Identity Attack and Anonymity Protection for P2P-VoD Systems

Mengwei Lu Patrick P.C. Lee John C.S. Lui
The Chinese University of Hong Kong, CS&E Department
{mwlu, pclee, cslui}@cse.cuhk.edu.hk

***Abstract*—As P2P multimedia streaming service is becoming more popular, it is important for P2P-VoD content providers to protect their servers identity. In this paper, we first show that it is possible to launch an *"identity attack"*: exposing and identifying servers of peer-to-peer video-on-demand (P2P-VoD) systems. The conventional wisdom of the P2P-VoD providers is that identity attack is very difficult because peers cannot distinguish between regular peers and servers in the P2P streaming process. We are the first to show that it is otherwise, and present an efficient and systematic methodology to perform P2P-VoD servers detection. Furthermore, we present an analytical framework to quantify the probability that an endhost is indeed a P2P-VoD server. In the second part of this paper, we present a novel architecture that can hide the identity and provide anonymity protection for servers in P2P-VoD systems. To quantify the protective capability of this architecture, we use the *"fundamental matrix theory"* to show the high complexity of discovering all protective nodes so as to disrupt the P2P-VoD service. We not only validate the model via extensive simulation, but also implement this protective architecture on PlanetLab and carry out measurements to reveal its robustness against identity attack.**

I. Introduction

Peer-to-peer video-on-demand (P2P-VoD) streaming is one of the most promising P2P applications [2]. The aim of this service is to provide users an almost instant access to a large number of videos that are stored in the servers of a P2P-VoD system. P2P-VoD is gaining popularity, for example, companies like PPLive and PPStream are providing such service and they can support a tens of thousands of concurrent users. However, it is also due to this popularity that P2P-VoD servers are often vulnerable to many security attacks. One of them is called the "*identity attack*", which is to discover the identity, e.g., IP addresses, of P2P-VoD servers.

The architecture of most P2P-VoD systems [2] can be briefly described as follows. A P2P-VoD system maintains a set of content servers which store all available movies. Furthermore, the system also has a set of trackers (or special nodes) to assist peers to discover available movies among others peers in the P2P network. To access a movie, a peer first determines whether the movie is available among other peers in the P2P network. If yes, then this peer can access the movie via other peers. If not, this peer will then connect to the content server. During the streaming service of the entire movie, this peer may retrieve video blocks either from (1) other peers which are watching this movie or have cached the video blocks of this movie, or (2) from the content server which stores the entire movie. To reduce the workload to the content server, peer always try to first retrieve video blocks among peers in the network. Only when the video block is not available or other peers are busy, then this peer will request the video block from the content server. The rationale for this video retrieval priority is to reduce the workload to the content server so that the P2P-VoD system an scale and support more users.

There are a number of justifications as to why one wants to discover the identity of content servers in a P2P-VoD system:

- A P2P-VoD company wants to discover content servers of its competitor so that it can easily launch network attacks (e.g., DDoS [4], [13], [15], low rate attacks [9]) on its competitor's servers so to degrade the streaming quality. Users of this victimized P2P-VoD system may abandon the service and opt for other P2P-VoD distributors.
- A law enforcement agency may want to identify the IP addresses of content servers that distribute any illegal movie so that have sufficient evidence to persecute the operators of such P2P streaming service.
- Owners of movies/audio records may want to discover the identity of P2P-VoD content servers and persecute the P2P-VoD operators if they find out that their copyrights are infringed.

So it is to the best interest of P2P-VoD content providers to understand the risk of exposing their content servers, and how to protect resources if identity leakage is deem possible.

The conventional wisdom is that it is technically difficult to discover servers in P2P-VoD systems. People argue that under the P2P paradigm, a peer changes connections with its neighbors from time to time, and it is difficult to distinguish whether a neighbor is a server, or simply another peer in the network. Another argument for the difficulty to identify content servers is that most of these systems are closed: users do not have access to the P2P-VoD source codes, so it is difficult to extract the semantic of the communication protocol and it is a challenge to identify content servers. The contributions of our paper are:

(1) We challenge this conventional wisdom by presenting a systematic methodology to identify content servers in P2P-VoD systems.

(2) The identification methodology *does not require access to the P2P-VoD's source code*, or *full examination of packets' payloads*. But rather, we use *capture-and-filter techniques* to identify content servers.

(3) We present analytical model to quantify the probability of

an endhost being a content server.
(4) We carry out experiment on realistic P2P-VoD systems, i.e., PPLive and PPStream, and validate the effectiveness of our identification algorithm.
(5) We propose a novel architecture which we call *shield-nodes* to counter this identity attack. We analytically show that it is computationally expensive to identify all shield nodes so to disrupt the P2P-VoD service.
(6) We carry out PlanetLab experiments to demonstrate the effectiveness of our protective architecture.

II. Detection Methodology

Let us present the methodology to identify content servers in a P2P-VoD system. We want to emphasize that the methodology we propose *does not require* one to have any knowledge of the source code of a P2P-VoD system, and it *does not require* one to examine the full payload of packets. Instead, we propose to monitor the behavior of end hosts, and to infer the statistical properties of such end host, e.g., the probability that this end host is a content server in a P2P-VoD system. After executing the detection algorithm for a particular movie, we will have the information of all suspicious IP addresses together with their probabilities of being a content server (Fig. 1). In what follows, we describe in detail our detection methodology as well as the statistical inference procedure.

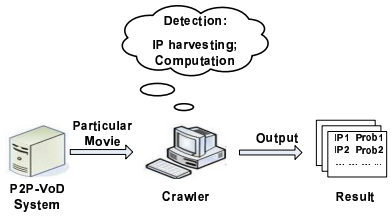


Fig. 1. Overview of our Detection Procedure

A. Capturing Technique for Suspicious IP Addresses

A P2P-VoD system usually contains tens of thousands of movies which are accessible to any user. Our goal is to determine the potential content servers that store a particular movie. Based on this methodology, one can recursively discover content servers of other movies in the system.

Let us first describe our capturing technique, which is to *harvest* and *identify* suspicious content servers of a particular movie, say M_k. Note that content servers of P2P-VoD systems usually have the following properties, based on which, we design the capturing technique:

High upload volume/capacity: Since a P2P-VoD system needs to provide service to users and uploads data to many peers, servers usually have a high upload capacity. Furthermore, the total amount of upload traffic from a content server is much higher than regular peers.

