Finding Peer-To-Peer File-sharing Using Coarse Network Behaviors*

Michael P. Collins[1] and Michael K. Reiter[2]

[1] CERT/Network Situational Awareness, Software Engineering Institute, Carnegie Mellon University mcollins@cert.org
[2] Electrical & Computer Engineering Department, Computer Science Department, and CyLab, Carnegie Mellon University reiter@cmu.edu

Abstract. A user who wants to use a service forbidden by their site's usage policy can masquerade their packets in order to evade detection. One masquerade technique sends prohibited traffic on TCP ports commonly used by permitted services, such as port 80. Users who hide their traffic in this way pose a special challenge, since filtering by port number risks interfering with legitimate services using the same port. We propose a set of tests for identifying masqueraded peer-to-peer file-sharing based on traffic summaries (flows). Our approach is based on the hypothesis that these applications have observable behavior that can be differentiated without relying on deep packet examination. We develop tests for these behaviors that, when combined, provide an accurate method for identifying these masqueraded services without relying on payload or port number. We test this approach by demonstrating that our integrated detection mechanism can identify BitTorrent with a 72% true positive rate and virtually no observed false positives in control services (FTP-Data, HTTP, SMTP).

1 Introduction

Peer-to-peer file-sharing services are often constrained by organizations due to their widespread use for disseminating copyrighted content illegally, their significant bandwidth consumption for (typically) non-work-related uses, and/or the risk that they may introduce new security vulnerabilities to the organization. Karagiannis et al. [5] have shown that instead of obeying site bans on file-sharing, however, users hide their file-sharing activity. Moreover, file-sharing tools themselves are being updated to circumvent attempts to filter these services; e.g., BitTorrent developers now incorporate encryption into their products in order to evade traffic shaping.[3]

While encryption makes filtering based on traffic content difficult, filtering packets by port number (as would typically be implemented in router ACLs,

* This work was partially supported by NSF award CNS-0433540, and by KISA and MIC of Korea.

[3] "Encrypting BitTorrent to Take Out Traffic Shapers", TorrentFreak Weblog, http://torrentfreak.com/encrypting-BitTorrent-to-take-out-traffic-shapers/

for example) remains an obstacle to peer-to-peer file-sharing. As such, common hiding methods also involve changing the port number used by the service to something that is not filtered. In networks that implement a "deny-than-allow" policy, the service traffic may be sent on a common service port, in particular 80/TCP (HTTP).

In such cases, ports do not reliably convey the services using them, while deep packet examination is viable only as long as packet payload is unencrypted. Analysts therefore need alternative methods to characterize and filter traffic. In this paper, we propose an alternative service detection and identification method that characterizes services behaviorally. We hypothesize that TCP services have quantifiable behaviors that can be used to identify them without relying on payload or port numbers. For example, we expect that the majority of HTTP sessions begin with a small initial request followed by a larger response, and then terminate. If a presumed HTTP client and HTTP server were communicating using symmetric short bursts of traffic in a single long-lived session, then we would have reason to consider an alternative hypothesis, such as a chat service.

Within this paper, we focus on a specific problem that motivated this line of research: demonstrating that a user who claims to be using a common service on its standard port (such as HTTP) is using another service, specifically BitTorrent. To do so, we implement tests that characterize traffic and show how they can be used together to effectively differentiate BitTorrent traffic from common services. The goal of our research is a collection of tests which can be used by analysts or automated systems to classify traffic. Given the increasing sophistication of evasion strategies, we seek to find behaviors that can be effective with as few assumptions as possible. For example, these tests do not use deep packet examination, and are therefore applicable to encrypted and unencrypted traffic.

We calibrate and validate our approach using logs of traffic crossing a large network. From these logs, we select traffic records describing BitTorrent and major services, specifically HTTP, FTP data channel and SMTP. The log data consists of NetFlow, a traffic summarization standard developed by CISCO systems[4]. Flow data is a compact representation of an approximate TCP session, but does not contain payload. In addition, we do not trust port numbers, making our tests port- and payload-agnostic. Despite this, we show that by classifying flows based on several behaviors we can effectively differentiate source-destination pairs engaged in BitTorrent communication from those involved in HTTP, FTP or SMTP exchanges. Specifically, our integrated test identifies BitTorrent with a 72% true positive rate and virtually no observed false positives in control services (FTP-Data, HTTP, SMTP).

The rest of this paper is structured as follows. Section 2 describes previous work done in service detection. Section 3 describes the behaviors with which we characterize flows, and that we use to distinguish certain file-sharing traffic from other types of traffic. Section 4 describes our experiments using these classifi-

[4] CISCO Corporation, *Netflow Services and Applications*, `http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/nappswp.htm`

cations both individually and in aggregate, to identify BitTorrent activity. We conclude in Section 5.

