Cocktail method for BitTorrent traffic identification in real time

Zhe Yang
School of Computer Science and Technology, Soochow University, Suzhou, P.R.C.
Email: yangzhe@suda.edu.cn

Lingzhi Li, Qijin Ji, Yanqin Zhu
School of Computer Science and Technology, Soochow University, Suzhou, P.R.C.
Email: {lilingzhi, ji, yqzhu }@ suda.edu.cn

***Abstract*—Peer-to-peer (P2P) applications generate a large volume of traffic and seriously affect quality of normal network services. Accurate identification of P2P traffic is especially important for network management. The simplest method is based on port mapping. But dynamic port technique makes it ineffective. Signature-based approach is useless when facing encrypted traffic. Recently, some approaches use more complex machine learning and data mining algorithms relying on flow statistics or host behaviors. Due to the sophisticated algorithms, they need a time-consuming process for training or calculating, they can hardly be used in real-time identification. In this paper, we propose a cocktail approach consists of three sub-methods to identify BitTorrent (BT) traffic. We apply application signatures to identify unencrypted traffic. And for those encrypted flows, we propose the message-based method according to the features of the message stream encryption (MSE) protocol. At last, we propose a pre-identification method based on signaling analysis. It can predict BT flows and distinguish them even at the first packet with SYN flag only. And we use modified Vuze clients to label BT traffic in real traffic traces, which help us to make high accuracy benchmark datasets to evaluate our approach. The results illustrate the effectiveness of our approach, especially for those un- or semi- established flows, which have no obvious signatures or flow statistics.**

***Index Terms*—Peer-to-peer, traffic identification, application signature, message stream, signaling analysis, benchmark dataset**

I. INTRODUCTION

In recent years, P2P protocol is widely used in many network applications, which generate 50-70% of the Internet traffic [1]. They greatly consume network bandwidth and seriously affect QoS of normal network activities. Thus, accurate identification of P2P traffic is especially important for network management and traffic optimization. But, P2P traffic is more difficult to be identified because of its quite different communication model. Furthermore, some newer-generation P2P applications apply various strategies to avoid detection, such as masquerade, obfuscation and encryption.

To identify the traffic of P2P applications, the simplest method is port-mapping on transport-layer. However, it becomes less accurate and ineffective, since P2P applications do not rely on those well-known ports anymore, and usually use dynamic port numbers. Another method is payload-based analysis, which searches packet payload for the application signatures or the sharing data blocks by TCP/UDP flows. But, this method is ineffective in identifying encrypted P2P traffic. Recently, some researchers try to use machine learning algorithms to identify P2P traffic with flow statistical information. But different flows, such as TCP and UDP, may have different statistical characteristics. And the statistical characteristics may be unstable because of the dynamic network situation. Instead of analyzing individual flows, a new method is based on the patterns of host behavior at the transport layer. It pays attention to all the flows generated by a specific host. But this method must gather information from several flows for each host before it can decide whether the host runs P2P applications. And like per-flow based method, it also needs a training process, and in many cases due to its sophisticated algorithms, it can hardly be used for real-time identification. Furthermore, for any identification method, benchmark traces containing prior labeled traffic of the P2P application are necessary to verify the effectiveness. But it is difficult to get 100% accuracy.

To resolve these problems, we propose a cocktail approach in our previous work [35], to identify BitTorrent (BT) traffic, the dominating one of P2P file-sharing applications, which has three sub-methods. This paper gives more explanations on identifying different kinds of BT traffic at the very beginning of or even before the BT flows generated.

a) Sub-method one: signature-based identification. It is used to identify the unencrypted BT traffic. Though BT applications have applied the message stream encryption (MSE) protocol to obfuscate the traffic, there are still over one third of them unencrypted [2]. So we recall this veteran, and call it *M1*.

Manuscript received December 27, 2010; revised March 24, 2011; accepted May 31, 2011.
Supported by the NSFC under Grant No. 61070170, the NSFC of Jiangsu under Grant No.BK2009589.
Corresponding author: Zhe Yang (yangzhe@suda.edu.cn).

doi:10.4304/jcp.7.1.85-95

b) Sub-method two: message-based identification. For those encrypted BT traffic, we resemble the bidirectional TCP flows into message streams. And if the direction and length of first three messages satisfy the criteria of MSE, this flow is identified as encrypted BT traffic. In our work, it is called *M2*.

c) Sub-method three: signaling-based pre-identification. Theoretically, *M1* and *M2* can identify the almost all BT traffic, un- and encrypted. But, matching the signatures or resembling the bidirectional flows are time-consuming tasks. And they may still miss the first few packets of the flows. Thus, we propose the pre-identification method based on signaling packets analysis, called *M3*. It can can predict BT flows and identify them at the first packet with *SYN* flag only.

Besides the approach itself, we also propose an easy method to make benchmark traces with almost 100% accuracy. Some modified Vuze clients are used to report how much traffic on earth is generated themselves.

In summary, the main contributions of this paper are as follows.

First, we propose a cocktail identification approach consists of three sub-methods. They combine with each other and aim at different BT traffic identification. Especially, the signaling packets are analyzed, which help us to predict the upcoming data flows. Results indicate that near 84% flows can be identified at the first packet and in 1ms, even they have not finished the 3-way TCP handshake. Actually, some of them are un- or semi-established TCP connections, which always are omitted by other approaches because they have no obvious signatures or flow statistics. Thus this cocktail approach achieves the ability of real-time identification with high accuracy and low overhead.

Second, we use modified Vuze clients to evaluate our approach, not only to generate real BT traffic, but also to label out these traffic in benchmark traces by themselves. It can reach near 100% accuracy and help us to well evaluate our cocktail approach.

The rest of the paper is organized as follows. We review the related works in Section 2. Section 3 explains our cocktail approach in details. In Section 4, we explain how to establish our experimental environment and define some metrics. And the evaluation results are given in Section 5. Finally, we discuss and conclude our work in Section 6.

II. Related Works

A. Port-based Identification

Earlier P2P applications, like many Internet services such as Web, Email and FTP, mostly run on the default ports, e.g. TCP 6881~6889 for BT, TCP 6346~6347 for Gnutella. So the simplest method for detection systems and firewalls to identify, control or even block the corresponding P2P traffic is based on these pre-defined ports [3-5]. But to circumvent the firewalls and detection systems, P2P applications intentionally disguise their traffic by using other applications' ports, even port 80, or random ports at each communication. Thus, this traditional port-based technique becomes less accurate and ineffective.

B. Payload-based Identification

Payload-based method searches the packet payload at application layer to find some common features.

On kind of feature are application signatures, which are the common strings in P2P protocols. These signatures often appear in the signaling and handshake packets before the data transmission. And they are useful in real time identification of network traffic [6-8]. This signature-based method achieves high accuracy once and is employed in some commercial network management products. However it also has some limitations. First, it is ineffective when P2P traffic is encrypted. Second, a lot of P2P applications use proprietary protocols. Lacking open protocol specifications makes finding appropriate signatures difficult. And maintaining up-to-date signatures for various newly emerged P2P applications are daunting tasks.