High upload/download ratio: Some resourceful peers in a P2P-VoD system may also have high upload volume or capacity. This is especially true for those peers that have broadband access or have cached large number of movies [2]. To distinguish these resourceful peers from servers of a P2P-VoD system, we exploit the second characteristic that a server usually has a high upload/download ratio. In other words, servers provide more traffic (or data) to peers than that they receive from peers.

High availability: One cannot solely rely on the first two characteristics to determine whether an end host is a content server or not. Due to the multiple movie caching (MVC) replication strategy of the current P2P-VoD systems [2], it is possible that a resourceful peer can simply provide upload service of its cached movie while requesting small amount of meta information. So we exploit the third characteristic that content servers are *online* and *operational* most of the time. The high availability is an essential characteristic since content servers have to provide instant access. Other peers do not have this characteristic since they may leave the P2P network after they finish watching their desired movies. Based on these characteristics, our capturing methodology can be divided into two steps:

Step 1: We use an "*exhaustive method*" to harvest all peers and content servers which can provide upload service for a particular movie, say M_k. A P2P-VoD client software will be running in one computer, say $\mathcal{C}_n$, which serves as a normal peer, while another computer, $\mathcal{C}_d$, discovers all communicating peers of $\mathcal{C}_n$ via the "iptables" software. Any communicating peer that satisfies the first property (i.e., high upload volume/capacity) is kept in the *suspicious list* (SL). Since a P2P client can only communicate with a finite number of peers at any given time, we also use iptables to block those peers in SL. This way, we force the normal client $\mathcal{C}_n$ to communicate with a new set of peers and repeat this process until we discover all peers that have a copy of the movie M_k. To reduce the size of SL and enhance the accuracy to discover the content server of M_k, we use the second property: servers have high upload/download ratio, to further filter unqualified peers from SL.

Step 2: After step 1, SL contains content servers *and* resourceful peers that have cached movie M_k. We exploit the third property: content servers are highly available so to differentiate content servers from those resourceful peers. We only focus on those peers in SL. We check their online status each time by probing their status in the P2P-VoD network. After a sequence of probes, we will show how to accurately determine the probability that a given peer in SL is indeed a content server. Detail explanation will be given in later subsections.

Details of the capture methodology: In order to implement this harvest and capture technique, we maintain two computers, $\mathcal{C}_n$ and $\mathcal{C}_d$, and we establish a network address translation (NAT) system between them. Computer $\mathcal{C}_n$ uses the Windows OS and P2P-VoD softwares (e.g., in our case, PPLive or PPStream client software, but it can be any P2P-VoD software) and acts as a normal client. The other computer $\mathcal{C}_d$ uses the Linux OS and acts as a NAT server that monitors any incoming/outgoing traffic of $\mathcal{C}_n$. All IP capturing and filtering operations are implemented on the NAT server $\mathcal{C}_d$.

We implement a software to further analyze all these captured packets. Since all packets are stored in the netfilter

queue, we use the libipq library to extract them. The main capturing algorithm is depicted in Algorithm 1.

Algorithm 1 Capturing Algorithm

```
1: ihandle=ipq_create_handle(0, PF_INET);
2: ipq_set_mode(ihandle,IPQ_COPY_PACKET,BUF_SIZE);
3: while TRUE do
4:     ipq_read(ihandle, buf, BUF_SIZE, 0);
5:     msg = ipq_get_packet(buf);
6:     iph = (struct iphdr*)msg→payload;
7:     packet_handling_procedure(iph);
8: end while
```

Packet handling procedure is described in Algorithm 2. Source addresses and destination addresses of the packets are stored in *iph→saddr* and *iph→daddr* respectively. Based on this information, we can maintain a *PEERLIST* with all peers that are communicating with our client $\mathcal{C}_n$ (with our client being the receiver of these packets). In addition, we can determine whether the packet is incoming or outgoing, e.g., if *iph→saddr* = {*IP address of the Client*}, then it is an outgoing packet. Furthermore, *iph→tot_len* stores the size of the packet. Once the upload volume of a peer exceeds a pre-defined *THRESHOLD*, we set the third parameter of *ipq_set_verdict()* to be *NF_DROP*, which signals $\mathcal{C}_d$ to drop this packet so that it would not reach our client $\mathcal{C}_n$. This way, we force $\mathcal{C}_n$ to contact the tracker to seek more peers or content servers to download the video blocks. Any IP addresses in which we need to drop their packets to $\mathcal{C}_n$ are stored in the *BLOCKLIST*. The *BLOCKLIST* contains all *suspicious IP addresses*.

Algorithm 2 Packet Handling Procedure

```
1: if iph→daddr = client_IP then
2:     if iph→saddr in BLOCKLIST then
3:         ipq_set_verdict(ihandle,msg→packet_id,
           NF_DROP, 0,NULL);
4:     else
5:         ipq_set_verdict(ihandle,msg→packet_id,
           NF_ACCEPT, msg→datalen, msg→payload);
6:         if iph→saddr in PEERLIST then
7:             PEERLIST(iph→saddr).tot_upv =
               PEERLIST(iph→saddr).tot_upv + iph→tot_len
8:             if PEERLIST(iph→saddr).tot_upv>THRESHOLD then
9:                 ADD iph→saddr TO BLOCKLIST
10:            end if
11:        else
12:            ADD iph→saddr TO PEERLIST
13:            PEERLIST[iph_saddr].tot_upv=iph→tot_len
14:        end if
15:    end if
16: else
17:    if iph→daddr in PEERLIST then
18:        PEERLIST(iph→daddr).tot_downv =
           PEERLIST(iph→daddr).tot_upv + iph→tot_len
19:    else
20:        ADD iph→daddr TO PEERLIST
21:        PEERLIST[iph_daddr].tot_downv=iph→tot_len
22:    end if
23: end if
```

In summary, the harvest methodology works as follows: for all incoming packets to our client $\mathcal{C}_n$, we first check its source address. If it is in the *BLOCKLIST*, we simply drop the packet. Otherwise we accept it and keep track of its payload length. If the total upload volume exceeds a pre-defined *THRESHOLD*, we add this IP address to the *BLOCKLIST*. For any outgoing packets from $\mathcal{C}_n$, we also maintain its payload length information because we will use it to determine the upload/download ratio for further peers' filtering.