2 Previous Work

Prior work in identifying file-sharing traffic varies primarily in the information used to do so. Several file-sharing detection tools have analyzed packet payload (e.g., [9, 20]), a method which will not survive encryption of packet contents or might simply be infeasible due to performance or other limitations. Other approaches utilize aggregate packet attributes, such as interstitial arrival times or the presence of specific packet sequences (e.g., [22, 8, 2, 24, 3, 10–12]). However, in sufficiently large and busy networks, even this degree of packet analysis can be problematic.

As a result, flows are increasingly used for various types of security analysis (e.g., [18, 13]). Flows were originally specified by Partridge [15] for traffic summarization, and have since been adopted by CISCO for traffic reporting. NetFlow, the CISCO standard, uses timeouts to approximate TCP sessions, an approach originally developed by Claffy et al. [1]. Since flow records do not contain payload information, they are generally used for large-scale and statistical analysis. Notably, Soule et al. [21] developed a classification method to cluster flows, though they stopped short of mapping them to existing applications (or types of applications).

Since their development, peer-to-peer file-sharing systems have become targets of filtering and detection efforts. Karagiannis et al. [5] showed that peer-to-peer users increasingly evade detection by moving their traffic to alternate port numbers. Studies conducted on BitTorrent and other peer-to-peer file-sharing applications have examined the behavior of individual nodes (e.g., [4, 23, 7]) and application networks (e.g., [19, 17]), but have not compared the behaviors observed to the behavior of more traditional services. Ohzahata et al. [14] developed a method for detecting hosts participating in the Winny file-sharing application by inserting monitoring hosts within the file-sharing network itself. Karagiannis et al. [6] developed a general method for identifying applications called Blinc, which uses various heuristics and interconnection patterns exhibited by groups of nodes to identify services. In contrast, we focus on the flow characteristics between a pair of nodes in isolation to identify the service in which they are participating, and as such our approach is complementary. Nevertheless, we believe there is potential in combining our point-to-point analyses with Blinc's on interconnection patterns, and hope to investigate this in future work.

3 Application Classification

In this section, we describe the behaviors used to differentiate BitTorrent traffic from other services. In Section 3.1 we describe a classification tree that we will use to classify flows into different types, and in Section 3.2 we describe the intuition and formulation of our tests.

3.1 Simple Taxonomy

Our analyses use flow records; a *flow* is a sequence of packets with the same addressing information (source and destination addresses, source and destination ports, and protocol) which occur within a short time of each other [1]. A *flow record* is a summary consisting of addressing, size and timing information about the flow, but no payload. We will refer to fields of a flow record f with "dot" notation (e.g., f.duration or f.bytes).

We restrict our data to TCP flows. Flow collection systems such as CISCO NetFlow record TCP flags by ORing the flags of every packet. As a result, flag distributions cannot be derived from multi-packet flow records, and certain behaviors—notably whether an endpoint is the initiator or responder of the TCP connection of which the flow represents one direction—are not discernible.

We divide flows into three categories: **Short Flows**, comprising three packets or less; **Messages**, which are 4–10 packets but less than 2 kB in size; and **File Transfers**, which are any flows longer than a Message. Figure 1 represents our taxonomy as a decision tree and the categories that this tree covers.

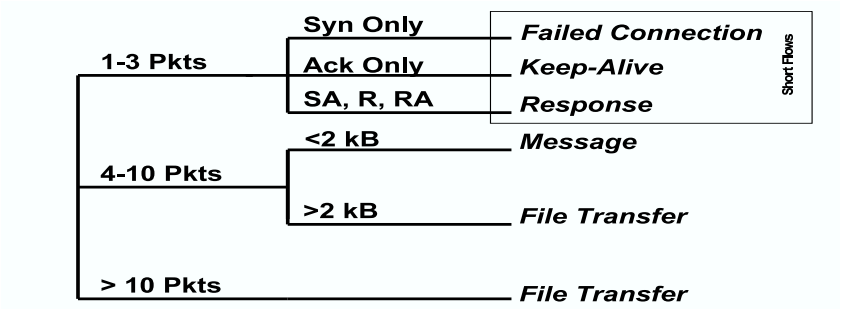


Fig. 1. Flow Classification Tree: The rules used on this tree assign each flow to a class

