODELING

In this section, we firstly describe some terminology and notations used in our model. We then formulate a hybrid model to display the propagation of the A-BT worm in a swarm.

4.1. Terminology and Notations

Before describing our model, we define some basic terminology and notations. Figure 4 shows a complete classification tree of vulnerable peers during an infection. The vulnerable peers are classified into two categories: *infected peers*, which are vulnerable and infected, and *uninfected peers*, which are vulnerable but not infected.

The *infected peers* can be further classified into two categories: *effective peers*, which are active propagating worms, and *ineffective peers*, which stop propagating worms. This classification is based on an assumption that infected peers who leave the swarm are not effective any more. Such an assumption can make the swarm size predictable, and thus, simplify our model. All effective peers will eventually leave the swarm and become ineffective. Based on whether they have a completed file or not, we further group effective peers into two subcategories: *effective seeders* and *effective leechers*.

The *uninfected peers* can be classified into two categories according to whether or not they have been returned in a request: *returned peers* and *unreturned peers*. A returned peer can be either a seeder or a leecher, called *returned seeder* or *returned leecher*, respectively.

Table I. Parameters Used in the A-BT Worm Model

$L(t)$	number of leechers at time index t
$S(t)$	number of seeders at time index t
T	length of a sampling period
$J[n]$	number of leechers that join a swarm within the n-th sampling period
$B[n]$	number of leechers that become seeders within the n-th sampling period
$D[n]$	number of seeders that depart from a swarm within the n-th sampling period
$L[n]$	number of leechers in a swarm at time index n
$S[n]$	number of seeders in a swarm at time index n
$N[n]$	total number of peers in a swarm at time index n
$R_L[n]$	number of returned leechers within the n-th sampling period
$R_S[n]$	number of returned seeders within the n-th sampling period
$E_L[n]$	number of effective leechers at time index n
$E_S[n]$	number of effective seeders at time index n
$I[n]$	total number of infected peers at time index n
$\Delta L[n]$	changes of leechers in a swarm within the n-th sampling period
$\Delta S[n]$	changes of seeders in a swarm within the n-th sampling period
$\Delta E_L[n]$	changes of effective leechers within the n-th sampling period
$\Delta E_S[n]$	changes of effective seeders within the n-th sampling period
$\Delta I[n]$	increasing number of infected peers within the n-th sampling period
n_0	initial time index of worm attack
λ_0	initial value of peer arrival rate
τ	attenuation parameter of peer arrival rate
μ	uploading bandwidth
η	mean upload utilization of leechers
γ	rate at which seeders leave a swarm
α	probability that a peer to be vulnerable
ω	number of returned peers in a request
β_i	probability that the i-th infected peer generates a forged request within the n-th sampling period

Following the idea of the fluid model proposed in [Guo et al. 2005; Qiu and Srikant 2004], we have:

$$J[n] = \int_{nT}^{(n+1)T} \lambda_0 e^{-\frac{t}{\tau}} dt, \tag{3}$$

$$B[n] = \int_{nT}^{(n+1)T} \mu\big(\eta L(t) + S(t)\big) dt, \tag{4}$$

$$D[n] = \int_{nT}^{(n+1)T} \gamma S(t) dt, \tag{5}$$

$$\Delta L[n] = J[n] - B[n], \tag{6}$$

$$\Delta S[n] = B[n] - D[n]. \tag{7}$$

By Equation (6) and (7), we can obtain the number of leechers $L[n]$ and the number seeders $S[n]$. We focus here on the physical meaning of each parameter. The detailed resolution of the fluid model can be found in Guo et al. [2005]. The total number of peers at time index n is given by:

$$N[n] = L[n] + S[n]. \tag{8}$$

We now derive how $E_L[n]$, $E_S[n]$ and $I[n]$ change over time index n. In the following we compute the amounts, $\Delta E_L[n]$, $\Delta E_S[n]$ and $\Delta I[n]$, by which they change respectively over a small sampling period after time index n. This will give us a set of difference equations that together characterize the A-BT worm propagation.