We want to emphasize that the capture and harvest technique is totally automated and this technique has a very small computational overhead. For each incoming/outgoing packet, the time complexity is just $O(N)$, where N is the number of IP addresses in the suspicious list SL. In all experiments that we carried out, we were able to capture *several hundreds* of suspicious IP addresses for each targeted movie, and each time when we instantiate the capture and harvest procedure, we were able to harvest and capture *all potential* content servers in five minutes. In the following, we describe the *analytical framework* to determine the probability that an IP address in the SL is a content server given its online availability.

B. Analytical Framework

As stated above, servers of a P2P-VoD system are usually highly available. To test this characteristic, we propose to apply the above mentioned *capture-and-harvest procedure* $n > 1$ times so to obtain the online status of each IP address in the suspicious list (SL). We develop a statistical model to quantify the probability that the given IP address (say, peer x) is a content server. To determine the probability that an IP address (or say, peer x) in the suspicious list is indeed a content server, we consider the following baseline cases:

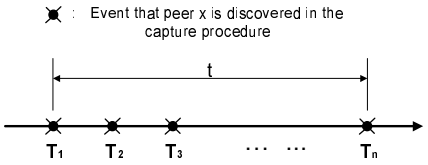


Fig. 2. Statistical Analysis for Case 1

Case 1 (E_1) : We denote this event as E_1 and it represents the case that we detect peer x in *all* of the n capture and harvest procedures. Fig. 2 illustrates this scenario. T_i denotes the time instance of capture. The total interval of these n captures is denoted by t. In each capture, the probability that peer x is detected is denoted by g. Let $\mathcal{S}$ denote the event that peer x is a content server and $\mathcal{N}$ denote the event that the peer x is a normal peer. The probability that peer x is a normal peer given E_1 occurs is:

$$P(\mathcal{N}|E_1) = \frac{P(\mathcal{N},E_1)}{P(E_1)} = \frac{P(\mathcal{N},E_1)}{P(\mathcal{S},E_1)+P(\mathcal{N},E_1)} = \frac{P(E_1|\mathcal{N})P(\mathcal{N})}{P(\mathcal{S},E_1)+P(E_1|\mathcal{N})P(\mathcal{N})}. \quad (1)$$

The probability that E_1 occurs given x is a normal peer is:

$$P(E_1|\mathcal{N}) = P(\mathcal{U}>t)g^n, \quad (2)$$

where $\mathcal{U}$ is the random variable denoting the uptime (or availability) of peer x. In summary, a normal peer x is detected if its uptime is longer than t, which is the total capturing duration, and that x is discovered in all n captures. It is easy to see that given peer x is a content server, the probability that E_1 occurs is $P(E_1|\mathcal{S}) = g^n$. So we obtain:

$$P(\mathcal{S}, E_1) = P(E_1|\mathcal{S})P(\mathcal{S}) = P(\mathcal{S})g^n. \quad (3)$$

Substituting Eq. (2) and (3) into Eq. (1), we have:

$$\begin{aligned} P(\mathcal{N}|E_1) &= \frac{P(\mathcal{U}>t)g^n P(\mathcal{N})}{P(\mathcal{S})g^n + P(\mathcal{U}>t)g^n P(\mathcal{N})} \\ &= \frac{P(\mathcal{U}>t)}{P(\mathcal{S})/P(\mathcal{N}) + P(\mathcal{U}>t)}. \end{aligned}$$

Hence, the probability that peer x is a content server given that the event E_1 is:

$$P(\mathcal{S}|E_1) = 1 - P(\mathcal{N}|E_1) = \frac{P(\mathcal{S})/P(\mathcal{N})}{P(\mathcal{S})/P(\mathcal{N}) + P(\mathcal{U}>t)}. \quad (4)$$

In here, we assume that each movie is stored in one logical server, therefore if we discovered M peers in the suspicious list, $P(\mathcal{S})/P(\mathcal{N}) = 1/(M-1)$. Now, the only unknown is the random variable $\mathcal{U}$. We will describe in later subsection how to determine the probability distribution of $\mathcal{U}$.

Case 2 (E_2) : We denote this as event E_2 and it represents the case that peer x is captured in the first $n' < n$ times, but for the remaining $n - n'$ capturing procedures, peer x is not detected. This case is illustrated in Fig. 3.

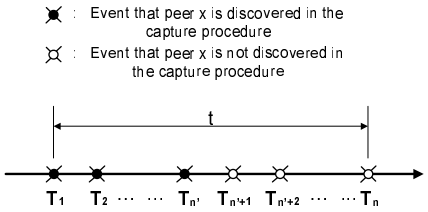


Fig. 3. Statistical Analysis for Case 2

Let us first define $\mathcal{A}_i$ ($i = 1, .., n - n'$) as the event that peer x is available only in the interval of $[T_1, T_{n'+i})$. The probability that $\mathcal{A}_i$ happens can be expressed as:

$$P(\mathcal{A}_i) = P(\mathcal{U} \geq T_{n'+i-1} - T_1)P(\mathcal{U} < T_{n'+i} - T_1). \quad (5)$$

Similar to the previous case, we can express:

$$P(\mathcal{N}|E_2) = \frac{P(\mathcal{N}, E_2)}{P(E_2)} = \frac{P(E_2|\mathcal{N})P(\mathcal{N})}{P(E_2|\mathcal{S})P(\mathcal{S}) + P(E_2|\mathcal{N})P(\mathcal{N})}. \quad (6)$$

Let us first derive $P(E_2|\mathcal{N})$. In this case, peer x can be online in interval $[T_1, T_{n'+i})$, for $i = 1, .., n - n'$. Therefore:

$$P(E_2|\mathcal{N}) = \sum_{i=1}^{n-n'} P(\mathcal{A}_i) g^{n'}(1-g)^{i-1} + P(\mathcal{U}>t) g^{n'}(1-g)^{n-n'}. \quad (7)$$

Given x is a content server, the probability that E_2 occurs is:

$$P(E_2|\mathcal{S}) = g^{n'}(1-g)^{n-n'}. \quad (8)$$

Substituting Equation (7) and (8) into Equation (6), we have

$$P(\mathcal{N}|E_2) = \frac{\Omega}{(1-g)^{n-n'}P(\mathcal{S})/P(\mathcal{N}) + \Omega}, \quad (9)$$

$$P(\mathcal{S}|E_2) = 1 - \frac{\Omega}{(1-g)^{n-n'}P(\mathcal{S})/P(\mathcal{N}) + \Omega}, \quad (10)$$

$$\text{where } \Omega = \sum_{i=1}^{n-n'} P(\mathcal{A}_i)(1-g)^i + P(\mathcal{U}>t)\cdot(1-g)^{n-n'}. \quad (11)$$

Again, the only unknown is the random variable $\mathcal{U}$, we will derive its probability distribution in later subsection.