A **Short Flow** consists of three or fewer packets; since a complete TCP session will require at least three packets, Short Flows indicate some error in communication or anomaly in flow collection. Within Short Flows, we can acquire more information by examining the TCP flags of the flow; we use the flags to create three sub categories. A **Failed Connection** has a SYN flag and no ACKs. A **Keep-Alive** has ACK flags only. Since flow recording is timeout-based, Keep-Alives are recorded by the flow collector during long-lived sessions but currently do not have any substantial impact on analysis. A **Response** consists of any Short Flow whose flags imply a response from the TCP connection responder to the initiator: a SYN-ACK, a RST-ACK or a RST. We do not consider the other TCP flags (e.g., PSH) significant in this analysis. As noted above, TCP flags are OR'ed in flow records, and as a result we only use flags in the Short Flow case, where the results are least ambiguous.

We define a **Message** as a flow consisting of 4–10 packets and with a total size less than 2 kB. We assume that Messages represent the structured exchange of service data between the source and destination. Example Messages include HTTP requests and the control messages sent by BitTorrent. We assume that Messages contain structured communication, as opposed to data intended for the application's users. Consequently, we expect that Messages will have fixed sizes and that certain Messages (with specific sizes) will appear more often than other ones.

We label any flow longer than 2 kB or 10 packets a **File Transfer**. We assume that a File Transfer is the exchange of non-service-specific information between two sites. We expect that certain services will tend to send shorter files than others. For example, we expect that HTTP servers will transfer less data than BitTorrent peers typically, since HTTP clients interact with users and therefore need a rapid response time.

Table 1 is a log showing BitTorrent flows; in this log, we have labeled each flow with its corresponding category. Of particular interest is the presence of repeated Failed Connections (the 144-byte SYN-only packets) and the 276-byte Message packets. Both of these behaviors will be used to construct tests in Section 3.2.

Source Port	Destination Port	Packets	Bytes	Flags				Start Time	Class
				F	S	A	R		
3584	6881	1637	1270926		x	x		11/04/2005 21:09:33	File Transfer
3586	6881	5	276	x	x	x		11/04/2005 21:09:36	Message
3619	6881	5	276	x	x	x		11/04/2005 21:10:18	Message
3651	6881	5	276	x	x	x		11/04/2005 21:11:01	Message
3701	6881	5	276	x	x	x		11/04/2005 21:12:04	Message
1290	6881	3	144		x			11/04/2005 21:53:56	Failed Connection
2856	6881	5	636		x	x		11/04/2005 22:33:11	Message
3916	6881	5	276	x	x	x		11/04/2005 23:03:44	Message
4178	6881	5	636		x	x		11/04/2005 23:12:01	Message
4884	6881	3	144		x			11/04/2005 23:32:05	Failed Connection

Table 1. Log of traffic and associated classification

3.2 Tests

In this section, we describe four tests for characterizing the flows generated by various services. Each test is performed on a log of flow records bearing the same source and destination, and hence are unidirectional. We rely on unidirectional flows for three reasons. First, CISCO NetFlow is reported unidirectionally; i.e., each direction of a connection is reported in a different flow. Second, on a network with multiple access points, there is no guarantee that entry and exit traffic

passes through the same interface. As a result, determining the flows representing both directions of a connection on a large network can be prohibitively difficult. Finally, we wish to acquire results using as little information as possible.

Each test outlined in this section is characterized by function $\theta(x, F)$. This binary-valued function applies a particular threshold x to a measure calculated on a *flow log* F, where F consists of flows all bearing the same source and destination.

Failed Connections We expect that, except for scanners, clients (respectively, peers) open connections only to servers (respectively, other peers) of which they have been informed by another party. For example, BitTorrent peers connect to peers that they have been informed of by their trackers. We further expect that when a client (peer) attempts to connect to a server (peer) that is not present, we will see multiple connection attempts.

We thus expect that Failed Connections occur more frequently in traffic from services with transient server/peer populations. We expect that BitTorrent peers regularly disconnect and that other peers will try to communicate with them after this. In contrast, we expect that the providers of HTTP and SMTP servers will implement policies to ensure maximum uptime. Therefore, under normal circumstances, we expect that the rate of Failed Connections for BitTorrent will be higher than it would be for SMTP or HTTP, for example.

Table 2 summarizes Failed Connections in an hour of observed data from four services: HTTP, FTP, SMTP and BitTorrent. From the observed flow records, we count the number of unique source-destination pairs and then count the number of pairs that have multiple Failed Connections. As the table shows, only SMTP and BitTorrent have a significant rate of Failed Connections, and BitTorrent has a somewhat higher rate of Failed Connections than SMTP does.