In addition to signature-based method, some other payload-based methods are proposed, which inspect the data transfer behavior in P2P applications. One basic observation is that a P2P peer uploads data to others after downloading it. Lu et al. [9] store the first *K* bytes of each packet in downloading flows for each host. When the same content is found in uploading flows of the host, these flows are identified as P2P flows. ACAS [10] uses the first *N* bytes of payload as input to train a machine learning model to classify flows. But only searching the some first bytes of payload to find the shared data is poor to identify some P2P applications. In [11], Xu et al. search shared data in the whole payload to solve the problem. They divide payloads of flows into data blocks. For flows sharing the same data blocks are identified as P2P traffic. Levchenko et al. [12] build several probabilistic models on payload, including the statistical model treating each n-byte flow distribution as a product of *N* independent byte distributions and the Markov process model which relies on introducing independence between bytes. These approaches are quite impractical in real-time identification systems. Because they will not identify any flows until the stored data in downloading flows reappears in uploading flows later. Storing and finding the shared data is memory- and time-consuming.

C. Flow-based Identification

Recently, some methods use machine learning algorithms to classify network traffic by using flow statistical information [13-23]. They develop discriminatory criteria based on statistical observations of various flow properties, such as the packet size distribution per flow, flow duration, mean inter-arrival times between packets, and number of upload/download packets/bytes, etc. And then they employ classification, clustering and other machine learning algorithms to assign flows to classes. The algorithms can be further summarized into supervised and unsupervised methods.

Moore et al. [14] apply Bayesian analysis techniques to categorize traffic by application. Roughan et al. [15] classify traffic flows into four classes suitable for quality

of service. They demonstrate the performance of nearest neighbor and linear discriminant analysis algorithms. Zander et al. [16] compare several supervised algorithms, including Naive Bayes, C4.5 decision tree, Bayesian Network and Naive Bayes Tree. They find that the classification precision of the algorithms is similar, but computational performance can be significantly different. In addition to these supervised classification techniques, unsupervised clustering methods are also used [17-20]. Campos et al. [21] apply some different hierarchical clustering methods to identify groups of similar communication patterns. Similarly, McGregor et al. [22] use the expectation maximum algorithm to cluster the flows. The experiment finds that the average precision of classification is very high, but some applications are difficult to distinguish.

Most of these methods only focus on the statistics of a single flow. And there are four challenges in identifying P2P traffic by using per-flow properties. First, in order to generate the appropriate traffic classes, these machine learning algorithms need a training process with labeled sample traces. Since network situations are quite different and dynamic, flow statistical characteristics in different sample traces are quite different. So the traffic classes relying on different sample traces may be not suitable for any networks in any time. Second, different flows in same application may have different statistics and even use different transport protocols. For example, in BT applications, some flows are used to get peer's information, some for negotiation between peers, and others for data transferring. Third, some flows may have no obvious or specific statistics. If only looking at per-flow statistics, flows belonging to different applications may have similar characteristics. Fourth, in many cases due to their sophisticated algorithms, they can hardly be used in real-time identification.

D. Behaviour-based Identification

Another identification approach is based on the patterns of host behavior or special communication patterns in transport layer. Instead of analyzing individual flows, it pays attention to all the flows generated by a specific host. Karagiannis et al. [23] attempt to capture the inherent behavior of a host at three levels: the social, functional and application levels. This approach mainly focuses on higher level communication patterns such as the number of source ports a particular host uses for communication. Constantinou et al. [24] investigated some fundamental characteristics of P2P applications, such as the huge network diameter and the presence of many hosts acting as both servers and clients, to identify P2P traffic. Xu et al. [25] use information theoretic and data mining techniques to build behavior profiles of Internet backbone traffic. Hu et al. [26] propose a method to build behavioral profiles of the target application which then describes the dominant communication patterns of the application.

However, this behavior-based method can not identify a single flow and is time-consuming, since it must gather information from several flows for each host before it can decide the role of a host. And it is hard to find a general pattern for all P2P applications. Like flow-based approaches, this method can hardly be used by real-time systems either.

E. Real-time Identification

In order to identify traffic in real time, some researchers try to only use the first *N* packets or first *M* bytes of each flow. Haffner et al. [10] automate the construction of protocol signatures by employing three supervised machine learning approaches on traffic containing known instances of each protocol, and relying solely on first 64 bytes of each flow. Bernaille et al [27] use the first 5 packets in each TCP flow to classifying traffic [18]. Li et al. apply decision tree algorithms using the first 5 packets of flows to identify the P2P traffic in earlier stage of the TCP connections. Ma et al. [28] develop three alternative mechanisms with statistical and structural content models for automatically classifying traffic into distinct protocols based on correlations between the packets with only first 64 bytes of each flow.

However these methods also have some limitations. The first *N* packets and first *M* bytes of each flow may have significant differences for the same P2P application, since some peers may re-send some packets because of the unreliable network situation. And the packets in the same position of the corresponding flows may also be different. Therefore in this case the positions of packets and bytes are not reliable information sources to identify a flow.

III. COCKTAIL IDENTIFICATION OF BT TRAFFIC

In this work, we combine three sub-methods as a cocktail approach to identify the traffic of BT applications, including signature-based, message-based and signaling-based pre-identification, donated as *M1*, *M2* and *M3* separately. In this section, we explain these three sub-methods.

A. M1: Signatures-based Identification

We first use traditional signature-based method to identify the unencrypted BT traffic. Though BT applications use the MSE protocol to obfuscate the traffic, there are still some parts of them unencrypted. According the comparative numbers in Germany and Southern Europe, the relative amount of unencrypted BT traffic in 2008 were still over 77% and 74% [1]. And in our pervious work [2], there are still have one-third of all BT clients establish unencrypted connections to others. So in this paper, we still recall this veteran, and call it *M1*.

The BT protocol signatures are identified from previous studies [7, 23], public BT protocol specifications [29], and by reverse engineering. The signatures for different BT protocols are represented by regular expressions, as shown in Table 1. Besides the common used signature "^\x13BitTorrent protocol" for identifying the data transferring protocol (peer wire), we pay more attention to another three protocols. Although they will not generate as much traffic as peer wire protocol, they play the critical roles in BT applications. They are necessary to maintain the BT system and exchange the

TABLE I.
APPLICATION SIGNATURES OF BITTORRENT PROTOCOLS

Protocol Name		Signatures	Protocol
Peer Wire		^\x13BitTorrent protocol	TCP
TCP Tracker		^get .*announce?info_hash	TCP
UDP Tracker		^\x00 00 04 17 27 10 19 80	UDP
DHT	request	^d1:a	UDP
	response	^d1:r	
	error	^d1:e	

peer information for data transferring, we call them signaling protocols. The purpose that we use these signaling protocols' signatures is not only for identifying their traffic, but also for obtaining the peers' information which is used by *M3*.

B. M2: Message-based Identification

Second, we use message-based method to identify the encrypted BT traffic, we call it *M2*. It is similar to the flow-based and behavior-based methods, but it still has some differences. Before we explain the *M2* in detail, we first discuss the MSE protocol.

1) MSE protocol analysis

MSE [30] is designed to provide security features to the BT at a lower layer by acting as a wrapper. It behaves randomly, both in terms of flow behavior (such as packet sizes) and in terms of application data. This makes it almost impossible to detect the MSE protocol with flow-based methods. Two communicating peers start the handshake of MSE protocol after a completed 3-way TCP handshake. The handshake of MSE is separated into 5 blocking steps, as shown in Fig. 1.