— $\mathbf{\Delta I[n]}$. The number of uninfected peers in a swarm at time index n is $\alpha N[n]-E_L[n]-E_S[n]$. Since each peer in the swarm, no matter infected or not, has an equal chance to be returned in a request, the probability that an uninfected peer is not returned to any infected peer within the n-th time period is $\left(1-\frac{min\{N[n],\omega\}}{N[n]}\right)^{\sum_{i=1}^{E_L[n]+E_S[n]}\beta_i}$. Whenever an uninfected peer is returned, it will be infected immediately. By the epidemic model, $\Delta I[n]=\left(\alpha N[n]-E_L[n]-E_S[n]\right)\Bigg(1-\left(1-\frac{min\{N[n],\omega\}}{N[n]}\right)^{\sum_{i=1}^{E_L[n]+E_S[n]}\beta_i}\Bigg)$.

— $\mathbf{R_L[n]}$. The fraction of returned uninfected leechers in all returned uninfected peers is proportional to that of uninfected leechers in all uninfected peers. Hence, $R_L[n]=\frac{\alpha L[n]-E_L[n]}{\alpha N[n]-E_L[n]-E_S[n]}\Delta I[n]$.

— $\mathbf{R_S[n]}$. The ratio of returned uninfected seeders in all returned uninfected peers is $\frac{\alpha S[n]-E_S[n]}{\alpha N[n]-E_L[n]-E_S[n]}$. We have $R_S[n]=\frac{\alpha S[n]-E_S[n]}{\alpha N[n]-E_L[n]-E_S[n]}\Delta I[n]$.

— $\mathbf{\Delta E_L[n]}$. Recall that all returned leechers directly become effective leechers. This increases the number of effective leechers by $R_L[n]$. On the other hand, some effective leechers may complete their download and become effective seeders instantly. There are totally $B[n]$ leechers who become seeders, and the fraction of effective leechers in all leechers is $\frac{E_L[n]+R_L[n]}{L[n]}$. Hence, the reducing number of effective leechers is $\frac{E_L[n]+R_L[n]}{L[n]}B[n]$. Combining these two numbers and representing the gross change, we have $\Delta E_L[n]=R_L(n)-\frac{E_L(n)+R_L(n)}{L(n)}B[n]$.

— $\mathbf{\Delta E_S[n]}$. $R_S[n]$ returned seeders instantly become effective seeders. As discussed above, there are $\frac{E_L[n]+R_L[n]}{L[n]}B[n]$ effective leechers who become effective seeders. Whenever an effective seeder leaves a swarm, it becomes ineffective. Within the nth sampling period, there are $D[n]$ seeders that leave the swarm, and the fraction of effective seeders is $\frac{E_S[n]+R_S[n]}{S[n]}$. Thus, there are $\frac{E_S[n]+R_S[n]}{S[n]}D[n]$ newly ineffective peers. Combining these three numbers to represent the gross change, we have $\Delta E_S[n]=R_S[n]+\frac{E_L[n]+R_L[n]}{L[n]}B[n]-\frac{E_S[n]+R_S[n]}{S[n]}D[n]$.

Thus, we have the following equations:

$$\Delta I[n] = \left(\alpha N[n]-E_L[n]-E_S[n]\right)\Bigg(1-\left(1-\frac{min\{N[n],\omega\}}{N[n]}\right)^{\sum_{i=1}^{E_L[n]+E_S[n]}\beta_i}\Bigg), \tag{9}$$

$$R_L[n] = \frac{\alpha L[n]-E_L[n]}{\alpha N[n]-E_L[n]-E_S[n]}\Delta I[n], \tag{10}$$

$$R_S[n] = \frac{\alpha S[n]-E_S[n]}{\alpha N[n]-E_L[n]-E_S[n]}\Delta I[n], \tag{11}$$

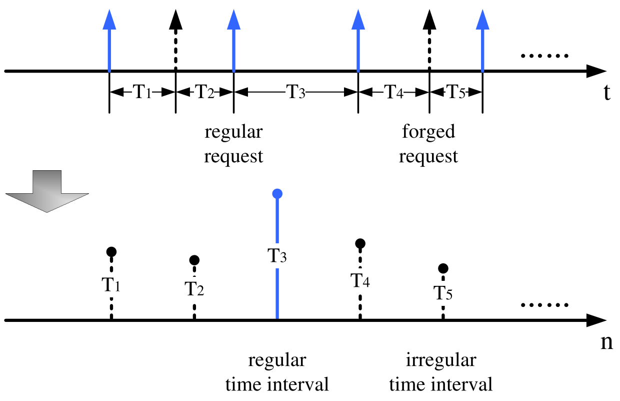


Fig. 6. An example of the sample sequence.