Service	Source-destination pairs	Pairs experiencing n Failed Connections			
		$n = 0$	$n = 1$	$n = 2$	$n > 2$
HTTP	3089	2956 (96%)	78 (3%)	22 (1%)	33 (1%)
FTP	431	431 (100%)	0 (0%)	0 (0%)	0 (0%)
SMTP	18829	15352 (82%)	1789 (10%)	619 (3%)	1069 (6%)
BitTorrent	49	37 (76%)	1 (2%)	0 (0%)	11 (22%)

Table 2. Failed Connections per service for sample data

To evaluate connection failure as a detection method, we will use the following threshold test, $\theta_c(x, F)$:

$$\theta_c(x, F) = \begin{cases} 0 & \text{if the percentage of Failed Connections in } F \text{ is less than } x \\ 1 & \text{otherwise} \end{cases}$$

Bandwidth In most client-server applications, a single server communicates with multiple clients; therefore, implementing faster transfers generally requires purchasing a higher-bandwidth connection for the server. BitTorrent increases the speed of a transfer by adding more peers to the BitTorrent "swarm": each peer transfers a fragment of the desired file, resulting in a high-bandwidth transfer comprised of multiple low-bandwidth connections. We expect that most dedicated servers will transfer data at higher speeds than a peer in a BitTorrent transfer will.

For flow logs of file transfer services (as opposed to a chat service, such as AIM), we of course expect that many of the corresponding connections are used for file transfers. The flow in the direction opposite the file transfer will consist almost exclusively of 40-byte zero-payload packets. Calculating bandwidth by counting the bytes in these packets will result in an artificially depressed value for the bandwidth consumed by the connection that gave rise to this flow. To compensate for this, we assume that the files transferred are considerably larger than the standard 1500-byte MTU for Ethernet, and consequently fragmented into MTU-length packets. We then estimate bandwidth for such flows by assuming that all of the ACK packets are in response to 1500-byte packets, resulting in the following formula:

$$b(f) = \frac{1500 \text{ bytes/packet} * f.\mathsf{packets}}{\max(f.\mathsf{duration}, 1 \text{ second})}$$

For bandwidth tests, our threshold function, θ_b, will be expressed as follows:

$$\theta_b(x, F) = \begin{cases} \perp & \text{if there are no File Transfers in } F \\ 0 & \text{if there is a File Transfer } f \in F \text{ such that } b(f) \geq x \\ 1 & \text{otherwise} \end{cases}$$

Note that $\theta_b(x, F)$ is undefined if there are no File Transfers in F. A bandwidth test cannot be meaningfully applied to a flow log with no File Transfers, since flow durations are recorded with second precision (while Messages and Short Flows typically take far less than a second).

Comparing Message Profiles We expect that Messages will have fixed formats specified by the service. For example, the first action taken by a BitTorrent peer upon connecting to another peer is to send a 68-byte handshake containing a characteristic hash for its target file.[5] We therefore expect that a disproportionate number of BitTorrent Messages will have 68-byte payloads.

Comparative histograms for BitTorrent and HTTP Message sizes are shown in Figure 2. As this example shows, BitTorrent traffic spikes at 76 bytes per Message. This is most likely due to the 68 bytes necessary to send the BitTorrent handshake, plus a common 8-byte set of TCP options [16].

[5] See `http://wiki.theory.org/BitTorrentSpecification`.

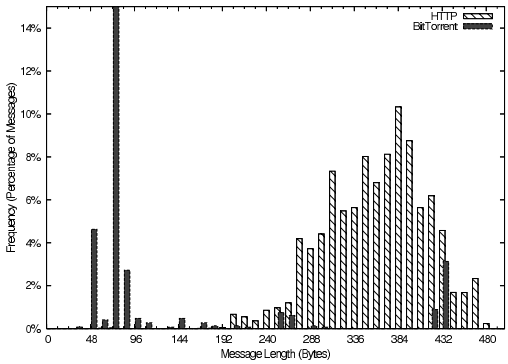


Fig. 2. Histogram comparing BitTorrent and HTTP Messages; the peak comprises 84% of BitTorrent Messages

We test this hypothesis by generating histograms for the Messages from each major data set and then use the L_1 distance as an indicator of similarity. We define a histogram H on the Messages in a flow log F as a set of values $h_0(F) \ldots h_n(F)$, such that $h_i(F)$ is the observed probability that a Message in F has a payload between $[k(i-1), ki)$ bytes in size. In particular, $\sum_{i=1}^{n} h_i(F) = 1$. Given a reference flow log F_R and a test flow log F_T, the L_1 distance is defined as:

$$L_1(F_T, F_R) = \sum_{i=0}^{n} |h_i(F_T) - h_i(F_R)|$$

Histogram calculation is complicated by TCP headers and TCP options. TCP packets contain 40 bytes of header data above any payload. In addition, TCP sessions specify TCP payload options in the first packet; these options will be some multiple of 4 bytes and also affect the payload size. To compensate for the headers, we calculate the payload size for a flow f as:

$$p(f) = f.\mathsf{bytes} - (40 \text{ bytes/packet} \times f.\mathsf{packets})$$

We set k, the size of the bins used to calculate probabilities, to 12 bytes. This bin size is larger than the sizes of the most common TCP option combinations [16] and should therefore reduce profile mismatches caused by different option combinations.