To achieve complete randomness from the first byte, initiating peer (denoted as A) and responding peer (denoted as B) use a D-H key exchange [31], which uses large safe prime P, to create a session key in step 1 and 2. After negotiating the D-H key, the rest of the handshake is encrypted using RC4 [32], with the first 1024 keystream bytes discarded. The RC4 key is based on both the D-H key and a weak shared secret which is the "InfoHash" value transmitted in the .torrent file. The payload stream is either encrypted using RC4 or sent in plaintext. The desired payload delivery method is negotiated in step 3 and 4. In step 3, A sends the method it is willing to use in *CryptoProvide* and B selects the desired method in *CryptoSelect*. Although MSE protocol makes it no chances to be detected, there are some distinctive properties that can be observed in the handshake process, especially in the first three steps.

$$1.\ A \to B\ :\ g^a \bmod P, PadA$$
$$2.\ B \to A\ :\ g^b \bmod P, PadB$$
$$3.\ A \to B\ :\ H('req1',S), H('req2',Skey) \oplus H('req3',S),$$
$$E(VC, CryptoProvide, len(PadC), PadC, len(IA)), E(IA)$$
$$4.\ B \to A\ :\ E(VC, CryptoSelect, len(PadD), PadD), \hat{E}(Payload\ Stream)$$
$$5.\ A \to B\ :\ \hat{E}(Payload\ Stream)$$

Figure 1. Handshake of MSE

In first step, peer A starts the MSE handshake by sending a D-H public key Y_a ($Y_a = g^a \bmod P$) and some padding (*PadA*) to peer B. In MSE, g is a generator and it is used to generate Z_P^* with safe prime P and its value is 2. a is a D-H exponent, which is a variable size random integer and MSE recommend 160 random bits. P is a 768 bit safe prime. Thus, the size of Y_a is 96 bytes (768/8=96 bytes) and its value is not bigger than P. And *PadA* is random data with a random length of 0-512 bytes. So, the length of message sent from A to B in step 1 will always be in the range of 96 to 608 bytes.

Next, in step two, peer B responds with its D-H public key Y_b ($Y_b = g^b \bmod P$) and some padding (*PadB*). Similarly, the length of message reply from B to A in step 2 is also in the range of 96 to 608 bytes.

In the message sent by peer A in step 3, *H* is the SHA-1 hash function with 20 bytes binary outputs. *S* is the negotiated D-H key. *Skey* is the weak shared secret key mentioned before. *E*() is RC4 encryption, with key *H*('*keyA*', *S*, *SKey*) for A and *H*('*keyB*', *S*, *SKey*) for B. *VC* is a verification constant, but in this version of BT protocol it is a string of 8 bytes set to 0x00. *CryptoProvide* and *CryptoSelect* are a 4-byte bitfields. *len*(*X*) specifies the length of *X* in 2 bytes. *PadC* and *PadD* are zero-valued padding of 0–512 bytes. *IA* is the initial payload data from A and it is considered as atomic. Thus there must be no blocking operations before *IA* is completely transmitted. But after that, a block occurs since A waits for B to send next message before A continues to send his bitfield, thus *IA* only includes the following fields (in order): PStrLen (1 byte), PStr (19 bytes), Reserved (8 bytes), InfoHash (20 bytes) and PeerID (20 bytes). Thus the length of *IA* is 68 bytes. So, the total length of the third message is in the range of 124 to 636 bytes.

2) MSE message resembling

Although, the first three messages of MSE handshake have some distinctive properties in length. But unfortunately, each of them is not transmitted in one packet. It is often divided into several packets and the payload size is around 100 bytes per packet. Thus we

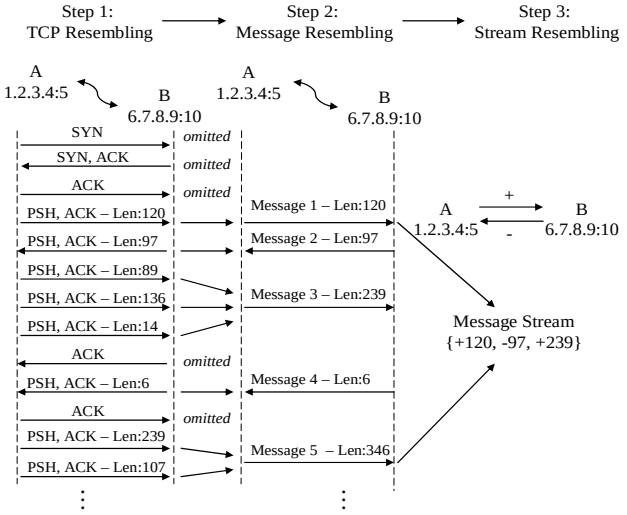

Figure 2. MSE message resembling

need to resemble packets in a TCP flow into messages first, and then determine whether this flow belongs to encrypted BT traffic according to direction and length of first three messages. Fig. 2 describes the main steps.

Step 1: TCP Resembling. First we group the traffic packets into TCP flows with the four tuples {*SrcIP*, *SrcPort*, *DestIP*, *DestPort*}. The source and destination of the flow is defined as the source and destination IP address of the flow's first packet.

Step 2: Message Resembling. Because MSE handshake messages are often divided into several packets, so we merge first few packets into messages according to the direction and the flags. In each flow, we only take the packets with the *PSH* flag into consideration. While those *ACK*, *RST* and *SYN* packets are omitted. We merge the *PSH* packets with same direction into one message, until meet the next reverse *PSH* packet. And the message length is the sum of payload length of these packets.

Step 3: Stream Resembling. Finally, we arrange these messages into a stream. A message stream is defined as a bidirectional, ordered sequence of messages. The source and destination of a stream is same as the flow. In each stream, if a message is sent from the source to the destination, then the message direction is positive and denoted as "+". Similarly, the message direction is negative if the message is sent from the destination to the source and denoted as "-". And because only the first three messages in MSE handshake have distinctive properties in length, so the message stream only include the first three messages in each flow.

Now the flow form A (1.2.3.4:5) to B (6.7.8.9:10) can be described in a message stream {+120, -97, +239}. Then we compare it with the stream of a normal MSE handshake process. According to aforementioned analysis, the stream of a normal MSE handshake should be {+[96,608], -[96,608], +[124,636]}. Thus, this flow between A and B can be identified as an encrypted BT traffic.

C. M3: Signaling-based Pre-identification

For a distributed application system, like BT, the most important thing to keep the system alive and running well is that all users may discover each other easily. So, the system must have a mechanism to maintain the information of all users. In BT, a peer can obtain the list of other pees from trackers, DHT and by peer exchange (PEX) protocol [34]. Thus, if any identification method can grasp the list of peers, it may find the Achilles heel.