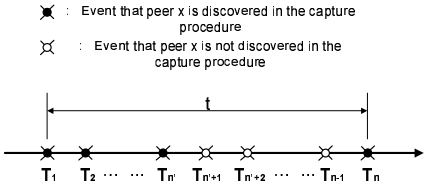


Fig. 4. Statistical Analysis for Case 3

Case 3 (E_3) : Denote this as event E_3 and it represents the case that peer x rejoins the system after it went offline for some time. From Fig. 4, we can see that peer x shows up for the first n' observations, and misses for the next $n - n' - 1$ observations, but it appears again in the last measurement. In this case, the probability that x is a normal peer is:

$$P(\mathcal{N}|E_3) = \frac{P(\mathcal{N}, E_3)}{P(E_3)} = \frac{P(E_3|\mathcal{N})P(\mathcal{N})}{P(E_3|\mathcal{S})P(\mathcal{S}) + P(E_3|\mathcal{N})P(\mathcal{N})}. \quad (12)$$

If x is a normal peer, it may leave the system in any of these time intervals $(T_{n'+i-1}, T_{n'+i}]$, for $i = 1, ..., n-n'$, and rejoin the system after $T_{n'+i}$, or peer x is always available in the measurement period t but just missed by the capture and harvest procedure for $n - n' - 1$ times. Here we use $\mathcal{R}_{\tau_{ij}}$ to denote the event that peer x rejoins the system after it left for τ_{ij} time, where τ_{ij} represents the time interval from $T_{n'+i}$ to $T_{n'+j}$. Consider all these possibilities, given x is a normal peer, the probability that event E_3 happens is

$$\begin{aligned} P(E_3|\mathcal{N}) = \sum_{i=1}^{n-n'-1} [P(\mathcal{A}_i)(\sum_{j=i+1}^{n-n'} P(\mathcal{R}_{\tau_{ij}})(1-g)^{\Theta})] \\ + P(\mathcal{U}>t)\cdot g^{n'+1}\cdot(1-g)^{n-n'-1}, \end{aligned}$$

where $\Theta = n - n' - j + i - 1$.

If x is a content server, event E_3 happens with probability $P(E_3|\mathcal{S}) = g^{n'+1}\cdot(1-g)^{n-n'-1}$. Therefore, the probability that x is a normal peer is

$$P(\mathcal{N}|E_3) = \frac{P(E_3|\mathcal{N})}{\frac{P(\mathcal{S})}{P(\mathcal{N})}g^{n'+1}(1-g)^{n-n'-1} + P(E_3|\mathcal{N})}, \quad (13)$$

and the probability that peer x is content server is:

$$P(\mathcal{S}|E_3) = 1 - P(\mathcal{N}|E_3). \quad (14)$$

In summary, we have to consider different combination of these three cases to estiminate the probability that the captured peer is indeed a content server.

Estimation of $P(\mathcal{U})$: $P(\mathcal{U})$ is an important probability function that appeared in all three baseline cases we mentioned above. Given a specific time $\widetilde{T}$, let $\tilde{t}$ denote the interval of $\widetilde{T} - T_1$, then $P(\mathcal{U} > \tilde{t})$ represents the probability that peer x's life time is longer than $\tilde{t}$. A statistical estimator of this survival function is the *product-limit estimator* [3] which can be expressed as:

$$P(\mathcal{U} > \tilde{t}) = \begin{cases} 1 & \text{if } \tilde{t} < T_1, \\ \prod_{T_i \leq \tilde{t}} (\frac{n_i - d_i}{n_i}) & \text{otherwise.} \end{cases} \quad (15)$$

Assume that the total number of observations is κ, T_i ($1 \leq i \leq \kappa$) are the observation instances in ascending order such that $T_1 < ... < T_\kappa$, while n_i denotes the number of remaining peers just after time T_{i-1}, and d_i denotes the number of peers that leave the P2P system in the interval $[T_{i-1}, T_i)$. In our derivation of $P(\mathcal{U})$, once a peer does not show up in a measurement observation, we consider this peer has departed from the P2P-VoD system. It is easy to see that $n_i - d_i = n_{i+1}$. Therefore, Equation (15) becomes

$$P(\mathcal{U} > \tilde{t}) = \begin{cases} 1 & \text{if } \tilde{t} < T_1, \\ \frac{n_i - d_i}{n_1} & \tilde{t} \in [T_i, T_{i+1}). \end{cases} \quad (16)$$

Based on Equation (16), the probability that a peer's uptime is longer than $\tilde{t}$, equals to the number of remaining peers divided by the number of initial peers. To evaluate this probability, we carried out extensive measurements on the Internet and this probability function is illustrated in Fig. 5.

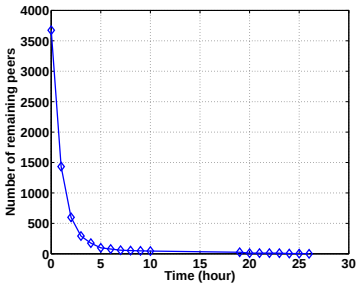


Fig. 5. Number of Remaining Peers vs. Time

In this measurement, we explore 200 distinct movies. For each movie, we only focus on their initial peers (i.e., peers that show up at the first observation). We keep observing these initial peers and record their online status at each observation. The total number of initial peers is around 3,600, and they leave the system gradually as time goes on. At the end of each observation, we keep the number of remaining peers in the system. The result is illustrated in Figure 5. Based on Eq. (16), we use this measurement result to estimate $P(\mathcal{U})$ in our model. For example, for $\tilde{t} = 2$, we have $P(\mathcal{U} > 2) \approx \frac{600}{3600} \approx 0.167$.

Estimation of $P(\mathcal{R}_\tau)$: Let $\mathcal{R}_\tau$ denote the event that a peer rejoins the P2P-VoD system after it left for τ time unit. The probability of this event, $P(\mathcal{R}_\tau)$, is used in Case 3 of our analytical framework. Again, we perform extensive measurement to characterize this probability function.

In our measurement, we set the time unit to be one hour. So $\mathcal{R}_\tau$ represents the event that a peer rejoins the system after it left for τ hours. Let $N_{\mathcal{R}_\tau}$ denote the number of times $\mathcal{R}_\tau$ occurs and N_0 denote the total number of departure events. If the sample size N_0 is large enough, we have $\frac{N_{\mathcal{R}_\tau}}{N_0} \approx P(\mathcal{R}_\tau)$. Therefore, we use $\frac{N_{\mathcal{R}_\tau}}{N_0}$ as an *unbiased estimator* of $P(\mathcal{R}_\tau)$.