The threshold test is then:

$$\theta_p(x, F) = \begin{cases} \perp & \text{if there are no Messages in } F \\ 0 & \text{if there exist Messages in } F \text{ and } L_1(F, F_R) \geq x \\ 1 & \text{otherwise} \end{cases}$$

It is understood that the L_1 distance is calculated only using the Messages in F and F_R. Note that $\theta_p(x, F)$ is undefined if there are no Messages in F: Short Flows should not have payload data, and we assume that File Transfers contain unstructured (and so arbitrary-length) data.

Volume Because HTTP files are immediately viewed by a user, we expect that web pages are relatively small in order to ensure rapid transfer. Therefore, we expect that if a BitTorrent user transfers a file, the resulting TCP session will transfer more bytes than a HTTP session would.

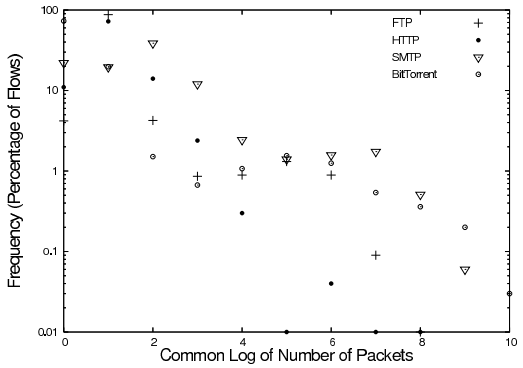


Fig. 3. Distribution of File Transfer volume for major sources

Figure 3 is a log plot of the frequency of flows; each flow is binned by the common logarithm of its packet count. The results in Figure 3 satisfy our intuition about file sizes: the majority of HTTP flows are relatively low-volume, with the majority consisting of 20 packets or less. We note that, in contrast to our original expectations, SMTP induces higher-volume flows than FTP.

For this behavior, the threshold function $\theta_t(x, F)$, is defined as:

$$\theta_t(x, F) = \begin{cases} 0 & \text{if } \forall f \in F : f.\mathsf{packets} \leq x \\ 1 & \text{otherwise} \end{cases}$$

Similar to the bandwidth test, the presence of even a single large-volume flow is an indicator.

4 Experiments

This section summarizes the tests conducted on the source data and their results. Section 4.1 describes the source data, and Section 4.2 describes how we calibrate the individual tests. Section 4.3 describes the integration and validation of these

tests into a combined test for BitTorrent, and evaluates its accuracy. Section 4.4 discusses what countermeasures a user can take to evade the tests.

4.1 Calibration Data

Calibration data is selected from a log of traffic at the border of a large edge network. Since the flow logs do not contain payload information, we cannot determine what service any flow describes via signatures. In order to acquire calibration and validation data, we use a combination of approaches which are described below.

Each data set consists of FTP, SMTP, HTTP or BitTorrent flow records. SMTP and HTTP were chosen because they comprise the majority of bytes crossing the monitored network. The FTP data set is actually FTP-Data traffic (i.e., port 20). We chose to use FTP-Data because we expected that it would consist primarily of large File Transfers and thus be difficult to distinguish from BitTorrent.

In the case of FTP, SMTP and HTTP, server names are used to validate services. HTTP server names usually begin with a "www" prefix; similarly, SMTP server names often begin with "mx" or "mail". We restrict our flows to those with destination addresses to which a name containing a relevant string resolves. We assume that if a destination has an appropriate name and participates in the flow on the relevant port, then it is providing the suggested service.