1) Peer list from tracker

Tracker is very critical for BT. After downloading the .torrent file, peers contact the specified tracker to request the current list of peers. The request and response

```
0000  00 22 64 6b fb f9 00 1e  08 08 0c 60 08 00 45 00   ."dk.... ...`..E.
0010  02 d3 b2 f5 40 00 2e 06  80 9a da ca e3 1b c0 a8   ....@... ........
0020  98 06 1f 95 04 83 6a d0  44 1c 92 9d 31 87 50 19   ......j. D...1.P.
0030  19 20 82 2f 00 00 48 54  54 50 2f 31 2e 30 20 32   . ./..HT TP/1.0 2
0040  30 30 20 4f 4b 0d 0a 0d  0a 64 31 30 3a 64 6f 6e   00 OK... .d10:don
0050  65 20 70 65 65 72 73 69  35 32 34 65 38 3a 69 6e   e peersi 524e8:in
0060  74 65 72 76 61 6c 69 31  38 30 30 65 39 3a 6e 75   tervali1 800e9:nu
0070  6d 20 70 65 65 72 73 69  32 31 35 36 65 35 3a 70   m peersi 2156e5:p
0080  65 65 72 73 36 30 30 3a  3d b2 15 9f 00 50 7d 27   eers600: =....P}'
0090  3f 02 00 50 75 41 2d d5  00 50 72 eb 54 2a 00 50   ?..PuA-. .Pr.T*.P
00a0  de 5e de b9 41 f1 de 47  82 8d 46 f4 4a 0c 81 70   .^..A..G ..F.J..p
00b0  40 3f de de 4d 66 00 50  45 56 7b d9 00 50 da 0d   @?..Mf.P EV{..P..
00c0  ae 9b 00 50 3b bc 99 9a  7f b3 76 75 0a 25 34 c9   ...P;... ..vu.%4.
```

Figure 3. Peer list from tracker over TCP

TABLE II.
STRUCTURE OF ANNOUNCE RESPONSE PACKET

Offset	Size	Name	Value
0	32-bit integer	action	1(announce)
4	32-bit integer	transaction_id	
8	32-bit integer	interval	
12	32-bit integer	leechers	
16	32-bit integer	seeders	
20+6*n	32-bit integer	IP address	
24+6*n	16-bit integer	TCP port	

message between peers and trackers is transferred by TCP or UDP protocol.

The TCP based tracker responds peer's GET requests with "text/plain" document consisting of a bencoded dictionary which has the following parameters: failure reason, interval, complete, incomplete, peers, etc. The value of parameter of "5:peers" is a list of dictionaries containing a list of peers. To reduce the size of responses and to reduce memory and computational requirements, trackers use a compact format of peer list, where each peer is represented using 6 bytes. The first 4 bytes contain the 32-bit IPv4 address. The remaining two contain the port number. Both address and port use network-byte order. So if we identify a TCP flow as a TCP_Tracker flow, we mark it. And after some time, if we receive a reverse packet in same flow which containing the parameter of "5:peers", we will obtain the list of peers. For example, in the TCP packet showed in Fig. 3, we match the parameter "5:peers". And behind the ":", there is a 600 bytes list of peers. Each peer is represented using 6 bytes. So we can find the first two peers are "0x 3d b2 15 9f 00 50" (61.178.21.149:80) and "0x 7d 27 3f 02 00 50" (125.39.63.2:80).

In UDP based tracker protocol, there are several types of packets, including connect request, connect response, announce request, announce response, scrape request, scrape response, error, etc. The list of peers is only in the announce response packets from tracker to peers. The structure of the announce response packets is shown in Table 2. And we can find that the value of the first 4 bytes always equals 1, according to the protocol specification. It can be used as an additional signature to identify the announce response packets. But this signature is too ordinary, and may introduce lots of false positive. Thus we first use the signature "^\x00 00 04 17 27 10 19 80" to identify a UDP flow as an UDP_Tracker flow. And if we receive a reverse packet in the same UDP flow and its first 4 bytes are "0x 00 00 00 01", we extract the list of peers from the twentieth bytes to the end.

2) Peer list from DHT

Because of the importance of tracker in BT, so it is easy for ISP to limit the BT traffic by block the trackers. To implement the trackerless system, BT uses a DHT for storing peer's information. The DHT protocol is a simple RPC mechanism consisting of bencoded dictionaries sent

```
0000  00 1e 08 08 0c 60 00 16  47 88 52 bf 08 00 45 00   .....`.. G.R...E.
0010  00 9b 4c d2 00 00 6b 11  12 5a be e0 d8 94 c0 a8   ..L...k. .Z......
0020  98 08 58 bd 1a e1 00 87  56 ae 64 31 3a 72 64 32   ..X..... V.d1:rd2
0030  3a 69 64 32 30 3a aa 5a  23 ad 0c e8 9e 0f b1 df   :id20:.Z #.......
0040  4a 95 bb 24 88 f0 ae 83  08 de 35 3a 74 6f 6b 65   J..$.... ..5:toke
0050  6e 32 30 3a 82 ea b5 bb  69 67 aa 23 26 b5 7e 39   n20:.... ig.#&.~9
0060  62 7f 1b d1 fc 1f fc 32  36 3a 76 61 6c 75 65 73   b......2 6:values
0070  6c 36 3a da 04 bd 17 1a  e1 36 3a 44 70 f1 b2 26   l6:..... .6:Dp..&
0080  e8 36 3a 7c 08 4f b0 36  35 36 3a 75 54 61 b6 00   .6:|.O.6 56:uTa..
0090  50 36 3a 3c df ff 05 53  80 65 65 31 3a 74 32 3a   P6:<...S .ee1:t2:
00a0  05 68 31 3a 79 31 3a 72  65                       .h1:y1:r e
```

Figure 4. Peer list from DHT over UDP

over UDP. A single query packet is sent out and a single packet is sent in response. There are three message types: query, response, and error. And there are four types of queries: ping, find_node, get_peers, and announce_peer. The list of peers is only in the response packet to "get_peers" query. If a queried node knows some peers, it returns the compact format peer information in a parameter "6:values" as a list of peers. So we use the signature "d1:r" to identify a UDP packet as a DHT response message. And if it has the parameter of "6:values", we will get the list of peers. For example, in the UDP packet showed in Fig. 4, we match the signature "d1:r", so it is a DHT response message. And we also match the parameter "6:values". And behind it, there is a list containing 5 peers. Each peer is represented using 6 bytes. So we can find the first two peers are "0x da 04 bd 17 1a e1" (218.4.189.23:6881) and "0x 44 70 f1 b2 26 e8" (68.112.241.178:9960).

3) Peer list from PEX

Most BT clients also use PEX protocol [33] to gather peers information more swiftly and efficiently. PEX greatly reduces the reliance of peers on a tracker by allowing each peer to directly update others in the swarm. PEX messages are exchanges periodically over peer wire protocol, which are bencoded dictionaries containing a list of peers to be added and removed. The parameters of "added" and "added6" are used to add one or more IPv4/6 addresses. In this work, we only consider the "added" field that contains the IPv4 addresses. To avoid the confusion with "added6", we use the string "5:added" instead of "added" to locate the peer list. Thus we first use the signature "^\x13BitTorrent protocol" to identify a TCP flow as a Peer_Wire flow. And in the same TCP flow, if we receive any packets containing the string of "5:added" in application payload, we extract the peers information.