$$\Delta E_L[n] = R_L[n] - \frac{E_L[n] + R_L[n]}{L[n]} B[n], \tag{12}$$

$$\Delta E_S[n] = R_S[n] + \frac{E_L[n] + R_L[n]}{L[n]} B[n] - \frac{E_S[n] + R_S[n]}{S[n]} D[n]. \tag{13}$$

We should note that, in Equation (9), the parameter β_i is essential to the worm speed, which has been discussed in Section 3.2. Finally, we add the incremental variables and get $I[n+1] = I[n] + \Delta I[n]$, $E_L[n+1] = E_L[n] + \Delta E_L[n]$. The boundary conditions for the set of equations above are: $L[0] = E_S[n_0] = 0$, $S[0] = N[0] = E_L[n_0] = I[n_0] = 1$.

5. A-BT WORM DETECTION AND CONTAINMENT

Most existing defense methods may fail to detect and contain the A-BT worm for the following reasons. The self-defense infrastructure methods [Douglas et al. 2008; Freitas et al. 2007; Zhou et al. 2005, 2006] cannot slow down the worm because the worm is not affected by the P2P topology. The feedback control approach [Wu and Feng 2006] is unable to detect the worm since the worm can adaptively adjust its speed. Thus, there is clearly a need for finding a new way to defend against the A-BT worm. In this section, we employ a statistical method to automatically detect the worm from the tracker by investigating the variance of time intervals of requests.

The variance analysis, including the sample mean analysis, can be used to measure the spread of samples, which has been widely used in many areas, such as seismic detection and investment management in business. To estimate the request generation rate of a peer, which may be greatly affected by the worm, we sample time interval between two consecutive requests of the peer. As shown in Figure 6, the samples represented as a chronological sequence $\{X[n]\}$ can be viewed as discrete-time signals. In normal cases, the amplitude of signals is stable, because regular requests are generated at a regular time interval (20 minutes), and the "background noise" (e.g., network delay) has little effect on the signal to noise ratio (SNR). However, when a peer is compromised by the worm, the amplitude of the signal varies a lot, because the forged requests may divide regular time intervals into some irregular short time intervals. By measuring the similarity of the signal on multiple peers, we can estimate the damage caused by the worm.

Assume that we have obtained M signals, and the signal length is N. The sample mean of M signals is defined by:

$$\overline{X}[n] = \tfrac{1}{M} \textstyle\sum_{m=0}^{M-1} X_m[n]. \tag{14}$$

We use the sample mean $\overline{X}[n]$ to denote a "reference" signal. The sample variance is given by

$$\begin{aligned} Var(X[n]) &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \left(X_m[n] - \overline{X}[n]\right)^2, \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X_m[n] - M \sum_{n=0}^{N-1} \overline{X}^2[n]. \end{aligned} \tag{15}$$

We transform the sample variance $Var(X[n])$ into the range of [0, 1]. Then, the normalized sample variance $Var'(X[n])$ can be written as

$$\begin{aligned} Var'(X[n]) &= \frac{Var(X[n])}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X_m^2[n]} \\ &= 1 - \frac{M \sum_{n=0}^{N-1} \overline{X}^2[n]}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X_m^2[n]}. \end{aligned} \tag{16}$$

When no attack occurs, $Var'(X[n])$ will be close to 0. When the worm is widespread, $Var'(X[n])$ will become bigger. Thus, we can set a threshold to detect the worm. We should point out that, in real cases, peers will request the tracker frequently, if they do not have sufficient neighbors. The threshold setting should be based on some statistical results. Some advanced techniques can also be applied for the worm detection, such as data mining, Fourier analysis, coherence analysis and wavelet analysis. With these techniques, we can identify not only worm activities but also infected peers.