In order to estimate $N_{\mathcal{R}_\tau}$ and N_0, we measure multiple movies and obtain about 4,600 initial peers. We then measure their online status every hour. When a peer is online in the last observation instant but off-line in the current observation instant, we mark it as a departure. Furthermore, if a departed peer is offline in previous observation instances but online in the current observation instance, we consider this as a rejoin event for this peer. The result is illustrated in Figure 6. With this information and the total departure events, we can use $\frac{N_{\mathcal{R}_\tau}}{N_0}$ to estimate $P(\mathcal{R}_\tau)$.

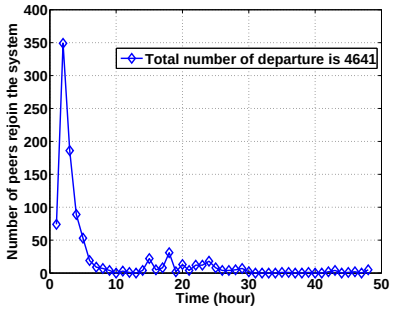


Fig. 6. Number of Rejoining Peers in Each Hour vs. Time

C. Results of our Detection Methodology

To validate our detection methodology, we carry out the following experiments. In June 2010, we carried out extensive measurements to detect content servers of PPLive's P2P-VoD system. Using our capture-and-harvest technique as well as our analytic framework, we aim to detect content servers of about 100 movies in PPLive system. In the experiment, we were able to discover 44 content servers from the suspicious list SL (note that multiple movies may be stored in the same content server). We also received results from engineers in PPLive, and they confirmed the correctness of our detection results. The sample output of our detection technique is depicted in Figure 7.

As showed in Figure 7, server 1 which has an IP address of 202.*.*.14 (we masked out some bits to provide confidentiality to PPLive's servers) were online for *all* of our observations. Using the analytical framework, we derived the probability of it being a content server and it has a high probability of 0.969 of being a content server in the PPLive's P2P-VoD system. Results for IP address 122.*.*.228 and 121.*.*.205 can be viewed as Case 2 and the combination of Case 3 and Case

		Time1		Time i_1	Time i_2	Time i_3		Time i_4	Time i_5		Time i_6	Pr(sr)
Server 1	220.*.*.14	1	… …	1	1	1	… …	1	1	… …	1	96.90%
Server 2	221.*.*.17	1	… …	1	1	1	… …	1	1	… …	1	95.32%
… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …
Server 44	124.*.*.106	1	… …	1	1	1	… …	1	1		1	96.60%
Peer 1	122.*.*.228	1	… …	1	0	0	… …	0	0	… …	0	$\leqslant$3. 85E−19
… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …
… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …	… …
Peer 15000	121.*.*.205	1	… …	1	0	1	… …	1	0	… …	0	$\leqslant$9.76E-13

Fig. 7. Sample Results after Running the Capture-and-Harvest Technique and Analytical Model

2, respectively. Based on our analytical framework, one can conclude that their probabilities of being content servers are extremely low. From this figure, one can see the effectiveness of our analytical framework. In particular, in *classifying* IP addresses in the suspicious list into content servers. We also carried out experiments on PPStream's P2P-VoD system. We aim to detect servers of three movies, and we were able to capture 430 suspicious IP addresses and discovered three IP addresses with high probability of being servers.

III. Protective Architecture for P2P-VoD Systems

In the previous section, we presented a methodology that can perform IP harvesting and filtering. More importantly, it can effectively identify content servers of a P2P-VoD system. This can be detrimental to P2P-VoD service providers because once these servers are discovered, malicious users can launch various attacks, e.g., DDoS or low-rate TCP attack, to disrupt the service. In this section, we propose a novel architecture, which we call *shield nodes*, so as to provide servers' anonymity. We also analyze the performance of this architecture and show its resiliency.

In order to protect servers' anonymity, the shield nodes architecture prevents normal peers from directly communicating with content servers. To achieve this goal, we create a new layer, which we call the shield nodes layer, between content servers and normal peers. These shield nodes act as transmission proxies in a P2P-VoD system. In addition, they provide *server association randomization* as well as *path redundancy* so as to achieve a high reliability under attack. Figure 8 illustrates this architecture.

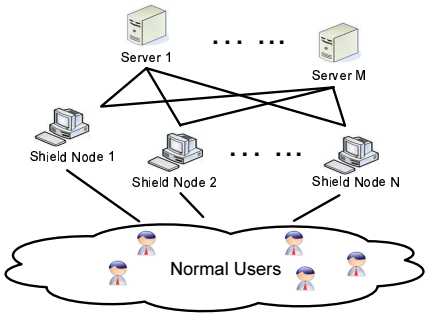


Fig. 8. Using Shield Nodes as Protection

In this architecture, shield nodes and servers are fully meshed to provide path redundancy. The interconnection between them can be achieved by high bandwidth LAN or switch. Each shield node maintains a hash table which contains the information of all available movies and their corresponding content servers. When a request arrives, shield nodes first find out the corresponding content server according to the hash table entry, and then forward the request to this server. When the video blocks are returned, shield nodes simply forward these video data to the requesting peer. This way, normal peers cannot directly communicate with these content servers.

Under this architecture, P2P users are oblivious to the presence of shield nodes since they can obtain the desired video blocks. For the content server, the probability of identity exposure is significantly reduced. The only change we need to make is on the trackers. When a peer asks for some neighboring peers, instead of presenting the IP addresses of content servers, the tracker simply returns the IP address of one of the shield nodes. This selected shield node would be in charge to relay the video block to that requesting peer.

We like to point out that attackers can use the capture-and-harvest technique we discuss to determine the IP addresses of shield nodes, but as we will show, one can *randomized* the movie-shield node assignment and makes the identity attack difficult, if not impossible. Moreover, one can periodically assigned new IP addresses to shield nodes to further enhance the availability of P2P-VoD service.

A. A Randomized Assignment Algorithm

In this subsection, we present a randomized algorithm which will be used by trackers. The goal is to randomize the *movie-shield node* assignment so that the P2P-VoD system can be more resilient to the IP capturing and filtering technique we introduced in Section II.