In each case, the flows are outgoing flows collected from the observed network, i.e., flow sources are always inside the network and flow destinations are always outside. The calibration data are detailed below:

- HTTP: The HTTP data set consists of flow records where the destination port is port 80 and the source port is in the range 1024–5000, the ephemeral port range for Windows. In addition, every destination address must be that of an HTTP service (such as Akamai or Doubleclick) or have a name including "www" or "web".
- SMTP: The SMTP data set consists of flow records where the destination port is 25 and the source port is ephemeral. Destination addresses must have names containing the string "smtp", "mail" or "mx".
- FTP: The FTP data set consists of FTP-Data (port 20) flows; the corresponding FTP-Control (port 21) information is not used in this analysis. The FTP data set consists of flow records where the destination port is 20 and the source port is in the ephemeral range. Destination addresses must have names that begin with the string "ftp".
- BitTorrent: The BitTorrent data set consists of logs from hosts running BitTorrent in the observed network; these hosts were identified using a banner grabbing system configured to identify BitTorrent prefixes. While this system can identify hosts running BitTorrent, it does not identify when BitTorrent communication occurred. Therefore, when examining traffic logs to and from a suspected host we assume that other applications do not masquerade as BitTorrent, and so any interaction between two IP addresses (one of which

is suspected) where the source port is ephemeral and the destination port is 6881 is assumed to be BitTorrent.

All source data comes from the first week of November 2005; due to the frequency with which particular services are used, each data set comprises a different period of time. In particular, BitTorrent traffic was collected over 2 days, while the other services were collected over several hours.

Flow Classification		Service			
		HTTP	FTP	SMTP	BitTorrent
All	flows	205702 (100%)	120144 (100%)	142643 (100%)	13617 (100%)
	src-dst pairs	3088 (100%)	746 (100%)	18829 (100%)	2275 (100%)
Failed Connection	flows	2427 (1%)	0 (0%)	13010 (9%)	9325 (68%)
	src-dst pairs	132 (4%)	0 (0%)	3476 (18%)	1992 (88%)
Keep-Alive	flows	5403 (3%)	1965 (2%)	31306 (22%)	1106 (8%)
	src-dst pairs	409 (13%)	504 (68%)	2566 (14%)	135 (6%)
Response	flows	18635 (9%)	23663 (20%)	12545 (9%)	199 (1%)
	src-dst pairs	730 (24%)	314 (42%)	2959 (16%)	53 (2%)
Message	flows	150937 (73%)	64558 (54%)	26271 (18%)	1615 (12%)
	src-dst pairs	2880 (93%)	558 (75%)	4704 (25%)	270 (12%)
File Transfer	flows	28300 (14%)	29958 (25%)	59511 (42%)	1372 (10%)
	src-dst pairs	1504 (49%)	504 (68%)	13771 (73%)	199 (9%)

Table 3. Classification of flow records in calibration data

Table 3 summarizes the flows in the calibration data sets. Consider any individual "Service" column, e.g., the column "HTTP" that indicates the HTTP data set. For each flow classification, the row marked "flows" shows the number of flows of that type that appeared in that data set, and the approximate percentage of the total number of flows that these comprised. Ignoring the "All" classification, the rest of the classifications are mutually exclusive and exhaustive, and so the flows accounted for in these "flows" rows total to the number of flows in the data set (indicated under the "All" classification). Similarly, for each flow classification, the number of source-destination pairs for which at least one flow of that type appeared in the data set is shown in the "src-dst pairs" row. Since the same source-destination pair can be listed for multiple flow classifications—if flows of each of those classifications were observed between that source-destination pair—the sum of the source-destination pair counts for the various flow types (excluding "All") generally exceeds the number of source-destination pairs in the data set. Note that FTP does not show any failed connections, presumably because FTP-Data connections are opened after FTP-Control connections have succeeded.

4.2 Calibration

We have hypothesized that BitTorrent can be distinguished from other services by examining bandwidth, volume, the profile of Messages and the rate of Failed Connections. To evaluate this hypothesis, we now construct a series of tests, one for each behavior. In each test, we specify the behavior as a parameterized function and plot the results on a ROC (Receiver Operating Characteristic) curve. The ROC curve is generated by using the appropriate θ-test against flow logs selected from the data sets in Table 4. In each case, a control data set (either HTTP, SMTP or FTP) is partitioned into flow logs $C_1 \ldots C_\ell$, where ℓ is the total number of distinct source-destination pairs in the set and each C_i is log of all flows between one source-destination pair. Similarly, the BitTorrent data set is partitioned into flow logs $T_1 \ldots T_m$, each between one source-destination pair.