For the worm containment, the tracker can collect some useful information on peers, such as the number of OS vulnerabilities and the version of the antivirus software installed. This information could be easily obtained, because most antivirus softwares provide the function of the vulnerability scanning. The tracker employs a safe strategy to returns a biased peer set, rather than a random peer set, in response to a request. The biased peer set, where peers have less OS vulnerabilities or have an up-to-date antivirus software, can reduce the probability that an uninfected peer is returned in a request. We believe that the safe strategy will not seriously affect the fairness of uploading/downloading strategy in the BT network, because it is used only when a worm attack is detected.

6. SIMULATION RESULTS

In this section, we firstly describe the environment and settings of our simulations. We then setup several simulations to verify our hybrid model and detection method.

6.1. Simulation Environment and Settings

The study of the worm propagation in a real-world BT network may cause some legal issues. We deploy our simulations on a powerful simulation platform developed by University of Electronic Science and Technology of China. This simulation platform has three components: (1) *tracker*: on one host, we setup a real-world tracker by using MyBT server 1.0, which is a software to setup a BT server. (2) *initial seeder*: on the same host, we install BitComet 0.93, which is a well-known BT download software, and

Table II. Simulation Settings of the A-BT Worm

File size	15.6MB (500 pieces)
Number of initial seeds $S[0]$	1
Normalized maximum upload bandwidth μ	0.01 (5 pieces)
Normalized maximum download bandwidth	0.03 (15 pieces)
Number of neighbors of each peer	10
Initial value of peer arrival rate λ_0	0.6
Attenuation parameter of peer arrival rate τ	300
Initial time index of worm attack n_0	150
Rate at which seeds leave a swarm γ	0.08
Probability that a peer to be vulnerable α	0.8
Number of returned peers in a request ω	10
Mean upload utilization of leechers η	1
Time interval between two normal requests	100
Time window size W	50

use the software to create a .torrent file. (3) *simulated peers*: on another host, we install PeerSim [Jesi and Patarin 2005], which is a Java-based discrete-event engine that can simulate hundreds of peers and observe their activities (e.g., arrivals, departures, pieces exchange), and implement the basic BT protocol (e.g., unchoking, rarest-first). Each simulated peer can communicate with the tracker and share real pieces with both initial seed and other simulated peers.

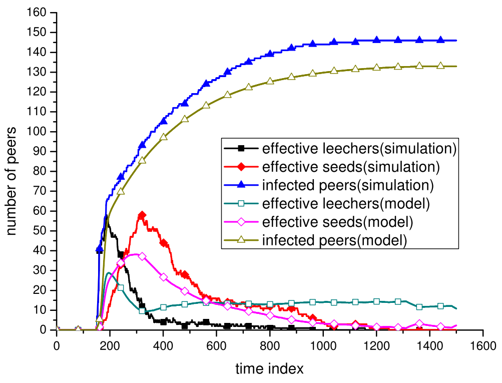
For simplicity, we do not consider free-ridering [Barbera et al. 2005] in our simulations. In other words, all peers are willing to contribute their upload bandwidth. We also do not strictly follow unchoking specified in the BT protocol. A peer simply chokes those neighbors who do not upload any data within a time interval. We believe that this change may have a very little effect on the results.

Since the piece transmission is resource consuming, we limit the maximum scale of our simulation to 200 peers. We define the sampling period (1 cycle in PeerSim) as the unit of time. Unless otherwise specified, the default settings in our simulations are shown in Table II.

In our simulations, we evaluate the changes in number of leechers $L[n]$, number of seeds $S[n]$, number of effective leechers $E_L[n]$, number of effective seeders $E_S[n]$, and number of infected peers $I[n]$.

6.2. Verification of the Hybrid Model

We compare the results of $E_L[n]$, $E_S[n]$ and $I[n]$ from Equations (12), (13), and (9) with that from simulations. Figure 7 shows that there exists a small gap between numerical results and simulation results. The peak of effective seeds in the simulation result is around 60, while that in the numerical result is less than 40. We guess that the gap is mainly caused by the uncertainty in the measurement of the complicated peer behavior in the BT network. We further measure the mean upload bandwidth utilization of leechers and seeders. From Figure 8, we can see that the upload bandwidth of seeds is fully utilized at the beginning, because the initial seed is called upon to do much of the serving when the swarm size is small. However, when more peers join the network, the upload bandwidth utilization of seeders drops dramatically from 1.0 to 0.27. On the other hand, the upload bandwidth utilization of leechers increases and arrives at its first peak 0.56 at time index 129, which indicates that P2P serving becomes very effective. We input the simulation results of $L[n]$ and $S[n]$ into the hybrid model, and recompute $E_L[n]$, $E_S[n]$ and $I[n]$. Figure 9 shows that the numerical results match with the simulation results very well.