Randomized Algorithm: Each time a peer sends a request to the tracker to obtain some neighbors, instead of presenting the IP address of a content server, the tracker uses the movie ID and the current time as inputs to the randomized algorithm to decide which shield node should be assigned to the requesting peer. Algorithm 3 depicts this assignment.

Algorithm 3 Randomized Algorithm

1: $\overline{T} \leftarrow (00:00:00, 1st, Jan, 2000)$
2: Time_Period $\leftarrow$ 12
3: Diff_Time $\leftarrow$ CTime - $\overline{T}$
4: Para $\leftarrow \lceil$ Diff_Time / Time_Period $\rceil$
5: SN_ID $\leftarrow$ Rand(Para, Movie_ID)
6: return SN_ID

The randomized algorithm can be explained as follows. Firstly, *Diff_Time* represents the time difference between the

current time *CTime* and $\overline{T}$. For every 12 hours, *Para* has the same value and *Rand*(x,y) is a hash function which generates a random number between 1 and n, where n is the number of shield nodes in a P2P-VoD system. This simple randomization allows the tracker to associate the movie and shield node pairing with a 12 hours duration. We will show this duration is sufficient to effectively protect the P2P-VoD system.

Note that the above randomization algorithm is based on the capture-and-harvest technique and measurement results we discussed in previous section. The length of each time period is decided by the P2P-VoD system (tracker). If this length is chosen properly (e.g., 12 hours in our case), then even if the allocation of movie and shield nodes is fixed during each time period, the shield nodes are relatively safe because it takes at least 24 hours (please refer to Fig. 5) to identify whether an IP address is a content server of a P2P-VoD system. Therefore, as long as the time period is shorter than 24 hours, the probability of exposing the shield nodes would be relatively low. Before the attackers could further filter the suspicious IP addresses, the *movie-shield node* assignment would change because the time period has expired. We will show this randomization makes it computationally difficult to discover shield nodes.

B. Markov Model for Shield Nodes Architecture

We present the analysis to quantify the time needed to disrupt the P2P-VoD service by using the previous mentioned identity attack. Note that under this enhanced architecture, one needs to discover *all* shield nodes to disrupt the service of a P2P-VoD system. First, we state some definitions and assumptions that we use in our analysis.

Definition (a): If a suspicious IP (say, x) is observed as online in an observation period, we say x is *identified* once.

Definition (b): According to the number of times the suspicious IP addresses have been *identified*, we divide them into three mutually exclusive sets:
SET 0 ($\mathcal{I}_0$): peers that have not been *identified*.
SET 1 ($\mathcal{I}_1$): peers that have been *identified* only once.
SET 2 ($\mathcal{I}_2$): peers that have been *identified* more than once.

Assumption (a): Each time period, the attackers can harvest *all* peers in a P2P-VoD system. Note that this is clearly an optimistic assumption and it provides an *upper bound* on the probability of identifying a shield node.

Assumption (b): When a shield node with an IP address x is *identified* twice, attackers can say with a high probability that this IP address x is a shield node.

It is easy to see that these two assumptions favor the attackers. According to our measurement results in the previous section, it is possible that a normal peer is in $\mathcal{I}_1$. But the probability that a normal peer is in $\mathcal{I}_2$ is extremely small, thus we can make assumption (b) and consider all peers in $\mathcal{I}_2$ are potential shield nodes.

We use a discrete time Markov chain $\mathcal{M}$ to describe the dynamic of the system. Assume that there are n shield nodes in the architecture, the state space of $\mathcal{M}$ is

$$S = \{(n_c, n_s) | n_c + n_s \leq n, n_c \geq 0, n_s \geq 0\}, \tag{17}$$

where n_s and n_c represent the number of shield nodes in $\mathcal{I}_1$ and in $\mathcal{I}_2$ respectively.

Note that in each time period, only one shield node would be assigned to deliver a particular movie. From Assumption (a), after each time period, attackers can always get the information of this assigned shield node (say, y), together with the information of some other suspicious IP addresses. There are three possibilities about y: $y \in \mathcal{I}_0$; $y \in \mathcal{I}_1$; $y \in \mathcal{I}_2$. Let $\boldsymbol{P}$ be the one step transition probability matrix of $\mathcal{M}$. For a general case (n_c,n_s), the one step transition probability is based on the three possibilities of y:
(1) $y \in \mathcal{I}_0$: it means y has not been *identified* before and this is the first time that y is *identified*. So y should be moved into $\mathcal{I}_1$. The state transfers from (n_c,n_s) to (n_c,$n_s + 1$). The transition probability of this event is $(n - n_s - n_c)/n$.
(2) $y \in \mathcal{I}_1$: then this is the second time y is *identified*, y should be moved from $\mathcal{I}_1$ to $\mathcal{I}_2$. The state transfers from (n_c,n_s) to (n_c+1,n_s−1). The transition probability of this event is n_s/n.
(3) $y \in \mathcal{I}_2$: then y has already been considered as a potential shield nodes, so the system remains in the same state. The transition probability is n_c/n.

Let $P((a,b)|(c,d))$ denote the one step transition probability from state (c,d) to state (a,b), then the one step transition probability of state (n_c,n_s) is:

$$\begin{aligned} P((n_c+1, n_s-1)|(n_c,n_s)) &= n_s/n, \\ P((n_c,n_s)|(n_c,n_s)) &= n_c/n, \\ P((n_c,n_s+1)|(n_c,n_s)) &= (n-n_c-n_s)/n, \\ P((n_i,n_j)|(n_c,n_s)) &= 0 \quad \forall (n_i,n_j) \quad \text{otherwise.} \end{aligned} \tag{18}$$

The performance we aim to derive is the average number of periods for the system to reach the absorption state $(n,0)$, given the initial state $(0,0)$. In other words, this is the average number of time periods for attackers to identify all n shield nodes. We can derive this performance measure based on the "*theory of fundamental matrix*" [11]. In general, we can consider the Markov chain $\mathcal{M}$ with N states, $S_1, S_2, ..., S_N$, with state S_N being $(n,0)$ as the absorbing state, and the remaining states are transient states. The transition probability matrix $\boldsymbol{P}$ of such a chain can be re-written as:

$$\boldsymbol{P} = \left[\begin{array}{c|c} \boldsymbol{Q} & \boldsymbol{C} \\ \hline \boldsymbol{0} & 1 \end{array}\right], \tag{19}$$

where $\boldsymbol{Q} \in \mathbb{R}^{(N-1)\times(N-1)}$ is a sub-stochastic matrix (i.e., with at least one row sum less than 1) describing the transition probabilities among the transient states. $\boldsymbol{C}$ is a column vector representing the transition from each transient state to the absorbing state, and $\boldsymbol{0}$ is a row vector of $(N-1)$ zeros. The $k-$steps transition probability matrix $\boldsymbol{P}^k$ is:

$$\boldsymbol{P}^k = \left[\begin{array}{c|c} \boldsymbol{Q}^k & \boldsymbol{C}' \\ \hline \boldsymbol{0} & 1 \end{array}\right], \tag{20}$$