4) Pre-identification of BT traffic

After obtaining the list of peers from tracker, DHT and PEX, we store them into a hash table, named list of pre-identified flow. The hash value of two tuples {*IP*, *Port*} of each peer extracted from the peer list is calculated with MD5 algorithm. For any newly arrived TCP flow, if the hash value of {*DestIP*, *DestPort*} has already in the list of pre-identified flow. This flow should be marked as BT traffic. But with the time elapsing, we will get more and more list of peers, thus list of pre-identified flow will expand rapidly. So some records in the list should be eliminated periodically. We used the LRU (Least Recently Used) algorithm to update the list of peers, and the update interval is an empirical value and will be discussed in section IV.

IV. Experimental Setup

In this section, we explain how to establish our experimental environment, make the benchmark traces and setup the update interval of list of pre-identified flow.

A. Environment and Traces

Another important point in traffic identification is to verify the approaches with benchmark traces containing accurate labeled traffic. Basically there are two methods to obtain traffic traces. One is to capture real traffic from the Internet and then label the target application with third party tools. But it is very difficult to reach 100% accuracy. Another method is to get self-generated traffic traces of the target applications in a controlled environment. This method can get higher accuracy.

Thus, we collect 7 real traffic traces at the up-link line of our lab, a total controlled environment, including self-generated BT traffic with modified Vuze clients [2]. In each trace, we used 10 computers to run the modified Vuze clients with 5 different .torrent files each to generate the real BT traffic, 5 for unencrypted and 5 for encrypted. The download speed is limited to 1KB/s and the upload speed, 10KB/s. We use these modified Vuze clients not only to generate real BT traffic, but also to enable them to label the BT traffic they generated. And we use another 10 computers to generate background traffic, such as HTTP(S), FTP, DHCP, SMTP and POP3, etc. In all traces, there are little packets generated by Windows OS or system services. All these packet traces are collected at the uplink line of our lab. For privacy, we anonymized all the IP addresses. Those .torrent files running on modified Vuze clients come from the www.torrentz.com, and belong to five categories: movie, TV, music, game and anime. And in each trace, every client randomly selects 5 different .torrent files.

Then we divided these 7 traces into two types, 2 for configuration and 5 for evaluation. The two configuration traces were collected from March 1 to 5, 2010 (denoted as C1) and March 11 to 15 (C2). We first apply our cocktail approach on these configuration traces offline to determine the update interval of list of pre-identified flow. And the five evaluation traces are collected from April 1 to 9, 2010 (denoted as T1), April 11 to 19 (T2), April 21 to 29 (T3), May 1 to 9 (T4) and May 11 to 19 (T5). After collecting these real traffic traces, we label them as benchmark traces.

B. Benchmark Traces

Like other research works, well labeled benchmark traces are very critical to verify the effectiveness of our approach. Because there are few benchmark traces containing accurate labeled traffic of the target BT applications, so we use two methods to label the BT traffic in real traffic traces, including signature-based methods and manual analysis.

In the signature-based method, the signatures we used are listed in Table 1. In this step, we process traces with Bro [35], an open-source network intrusion detection system. We added functions to Bro to match the signatures in packet payloads and label the corresponding

TABLE III.
NUMBER OF FLOWS/PACKETS/BYTES IN THE BENCHMARK TRACES

Protocol		C1	C2	T1	T2	T3	T4	T5
BT	flows	472k	597k	604k	815k	773k	761k	702k
	packets	19,826k	21,837k	20,375k	18,223k	23,148k	12,492k	14,910k
	bytes	4,133M	2,555M	4,107M	5,497M	5,644M	3,371M	5,111M
non-BT	flows	176k	257k	342k	297k	425k	259k	393k
	packets	10,862k	8,093k	7,295k	9,745k	8,624k	7,065k	7,058k
	bytes	2,663M	1,129M	1,454M	2,992M	2,787M	2,242M	2,327M

flows. Thus, the flows of unencrypted data transferring, TCP-Tracker, UDP-Tracker and DHT can be labeled accurately.

But not all BT flows have traceable signatures, especially for those encrypted BT traffic. And because there still have no well accepted methods, which can identify the encrypted BT flows with 100% accuracy, we do it manually. Actually, the best way to label the traces with high accuracy is to let the BT applications tell us how much traffic in the traces on earth is generated by themselves. So we label the traces with the help of those modified Vuze clients we used to generate the real BT traffic. We choose Vuze, formerly Azureus, because it is the most popular open source client with cross-platform compatibility and features a powerful API for dynamically adding new functionality. Like other ordinary BT clients, these modified Vuze clients can parse the .torrent files, query and connect to other peers, and download the sharing files. Moreover, we enable them to report some information of all other peers, which they try to connect to or receive the connections from.

But they only report the IP address and port of peers, so we still can not labeled BT flows directly. Thus we need further processing at these traffic traces. First we resemble the packets into flows according to the five tuples {*SrcIP*, *SrcPort*, *DestIP*, *DestPort*, *Protocol*}. For each flow, if the {*SrcIP*, *SrcPort*} or {*DestIP*, *DestPort*} is reported by modified Vuze clients, thus we label this flow as BT traffic. For example, if a modified Vuze client reports that it connected to a peer at {1.2.3.4, 5}. And in real traces, there are two flows {6.7.8.9, 10, 1.2.3.4, 5, TCP} and {10.9.8.7, 6, 1.2.3.4, 5, TCP}. Thus, we label these two flows as BT traffic.

And for those background traffics, such as HTTP(S), FTP, DHCP, SMTP, POP3, and those generated by Windows OS, system services and even known applications, we all labeled them as non-BT traffic. Finally, we labeled the two configuration traces, C1 and C2, and five evaluation traces, T1 to T5, as our benchmark traces. And the summary of these traces is given in Table III.

C. Metrics Definitions

We use three kinds of metrics to evaluate the accuracy of our method at the level of flow, packet and byte. First, we give the metrics of flow with examples. For example, if there are *n* flows as total in a traffic trace. So if a flow is identified as the BT flow but in fact it does not belong to it, we call it a false positive and use n_{fp} denotes the total number of false positive flows. Similarly if a BT flow is not identified, we call it a false negative and use n_{fn} denotes the total number of false negative BT flows.

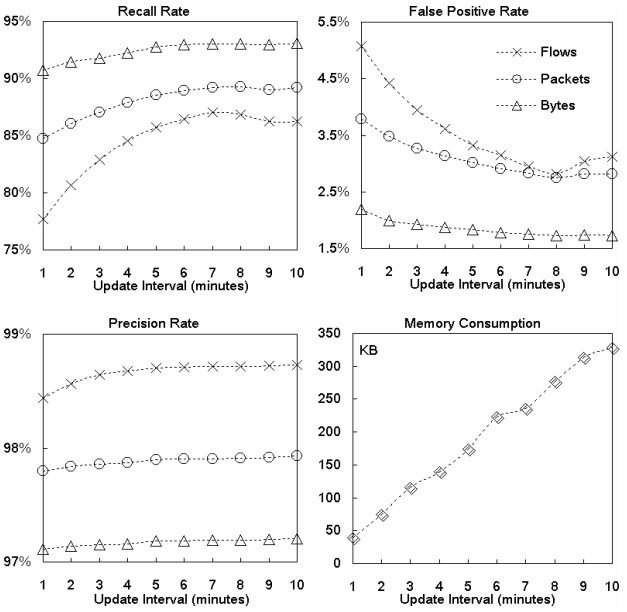

Figure 5. Update interval setup

And if it is correctly identified, we call it a true positive and use n_{tp} denotes the total number of true positive BT flows.