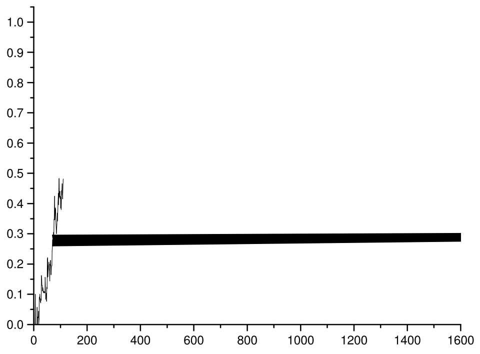


Fig. 7. Model validation using parameters obtained from the fluid model.

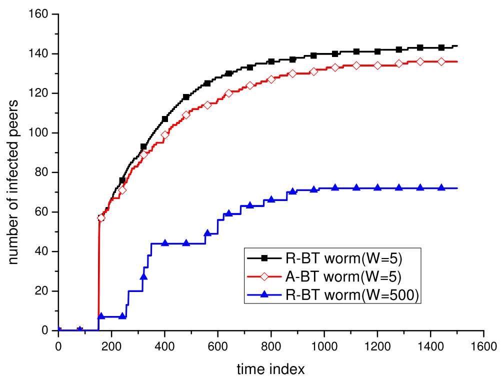


Fig. 11. Comparison between R-BT worm and A-BT worm.

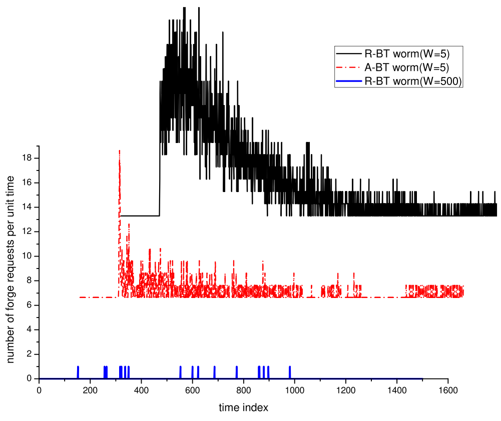


Fig. 12. Number of forged requests generated by worms.

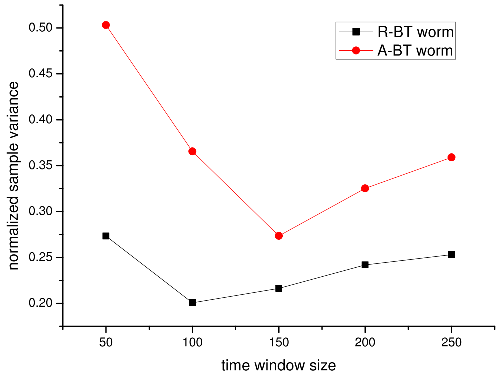


Fig. 13. Effect of time window size on worm detection.

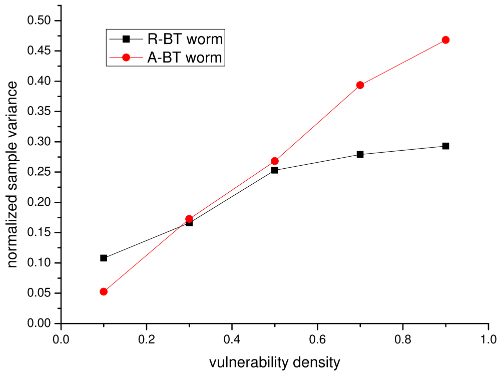


Fig. 14. Effect of vulnerability density on worm detection.

the vulnerable peers are infected. For the A-BT worm, we set the initial time window size to 5, the threshold rate of being accessed A_{th} to 0.4, and the increasing ratio of the time window size φ to 10. Recall that the time window size will increase exponentially if A_{pre} is larger than A_{th}. We then observe that the A-BT worm can be as efficient as the R-BT worm. Figure 12 shows the abnormal behavior of the two worms. We see that the forged requests generated by the A-BT worm are far less than that generated by the R-BT worm.