In here, $\boldsymbol{Q}^k[i,j]$ represents the probability of arriving in (transient) state S_j after exactly k transitions, given that the starting state is S_i. It can be shown that $\sum_{k=0}^{t} \boldsymbol{Q}^k$ converges as t approaches infinity [PARZ 1962]. This implies that the inverse

matrix $(\boldsymbol{I}-\boldsymbol{Q})^{-1}$, or what we call the fundamental matrix, $\boldsymbol{M}$, exists and is given by $\boldsymbol{M}=(\boldsymbol{I}-\boldsymbol{Q})^{-1}=\sum_{k=0}^{\infty}\boldsymbol{Q}^k$.

Given the starting state is state 1, let V_j denote the average number of times state j is visited before the *absorbing state* is reached, from the theory in [11], we have:

$$V_j=\delta_{1j}+\sum_{i=1}^{N-1}V_iP_{ij}, j=1,2,...,N-1, \qquad (21)$$

where δ_{ij} is the Knonecker δ function.

In our model, the starting state is $S(0,0)$. Let $V(n_c,n_s)$ denote the average number of times that state $S(n_c,n_s)$ is visited before reaching the absorbing state $S(n,0)$. From Eq (18) and Eq (21), we can express $V(n_c,n_s)$ recursively as:

$$\begin{aligned}
V(0,0) &= 1\\
V(n_c,0) &= \frac{n_c}{n}V(n_c,0)+\frac{1}{n}V(n_c-1,1), \quad 1\le n_c<n,\\
V(0,n_s) &= \frac{n-n_s+1}{n}V(0,n_s-1), \quad 1\le n_s\le n,\\
V(n_c,n_s) &= \frac{n_s+1}{n}V(n_c-1,n_s+1)+ \qquad (22)\\
&\quad \frac{n-n_c-n_s+1}{n}V(n_c,n_s-1)+\\
&\quad \frac{n_c}{n}V(n_c,n_s), \qquad n_c,n_s\ge 1; n_c+n_s\le n.
\end{aligned}$$

Define $E[\mathcal{T}]$ as the average number of time periods required to detect all n shield nodes, which can be expressed as: $E[\mathcal{T}]=\sum_{n_c=0}^{n-1}\sum_{n_s=0}^{n-n_c}V(n_c,n_s)$.

To validate the correctness of this model, we develop a discrete event P2P-VoD simulator. Peers in the simulation mimic the PPLive VoD streaming protocol. We simulate the shield-node architecture and varies the number of shield nodes to evaluate its resiliency under attack (e.g., DDoS attack to the exposed shield nodes). We average the time it takes to identify all shield nodes and compare this result with our theoretical results, which are obtained via the theory of fundamental matrix M. The results are depicted in Table I and it shows that our analytical model is very accurate. This also shows that using using 30 shield nodes, attackers have to continuously sustain their attack for 2,126 hours (or 88 days) so as to bring down the P2P-VoD service. Based on Table I, we can see that

# of SN	Theoretical Res(hr)	Simulation Res(hr)
2	66	66.03
5	228.50	229.78
10	554.76	555.48
15	917.76	914.22
20	1304.37	1310.51
25	1708.41	1712.19
30	2126.28	2124.55

TABLE I
AVERAGE TIME REQUIRED TO HARVEST ALL SHIELD NODES (TIME PERIOD IS 12 HOURS)

if one time period is set to be 12 hours, it will take $\frac{554.76}{24}\approx 23$ days to detect all $n=10$ shield nodes.

C. Performance Analysis

We carry out simulation to study the time overhead, in particular, the probability cumulative function (PDF) of detecting all n shield nodes. In Fig. 9, each point on the curves represents the probability that all n shield nodes are detected in less than or equal to a certain number of time periods as indicated in the x-axis. When the number of shield node increases, it takes a longer time to detect all shield nodes. As we discussed previously, the maintenance cost of shield nodes is much lower than that of servers. So it is justifiable for us to maintain a large number of shield nodes. For example, when one uses 20 shield nodes, it takes around 175 time periods to detect all these shield nodes. It is equivalent to $175*12/24=87.5$ days. This implies that attackers may need to sustain a DDoS for 87.5 days to disrupt the P2P-VoD service, which is computationally expensive for any attacker.

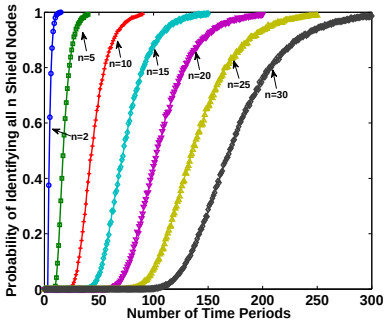
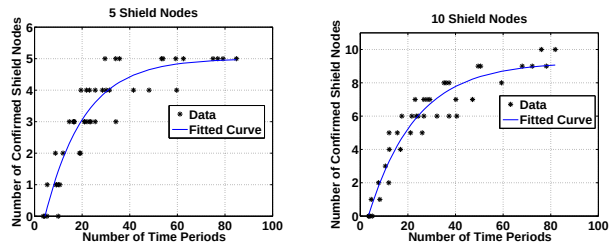


Fig. 9. PDF of Detecting All Shield Nodes (each time period is 12 hours)

Let us discuss the storage overhead to discover all n shield nodes. Note that the attacker needs to keep the information of all observed suspicious IP addresses. We carry out measurement on PPLive system and see that each observation on a popular movie will result in around 400 to 600 suspicious IP addresses. Each hour, around $1/3$ of these suspicious end hosts remain in the system. This implies that for each hour, at least 300 to 400 new suspicious IP addresses will join the system. If the attackers observe the P2P-VoD system for a long time, say 100 days, so as to detect all shield nodes, this implies that they need to process around $400\times 24\times 100=960,000$ suspicious IP addresses. Without the shield node architecture, one only needs to focus on 400 to 600 suspicious IP addresses is enough for the attackers to identify the server. Therefore, the computation and storage overheads are another justification that discovering all shield nodes is difficult and this architecture is effective in providing anonymity protection.