So the fist metric we used is the *recall rate of flows* (R_f). It is used to measure the portion of the flows that can be identified correctly by our method out of the all flows belonging to the BT application.

$$R_f = n_{tp} \Big/ \left(n_{fn} + n_{tp} \right) \tag{1}$$

The second one is the *precision rate of flows* (P_f). It is the number of the true positive flows as the fraction of the total flows we identified

$$P_f = n_{tp} \Big/ \left(n_{tp} + n_{fp} \right) \tag{2}$$

The third one if the *false positive rate of flows* (FP_f). It is used to measure the portion of non-BT flows but identified as BT flows out of the all non-BT flows.

$$FP_f = n_{fp} \Big/ \left(n - n_{fn} - n_{tp} \right) \tag{3}$$

Similarly, we use R_p, P_p and FP_p denote the three metrics of packet, R_b, P_b and FP_b for byte.

D. Update Interval Setup

We first applied our approach on configuration traces offline. And after considering the recall, precision, false positive and the memory consumption of list of pre-identified flow, we determined the update interval of list of pre-identified flow. Fig 5 shows the average value of four metrics on two configuration traces, C1 and C2. And we can find that when the update interval is 7 minutes, the *recall rates* of flows, packets and bytes can reach the maximum at the same time. While the *false positive rates* of flows, packets and bytes all reach the minimum when the update interval is set as 8 minutes. The *precision rates* are not influenced obviously by the parameter of update interval. But when the interval increases, the memory consumption of list of pre-identified flow increases simultaneously. It is almost a linear increase. A good identification approach should have low *false positive*

rate, high *precision rate* and *recall rate*. So we set the update interval at 8 minutes in following evaluation experiments.

V. Evaluation Results

In this section, we present the results of identifying BT traffic in the evaluation traces, T1~T5. Among our three sub-methods, *M1* and *M2* are enough to identify all BT traffic theoretically. And they may still miss some flows or some packets in the flows. To illustrate the effect of our cocktail approach, we first use *M1+M2* methods to identify the T1~T5 traces. And then we use cocktail approach (*M1+M2+M3*) to identify these traces again. The update interval is 8 minutes.

A. Recall Rates

Fig.6 shows the recall rates in trace T1~T5. Our cocktail approach can achieve very high recall rate, no matter what identification level is concerned. At the level of flows (R_f), cocktail approach can identify 80%~90% BT flows, while the *M1+M2* only identify 22%~39% flows. But at the level of packets and bytes, it looks like that *M1+M2* can achieve better performance. The recall rate of packets and bytes, R_p and R_b, are between 55%~72% and 70%~86%. While R_p and R_b of our approach are still between 85%~95% and 88%~98%. The reason lies in those un- or semi- established BT flows. For BT clients, after they obtain the list of peers they will try to connect the remote peers with the TCP handshake and then BT handshake or MSE handshake. Among these flows, part of them may have only a *SYN* packet and will not get any response for all sorts of reasons. And another part of them may establish the TCP connections, but can not establish BT or MSE connections. Thus in these flows, there are no application signatures or any MSE features. Thus the *M1+M2* will miss them, which are also missed by most of existing approaches. While our approach, especially *M3*, can foresee these BT flows by analyzing the signaling. So we will not miss them, although there are few packets or bytes belong to them. Therefore, no matter what identification level is concerned, our approach is stable.

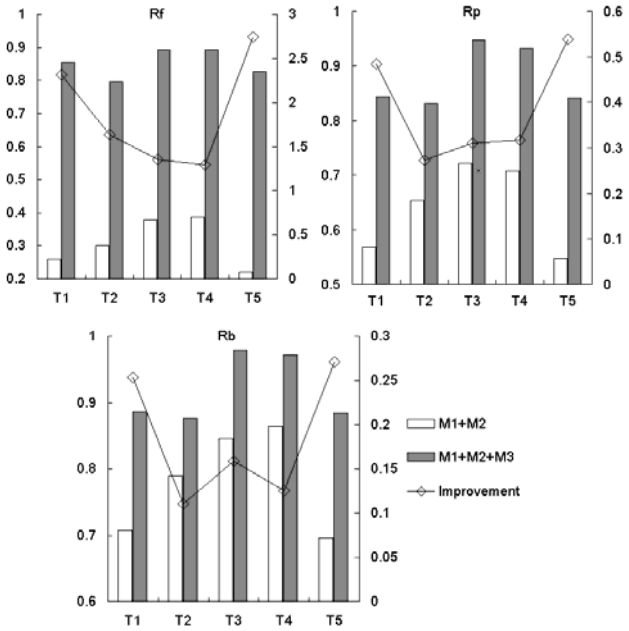

Figure 6. Recall Rates

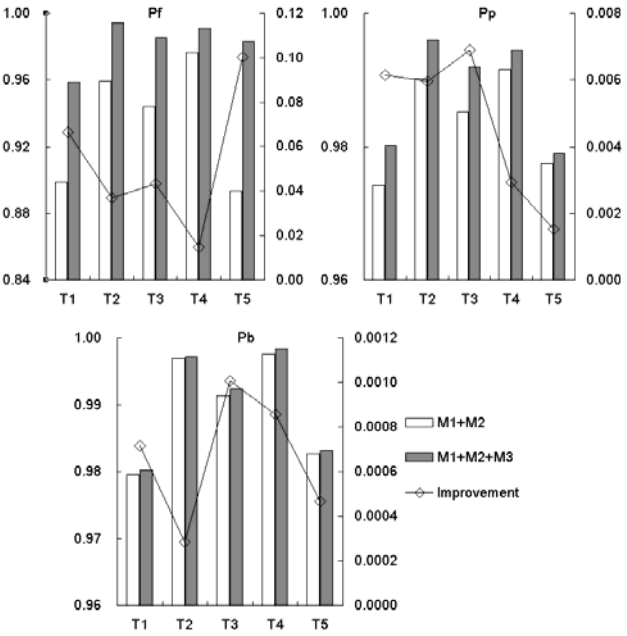

Figure 7. Precision Rates

B. Precision Rates

Fig.7 gives the data of precision rates. It looks like a draw between cocktail approach and *M1+M2*, especially at the level of bytes. The *M1+M2* have achieved high precision rates already. The P_f is between 89.35~97.65%, the average is 93.44%. The P_p and P_b are between 97.42~99.16% (98.37%) and 97.95~99.69% (98.96%) separately. While the P_f, P_p and P_b of cocktail approach are 95.87~99.46% (98.26%), 97.90~99.60% (98.83%) and 98.02~99.84% (99.03%) separately.