6.4. Effectiveness of the Detection Method

We firstly study the impact of worm parameters on the detection method. We set the time window size W to be one of $\{50, 100, 150, 200, 250\}$, the leaving rate γ to 0.002. For the A-BT worm, we set the increasing ratio of the window size φ to 5. We randomly select 20 peers from the swarm, and analyze the variance in the time intervals of requests. Figure 13 shows that $Var'(X[n])$ for R-BT worm drops from 0.273 to 0.201 when W increases from 50 to 100, and then goes up from 0.201 to 0.253. $Var'(X[n])$ for A-BT worm decreases from 0.503 to 0.274 when W increases from 50 to 100, and then increases to 0.359. When W increases, the forged request generation rate decreases. If W becomes too large, it may result in large variations in the time interval. That might be the reason why $Var'(X[n])$ of A-BT worm drops first and then goes up slightly. Even though the A-BT worm is “smart” enough to request the tracker at a very low rate, we can detect it from the tracker by setting an appropriate threshold for $Var'(X[n])$.

We then show the impact of network parameters on the detection method. We select the vulnerability density α from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Other parameters are the same as that in the previous simulation. From Figure 14, we see that $Var'(X[n])$ increases when the vulnerability density increases. We can also see that $Var'(X[n])$ for A-BT worm is much higher that for R-BT worm, when α is large. These results show that we can estimate the vulnerability density of the swarm by observing $Var'(X[n])$. We can take the safe strategy only when a large fraction peers are vulnerable for the worm attack.

7. RELATED WORK

P2P worms find targets and propagate themselves in P2P networks. To throttle against P2P worms, we need to investigate them in terms of: (1) target finding strategies, (2) worm modeling, and (3) worm containment.

Target finding strategies. P2P worms have various ways to locate potential victims in P2P networks. Three strategies are commonly used for target finding.

—*Passive*. Passive P2P worms, such as Gnuman worm [Vamosi 2001] and Banjamin worm [Singer 2002], attach themselves to shared files and wait for potential victims to discover and download them.

—*Topological*. Topological P2P worms propagate through P2P neighbors. These worms can be further classified into two categories: reactive and proactive. (1) *Reactive* worms, also called contagion worms, propagate parasitically along with normal communications. (2) *Proactive* worms achieve much faster propagation speed by directly connecting to other peers, whose addresses were cached after previous file requests and sharing.

—*Hit List*. Some open source P2P clients, such as Gnutella and BT, can be explored to generate a hit list. The modified clients can crawl P2P networks actively to discover as many0002Tc4.75over hem

detect P2P worms [Zhou et al. 2005]. Though simple it is, this method is hard to be implemented because the wide scale deployment of guardian nodes is resource consuming. (2) *Software diversity* can also be used to slow down P2P worms. In Verme [Freitas et al. 2007], all peers are divided into different groups, called *islands*, according to the type of vulnerabilities. To help peers select neighbors, D. McIlwraith, et al. [Douglas et al. 2008] employ a server, called di-jest server, to compute the distance of each peer pair which corresponds to the infection probability of a worm. Y. Zhou et al. [Zhou et al. 2006] give a model to study the effect of software diversity on P2P worm propagation. However, these methods are not practical, because software developers may not willing to provide several clients for the same protocol.
— *Worm-Anti-Worm*. The anti-worm spreads itself using the same mechanism as the malicious worm [Yao et al. 2008]. This method has some obvious limitations. One is that the spread of antiworm consumes a lot of bandwidth. The other is that most computer crime laws do not distinguish "worms" from "antiworms".
— *Feedback Control*. The feed back scheme [Wu and Feng 2006] slows down P2P worms by delaying a peer's request when the peer tries to make connections at a high rate. Unfortunately, in P2P networks, a normal peer may also attempt to connect a large number of peers for a high download rate. It is difficult to distinguish the abnormal behavior from the normal one.

8. CONCLUSION

In this article, we present a novel P2P worm, called *Adaptive BitTorrent worm* (A-BT worm), in BT-like networks, which propagates itself by requesting a tracker for vulnerable peers. We believe that such a worm could pose a vital threat to the P2P security for the following reasons: (1) it has the ability to locate most recent active peers; (2) it is easy to be implemented; (3) it can adaptively adjust its speed to reduce its abnormal behavior. By combining the fluid model with the epidemic model, we build a hybrid model to measure the worm damage. We also discuss the possible strategies to detect and contain the worm.