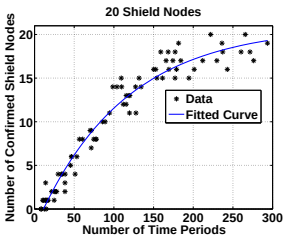
D. Experiment Result on PlanetLab

We also implement this shield nodes architecture and carry out a set of experiments on the PlanetLab [6]. In our experiments, there are five different types of nodes: (1) servers, which contain the movies; (2) trackers, which provide information to peers about their possible neighbors and shield nodes to contact; (3) shield nodes, which act as a proxy between servers and normal peers; (4) peers, which are normal users who want to watch a movie in the P2P-VoD's database. These peers can leave the system at any time; (5) crawler (or attacker), which attempts to harvest all shield nodes in the system. In our experiments, the number of shield nodes is set as 5, 10,

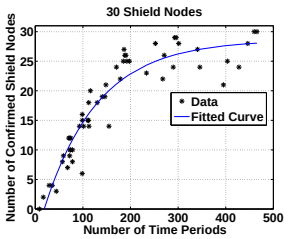


(a) 5 Shield Nodes

(b) 10 Shield Nodes



(c) 20 Shield Nodes



(d) 30 Shield Nodes

Fig. 10. Exp. Results on PlanetLab: Time Needed to Discover Shield Nodes

20, 30 respectively. The number of normal peers are randomly chosen from 10 to 100. There is a unique server, tracker and crawler respectively.

Trackers use the *randomized algorithm* to assign shield nodes to peers. Within a time period, the assignment of shield nodes and movies is fixed. Note that this assignment would change in the next time period. The crawler stays in the P2P system all the time and harvests all peers. If the crawler finds any peer that is online for more than one time period, this peer is confirmed as a shield node. Fig. 10 illustrates the results from our experiments. From these results, we can see that it confirms with our Fundamental Matrix analysis as well as the simulation results in the previous subsections. We like to point out that in a realistic networking environment, there are number of factors which make it very difficult to discover *all* n *shield nodes*, e.g., packets loss, dynamic topology in the P2P networks, results of the randomized algorithms, etc. Due to these factors, we see from the experimental results that it takes longer than the theoretical time to discover all shield nodes, and this confirms the protection capability of our architecture.

IV. Related Work

Significant efforts have been devoted to the measurement and improvement of P2P systems. Recently, researchers focus on the security issues in P2P systems, such like pollution attacks [1] [10] [8]. Another class of security issue is anonymity problems. Tsang *et al.* [12] proposed an authentication strategy to protect the anonymity in the system. Puttaswamy *et al.* [7] introduced a novel architecture called Bluemoon, to protect the anonymity of P2P networks. In [14], authors discussed the role of supernodes in P2P networks. Liu *et al.* [5] discuss how to distill superior peers in P2P streaming system, they propose a criteria to obtain the normal peers with better capability in P2P streaming system. However, all these existing works are not suitable for identifying content servers in P2P-VoD systems.

V. Conclusion

This paper investigates the identity exposure problem in current P2P-VoD systems. Contrary to the conventional wisdom, we design an efficient capture-and-filter technique that can expose the identity of content servers in a P2P-VoD system. We also present a mathematical model to quantify the probability that an end host is a server. We demonstrate our technique by detecting the servers of PPLive and PPStream systems and that our classification model is highly effective to identify content servers. To prevent identity exposure, we propose a novel architecture called shield nodes in which we add an extra layer as a transmission proxy between the servers and normal users. This way, there is no opportunity for normal users to have direct contact with servers. Further more, because of the connectivity between servers and shield nodes, data can always be provided from servers to users when *at least one shield node* is operational. This makes the P2P-VoD system more resilient to identity attack. To quantify the performance of our architecture, we use a DTMC and fundamental matrix theory to derive the average time to discover all shield nodes. We build a prototype in PlanetLab to demonstrate the robustness and resiliency of our architecture.

Acknowledgement: this research is supported in part by the SHIAE 8115032 and GRF 415309.

References

[1] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena. The pollution attack in p2p live video streaming: measurement results and defenses. In *P2P-TV '07*, pages 323–328, New York, NY, USA, 2007. ACM.

[2] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. *SIGCOMM Comput. Commun. Rev.*, 38(4):375–388, 2008.

[3] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958.

[4] T. K. T. Law, J. C. S. Lui, and D. K. Y. Yau. You can run, but you can't hide: An effective statistical methodology to trace back ddos attackers. *IEEE TPDS*, 16(9):799–813, 2005.

[5] Z. Liu, C. Wu, B. Li, and S. Zhao. Distilling superior peers in large-scale p2p streaming systems. In *IEEE INFOCOM*, pages 82–90, 2009.

[6] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of HotNets–I*, Princeton, New Jersey, October 2002.

[7] K. Puttaswamy, A. Sala, C. Wilson, and B. Zhao. Protecting anonymity in dynamic peer-to-peer networks. *ICNP*, pages 104–113, Oct. 2008.

[8] K. Shin, D. S. Reeves, I. Rhee, and Y. Song. Winnowing : Protecting p2p systems against pollution by cooperative index filtering. *TR-2009-2, Dept. of Computer Science, North Carolina State University*, 2009.

[9] H. Sun, J. C. S. Lui, and D. K. Y. Yau. Distributed mechanism in detecting and defending against the low-rate tcp attack. *Comput. Netw.*, 50(13):2312–2330, 2006.

[10] R. Thommes and M. Coates. Epidemiological modelling of peer-to-peer viruses and pollution. In *IEEE INFOCOM*, pages 1–12, April 2006.

[11] K. S. Trivedi. Probability and statistics with reliability, queuing and computer science applications. 1994.

[12] P. P. Tsang and S. Smith. PPAA: Peer-to-peer anonymous authentication. *Applied Cryptography and Network Security*, pages 55–74, 2008.

[13] T. Y. Wong, K. T. Law, J. C. S. Lui, and M. H. Wong. An efficient distributed algorithm to identify and traceback ddos traffic. *Computer Journal*, 49(4), 2006.

[14] B. Yang and H. Garcia-Molina. Designing a super-peer network. *Data Engineering, International Conference on*, 0:49, 2003.

[15] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw.*, 13(1):29–42, 2005.