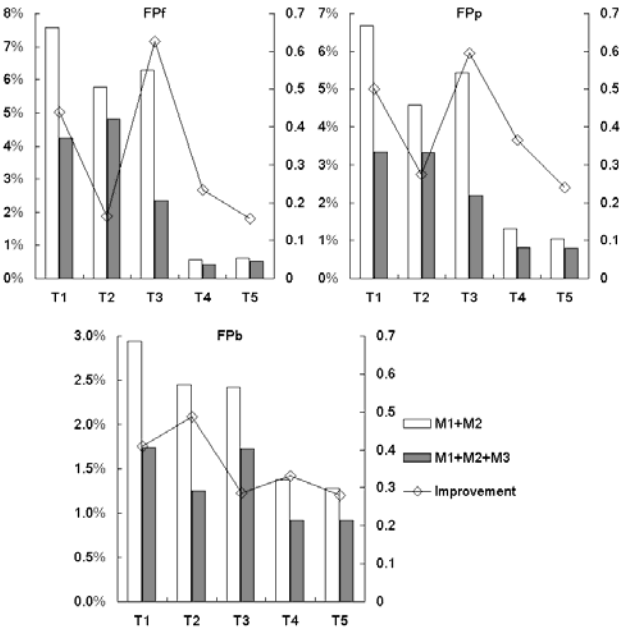

Figure 8. False Positive Rates

C. False Positive Rates

Fig.8 gives the data of false positive rates. And we can find that on all traces our approach can efficiently decrease the false positive rates. The FP_f of *M1+M2* on five traces is between 7.58~0.56%, the average is 4.16%. The FP_p and FP_b are between 6.68~1.05% (3.81%) and 2.94~1.28% (2.09%) separately. While the FP_f, FP_p and FP_b of cocktail approach can drop to 4.81~0.43% (2.47%), 3.34~0.80% (2.10%) and 1.74~0.92% (1.31%) separately. But our approach still introduces few false positives which are caused by HTTPS and SMB (Server Message Block) protocols. It is easy to distinguish these false positive flows by port-mapping, because they are running on fixed port of 443 and 445.

D. Real-time Identification

Now, we evaluate our approach the ability of real-time identification, as shown in Fig. 9 and Fig. 10.

In Fig. 9, we find almost 90% flows can be identified at the first packet, which is the TCP *SYN* packet. Because most upcoming flows are predicted by *M3* with the list of peers extracting from the signaling packets. And another peak appears at the point of 4 on x-axis. It means about 5% flows can be identified at the fourth packet. They are unencrypted BT flows, and the *M1* match the signatures in the first BT handshake packet after 3-way TCP handshake, which is the fourth packet of that flow. The remaining flows are left to *M2*. Most of them can be identified before the twelfth packet. So if a flow can not be identified in 12 packets, it has few chances, less than 2%, to be a BT flow. Thus, the weighted average number of packets should be checked by cocktail approach before the flow is determined as BT flow or not is 1.66.

If we consider the time before a flow can be identified, we find that over 85% flows can be identified in 100 ms, even in 1ms, as shown in Fig. 10. That is to say they are identified as soon as they are generated. This all contributes to the *M3* by wiretap the signaling. The most of remaining flows will be identified in 1000 ms by *M1* and *M2* together. And another interesting phenomenon is that the handshake process of some flows may continue a relative long time before the packet containing the signatures is transmitted or two peers finish the MSE handshake. It looks like a long tail. To identifying these flows, we must wait and store the necessary information of them, such as five tuples of flows, which occupy the resource of CPU and memory. It will greatly degrade the real-time identification ability of our approach. So if a flow can not be determined in 1000 ms, it should be omitted or treated as none BT traffic. But it will cause little false negatives.

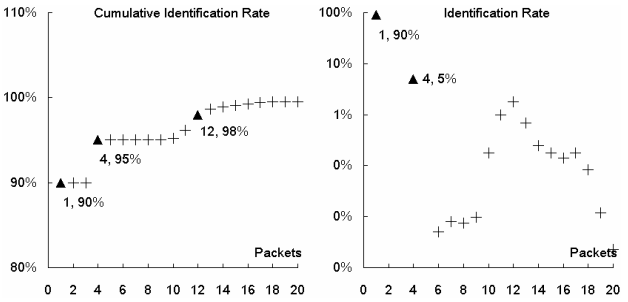

Figure 9. Identification cost on packets

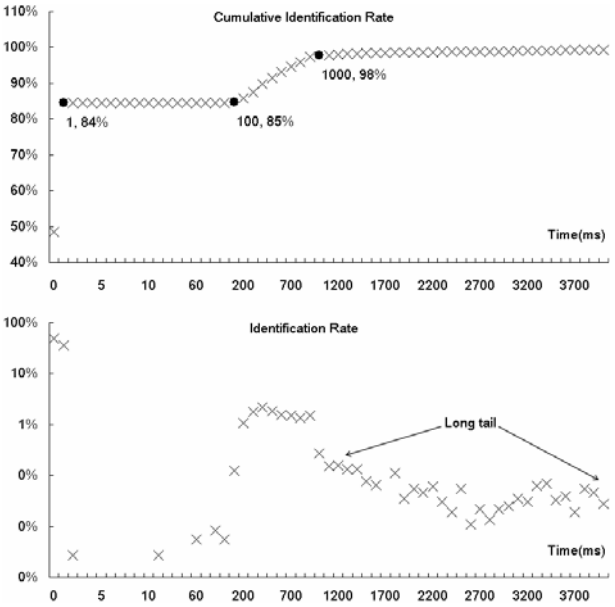

Figure 10. Identification cost on time

VI. Conclusions And Future works

P2P traffic identification is a hot spot of research in recent years. Lots of creative approaches are proposed from different viewpoints. But the real-time identification is still a hard task. In this work, we a cocktail approach consists of three sub-methods to identify BT traffic in real-time. These sub-methods combine with each other and aim at identifying different kinds of BT traffic at the very beginning of or even before the traffic generated. Results indicate that near 84% flows can be identified at the first packet and in 1ms. And to evaluate our approach, we use modified Vuze clients, not only to generate real BT traffic, but also to label out these traffic in benchmark traces by themselves. It can reach very high accuracy and help us to well evaluate our cocktail approach. Thus this cocktail approach achieves the ability of real-time identification with high accuracy, low overhead.

And next, we plan to model the BT signaling process to reduce the overhead our approach further.

Acknowledgment

We would like to thank the Suzhou Key Laboratory of Converged Communications for the assistance in deploying experiment environment. And this work is supported in part by the projects of Suzhou Science and Technology Department under grant No.SZS0805, No.ZXG0825, SYJG09024 and SYG201034.

References

[1] IPOQUE. Internet Study 2008/2009. *http://www.ipoque.com/resources/ internet-studies/ internet-study-2008_2009.*

[2] Z, Yang, L.Z. Li, Z.H. Wang and L. Q. Li. Active analysis of BT with a modified Azureus client, in proceedings of the *10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, 2010, pp. 371-376.