REFERENCES

Barbera, M., Lombardo, A., Schembra, G., and Tribastone, M. 2005. A markov model of a freerider in a bittorrent p2p network. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'05)*. Vol. 2. 985–989.

Briesemeister, L., Lincoln, P., and Porras, P. 2003. Epidemic profiles and defense of scale-free networks. In *Proceedings of the ACM CCS Workshop on Rapid Malcode (WORM'03)*.

Chen, G. and Gray, R. S. 2006. Simulating non-scanning worms on peer-to-peer networks. In *Proceedings of the 1st International Conference on Scalable Information Systems (INFOSCALE'06)*.

Douglas, M., Micael, P., and Evangelos, K. 2008. di-jest: Autonomic neighbour management for worm resilience in p2p systems. In *Proceedings of the International Symposium World of Wireless, Mobile and Multimedia Networks (WoWMoM'08)*. 1–6.

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

Ernesto. 2009. Bittorrent still king of p2p traffic. Torrentfreak.

Freitas, F., Rodrigues, R., Ribeiro, C., Ferreira, P., and Rodrigues, L. 2007. Verme: Worm containment in peer-to-peer overlays. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07)*.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. 2005. Measurements, analysis, and modeling of bittorrent-like systems. In *Proceedings of the Internet Measurement Conference. (IMC'05)*. 35–48.

Jesi, G. P. and Patarin, S. 2005. PeerSim HOWTO: Build a new protocol for the PeerSim 1.0 simulator. Peersim.surcefge.net.

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

Luo, J., Xiao, B., Liu, G., Xiao, Q., and Zhou, S. 2009. Modeling and analysis of self-stopping bt worms using dynamic hit list in p2p networks. In *Proceedings of the 5th International Workshop on Security in Systems and Networks (SSN'09)*.

Ma, J., Voelker, G. M., and Savage, S. 2005. Self-stopping worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM'05)*. ACM, New York, 12–21.

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

Qiu, D. and Srikant, R. 2004. Modeling and performance analysis of Bittorrent-like peer-to-peer networks. In *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM'04)*. 367–378.

Ramachandran, K. and Sikdar, B. 2006. Modeling malware propagation in gnutella type peer-to-peer networks. In *Proceedings of the Parallel and Distributed Processing Symposium*.

Singer, M. 2002. "Benjamin" worm plagues Kazaa. Tech. rep., siliconvalley.internet.com.

Tang, Y., Luo, J., Xiao, B., and Wei, G. 2009. Concept, characteristics and defending mechanism of worms. *IEICE Trans. Inf. Syst. E92-D,* 5, 799–809.

Thommes, R. and Coates, M. 2006. Epidemiological modeling of peer-to-peer viruses and pollution. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'06)*.

Vamosi, R. 2001. Gnutella worm: How to deal with it. http://www.zdnet.com/gnutella-worm-how-to-deal-with-it-3002084706/.

Wu, K. and Feng, Y. 2006. Proactive worm prevention based on p2p networks. *Int. J. Comput. Sci. Netw. Security, 6*.

Yao, Y., Luo, X., Gao, F., and Ai, S. 2006. Research of a potential worm propagation model based on pure p2p principle. In *Proceedings of the International Conference on Communication Technology (ICCT'06)*. 1–4.

Yao, Y., Wu, L., Gao, F., Yang, W., and Yu, G. 2008. A waw model of p2p-based anti-worm. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC'08)*. 1131–1136.

Zhou, L., Zhang, L., Mcsherry, F., Immorlica, N., Costa, M., and Chien, S. 2005. A first look at peer-to-peer worms: Threats and defenses. In *Proceeding of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*.

Zhou, Y., Wu, Z., Wang, H., Zhong, J., Feng, Y., and Zhu, Z. 2006. Breaking monocultures in p2p networks for worm prevention. In *Proceedings of the 5th International Conference on Machine Learning and Cybernetics (ICMLC'06)*.

Received March 2009; revised December 2009; accepted February 2010