The contents of C_i and T_j are then evaluated by the corresponding θ-function. The false and true positive rates are calculated as follows, given a test function $\theta(x, F)$.

$$FP(x) = \frac{\sum_{C \in \mathcal{C}} \theta(x, C)}{|\mathcal{C}|} \quad \text{where } \mathcal{C} = \{C_i : \theta(x, C_i) \neq \perp\}$$

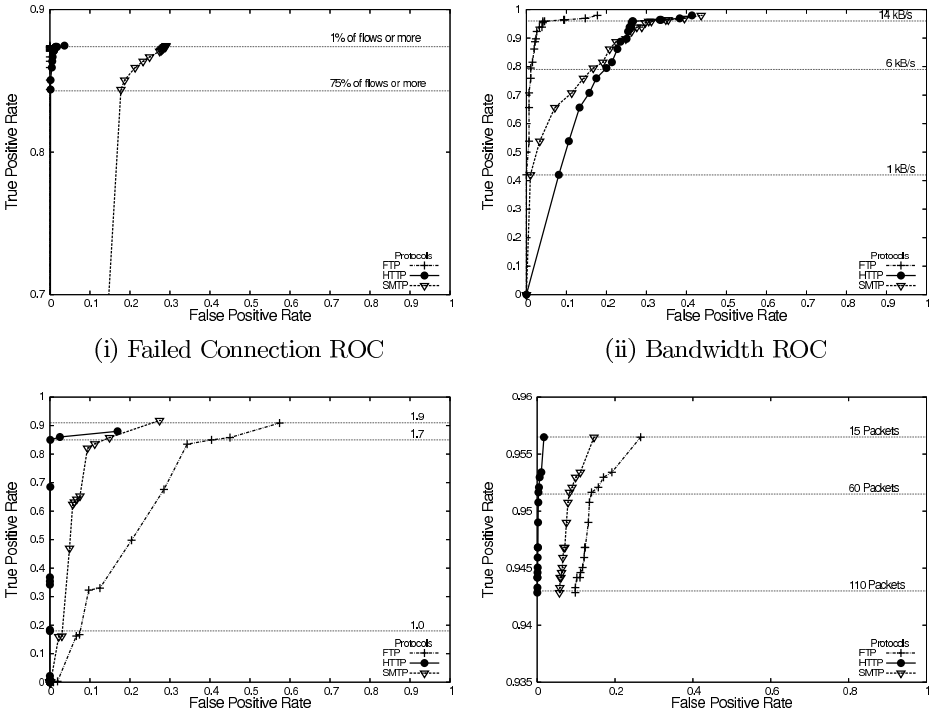
$$TP(x) = \frac{\sum_{T \in \mathcal{T}} \theta(x, T)}{|\mathcal{T}|} \quad \text{where } \mathcal{T} = \{T_j : \theta(x, T_j) \neq \perp\}$$

Figure 4(i) shows how effective Failed Connections are for detecting BitTorrent versus control data. As described in Section 3.2, BitTorrent has an usually high number of Failed Connections: for 85% of the observed source-destination pairs, at least 75% of their flow logs consist of Failed Connections. SMTP traffic is the least distinguishable from BitTorrent; we believe that this is because both BitTorrent and SMTP connections are opened as the application requires, while HTTP and FTP connections are opened by user request. If an HTTP site is not available, a user is likely to give up; conversely, an SMTP client will retry quickly.

While the Failed Connection test is highly distinctive on these data sets, we note that our test data may be biased by the varying times used to collect data. While, as Table 1 shows, BitTorrent connections can fail over very short periods of time, a better statistical model may produce more precise results over homogeneous time periods.

Figure 4(ii) summarizes the results of our bandwidth tests. The operating characteristic is the maximum observed bandwidth, estimated between 1 kB/s and 20 kB/s using 1 kB/s increments. As this curve shows, the bandwidth cutoff for BitTorrent is approximately 14 kB/s, which allows a 90% correct identification for both HTTP and FTP traffic. SMTP traffic is less distinguishable, with higher than a 35% false positive rate at the same value.

We note that the bandwidth estimates are low; e.g., according to Figure 4(ii), roughly 30% of source-destination pairs for HTTP and SMTP had bandwidths less than 14 kB/s. We conjecture that this is primarily a result of limitations



(i) Failed Connection ROC

(ii) Bandwidth ROC

(iii) Message profiling ROC

(iv) Volume ROC

Fig. 4. ROC curves generated from tests

in using flow records to estimate bandwidth. For example, the available bandwidth of a persistent HTTP connection is not fully utilized for the duration of the connection; instead, content transfers on the connection are separated by intervening idle periods. Consequently, the flows recorded for such connections will have a deflated average bandwidth. Such effects could be rectified by using alternative flow-like metrics; several are proposed by Moore et al. [12], but would require modifying existing flow formats.