[3] S. Sen and J. Wang, Analyzing peer-to-peer traffic across large networks, *Transactions on Networking*, 2004, 12(2), pp.219-232.
[4] A. Gerber, J. Houle, H. Nguyen, M. Roughan, and S. Sen, P2P The Gorilla in the Cable, in *National Cable &Telecommunications Association 2003 National Show*, Chicago, IL, 2003.
[5] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, An Analysis of Internet Content Delivery Systems, in proceedings of the *5th Symposium on Operating Systems Design and Implementation*, Boston, MA, 2002, pp. 315-328.
[6] A.W. Moore, K. Papagiannaki, Toward the Accurate Identification of Network Applications, in proceedings of the *6th International Workshop on Passive and Active Measurement*, Boston, MA, 2005, pp. 41-54.
[7] S. Sen, O. Spatscheck, D. Wang, Accurate, scalable in-network identification of P2P traffic using application signatures, in proceedings of the *13th international conference on World Wide Web*, New York, USA, 2004, pp. 512-521.
[8] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, Is P2P dying or just hiding? in proceeding of the *47th annual IEEE Global Telecommunications Conference*, Dallas, USA , 2004, pp. 1532-1538.
[9] X. Lu, H. Duan, X. Li, Identification of P2P traffic based on the content redistribution characteristic, in proceedings of the *International Symposium on Communications and Information Technologies*, Sydney, NSW, 2007, pp. 596-601
[10] P. Haffner, S. Sen, O. Spatscheck, D. Wang, ACAS: automated construction of application signatures, in proceedings of the *ACM SIGCOMM Workshop on Mining Network Data*, New York, USA, 2005, pp. 197-202.
[11] K. Xu et al., Identify P2P traffic by inspecting data transfer behavior, *Computer Communications*. 2010, 33(10), pp.1141-1150
[12] J. Ma, K. Levchenko, C. Kreibich, S. Savage, G.M. Voelker, Unexpected means of protocol inference, in: proceedings of the *6th ACM SIGCOMM Conference on Internet Measurement*, New York, USA, 2006, pp. 313-326.
[13] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, Transport layer identification of P2P traffic, in proceedings the *4th ACM SIGCOMM Conference on Internet Measurement*, Taormina, Italy, 2004, pp. 121-134.
[14] A. Moore and D. Zuev, Internet traffic classification using Bayesian analysis, in proceedings of the *2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Alberta, Canada, 2005, pp. 50-60.
[15] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification, in proceedings of the *4th ACM SIGCOMM Conference on Internet Measurement*, Taormina, Italy, 2004, pp. 135-148.
[16] N. Williams, S. Zander, G. Armitages, A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification, *SIGCOMM Computer Communication Review*, 2006, 36 (5), pp.5-16.
[17] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, Offline/realtime traffic classification using semi-supervised learning, *Performance Evaluation*, 2007, 64 (9-12), pp. 1194-1213.
[18] L. Bernaille, R. Teixeira, K. Salamatian, Early application identification, in proceedings of the *2nd ACM Conference on Future Networking Technologies*, Lisboa, Portugal, 2006, pp. 436-455.
[19] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofanelli, Revealing skype traffic: when randomness plays with you, in proceedings of the *SIGCOMM 2007*, Kyoto, Japan. 2007, pp. 37-48
[20] M. Crotti, F. Gringoli, P. Pelosato, L. Salgarelli, A statistical approach to ip-level classification of network traffic, in proceedings of the *IEEE International Conference on Communications*, Istanbul, Turkey, 2006, pp. 170-176.
[21] F. H. Campos, F.D. Smith, K. Jeffay, A.B. Nobel. Statistical clustering of internet communication patterns, in proceedings of the *35th Symposium on the Interface of Computing Science and Statistics*, Salt Lake City, UT, 2003, pp. 134
[22] A. McGregor, M. Hall, P. Lorier, J. Brunskill, Flow clustering using machine learning techniques, in proceedings of the *5th International Workshop on Passive and Active Network Measurement*, Antibes Juan-les-Pins, France, 2004, pp. 205-214.
[23] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, BLINC: multilevel traffic classification in the dark, in proceedings of the *ACM SIGCOMM 2005*, Philadelphia, PA, 2005, pp. 229-240.
[24] F. Constantinou and P. Mavrommatis, Identifying Known and Unknown Peer-to-Peer Traffic, in proceedings of the *5th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, 2006, pp. 93-102.
[25] K. Xu, Z.L. Zhang, S. Bhattacharyya, Profiling internet backbone traffic: behavior models and applications, in proceedings of the *ACM SIGCOMM 2005*, New York, USA, 2005, pp. 169-180.
[26] Yan Hu , Dah-Ming Chiu, John C.S. Lui. Profiling and identification of P2P traffic. *Computer Networks*, 2009, 53 (6), pp. 849-863
[27] J. Li, S. Zhang, Y. Lu, and J. Yan, Real-Time P2P Traffic Identification, in proceedings of the *49th annual IEEE Global Telecommunications Conference*, New Orleans, LA, 2008, pp. 1-5.
[28] J. Ma, K. Levchenko, C. Krebich, S. Savage, and G. Voelker, Unexpected Means of Protocol Inference, in proceedings of the *6th ACM SIGCOMM Conference on Internet Measurement*, Rio de Janeiro, Brazil, 2006, pp. 313-326.
[29] B. Cohen. The BitTorrent Protocol Specification. *http://www.bittorrent. org/beps/bep_0003.html*
[30] MSE: Message stream encryption protocol. *http://www.azureuswiki.com/index.php/MessageStreamEncryption*.
[31] Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Trans. Information Theory IT*, 1976, 22(6), pp. 644-654
[32] Thayer, R., Kaukonen, K. A stream cipher encryption algorithm 'arcfour'. *http://www.mozilla.org/projects/security/pki/nss/draft-kaukonen-cipher-arcfour-03.txt.*
[33] Peer Exchange. *http://wiki.vuze.com/w/Peer_Exchange.*
[34] V. Paxson, Bro: a system for detecting network intruders in real-time, *Computer Networks*, 1999, 31(23), pp. 2435-2463.
[35] Z. Yang, L. Z. Li, Q. J. Ji, Y. Q. Zhu. An omnibus identification of BitTorrent traffic in a stub network, in proceedings of the *3rd International Symposium on Parallel Architectures, Algorithms and Programming*, Dalian, China, 2010, pp. 346-353.

Zhe Yang was born in Suzhou, Jiangsu province, in 1978. He received the B.S. degree in computer application from Donghua University, Shanghai, in 2000. He received the M.S. and Ph.D degree in computer application technology form Tongji University, Shanghai, in 2003 and 2006.

He is working in Soochow University. And his research interests include traffic classification, P2P behavior analysis and network measurement.

Ling-Zhi Li was born in Dezhou, Shandong province, in 1977. He received the B.S. and M.S. degrees in information technology from East China Institute of Technology, Fuzhou, in 1998 and 2001. He received the Ph.D. degree in computer application from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2006.

He is working in Soochow University. His research interests include traffic engineering, network security.

Qijin Ji was born in Anhui province China, 1974. He received the M.S degree in information engineering from Nanjing University of Posts and Telecommunications, Nanjing, China and Ph.D degree in computer science and engineering from Southeast University, Nanjing, China, in 2002 and 2005 respectively.

After a two year experience as a Post-doctor fellow in Shanghai Jiaotong University, Shanghai China, he joined the School of Computer Science and Engineering of Soochow University and currently he is an assistant professor. His research focuses on performance evaluation and resource control of computer networks, distributed estimation and decision making in multi-agent systems and distributed multimedia streaming systems.

Yanqin Zhu was born in Wuxi, Jiangsu province, in 1964. She received the B.S. degree in Mathematics from Soochow University in 1985，the M.S. degree in computer application technology form Southeast University in 1991, and Ph.D degree in computer application technology form Soochow University in 2008.

She has been working in Soochow University since 1985. And her research interests include computer network and applied cryptography.