Figure 4(iii) summarizes the success rate for differentiating BitTorrent and other traffic using a profile of Messages. Each point in this graph is a mean of the results from 10 runs. In each such run, 20% of the source-destination pairs in the BitTorrent data set are selected uniformly at random and used to generate the profile. During the test, the remaining 80% are used to generate profiles for comparison. As this curve shows, Message profile comparison worked well for differentiating HTTP and SMTP traffic, but was considerably less accurate when comparing FTP traffic. We note that due to the random selection used during these tests, the resulting true positive rates vary for each data set; the thresholds shown in Figure 4(iii) are approximate across the three data sets.

Figure 4(iv) shows the efficacy of the volume test. In this curve, the operating characteristic ranges between 15 packets and 110 packets. We note that the true positive rate for the curve is very high, and that 94% of the source-destination flow logs observed for BitTorrent had one transfer of at least 110 packets, a number far higher than anything sent by our example HTTP flow logs.

4.3 Integration and Validation

We now combine the tests in order to identify BitTorrent. To do so, we use voting. For every source-destination pair in a data set consisting of mixed BitTorrent and control traffic, each test is applied to the corresponding flow log in order to determine whether the traffic is characteristic of BitTorrent. The result of the test is recorded as a vote, where an undefined result is a 0-vote. For each source-destination pair, the total number of votes is then used as the operating characteristic on a ROC curve. For this curve, a false positive is a flow log of control traffic that is identified as BitTorrent, and a false negative is a flow log of BitTorrent traffic identified as control traffic. This voting method is intended to be a simple integration attempt; in future work we may opt for more sophisticated methods.

The data sets used to validate these tests are summarized Table 4. These data sets are taken from different sampling periods than the calibration data sets and do not overlap them. Otherwise they are assembled using the same approach as the calibration data.

False positives and false negatives are weighed equally for each test: The upper left corner of a ROC curve represents perfect classification (i.e., no false positives and perfect true positives), and so we choose the point from each calibration curve that minimizes the distance from that corner. For the tests in Section 4.2, this results in a threshold x_c^{opt} for θ_c of 1%; for x_b^{opt} of 14 kB/s; for x_p^{opt} of 1.7; and for x_t^{opt} of 110 packets. Table 4 shows the results of each individual test applied against the validation data sets; we note that unlike calibration, the θ_p and θ_b tests are applied against all flow logs. As Table 4 shows, each behavior was demonstrated by at least 50% of the BitTorrent source-destination pairs. In comparison, none of the control services had a test which more than 30% of that service's flows passed.

Data Set	Flows	Src-dst pairs	Pairs for which $\theta(x^{\text{opt}}, F) = 1$			
			θ_c	θ_b	θ_p	θ_t
BitTorrent	9911	70	36 (51%)	51 (73%)	63 (90%)	51 (73%)
HTTP	42391	1205	27 (2%)	156 (13%)	5 (0%)	64 (5%)
SMTP	46146	4832	729 (15%)	1350 (28%)	105 (2%)	1451 (30%)
FTP	47332	561	0 (0%)	15 (3%)	164 (29%)	158 (28%)

Table 4. Validation data sets and individual test results

The integrated vote is plotted as a ROC curve in Figure 5. The integrated test results in a 72% true positive rate if 3 or more tests agree, and a corresponding false positive rate of 0% for all control services. None of the control source-destination pairs received four affirmative votes, and only source-destination pairs in the SMTP data set received three. HTTP was the most easily distinguished of the control services, with less than 1% of observed source-destination pairs having two affirmative votes.

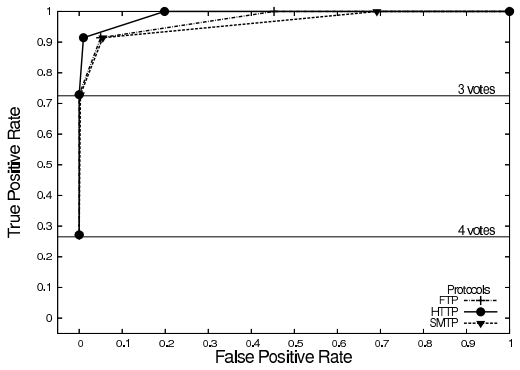


Fig. 5. Integrated ROC curve identifying BitTorrent.

4.4 Evasion

As noted in Section 1, users now actively evade detection methods. As a result, we must consider how the developers of a service can evade the tests discussed in this paper.

The BitTorrent specification can be changed to evade Message profiling by changing the sizes of the control Messages. If the Messages were randomly padded, a size-based profiling method would be less reliable. This approach is easy to implement.

The volume test can be evaded by limiting the maximum amount of data received from a single host. However, this approach means that BitTorrent peers will have to communicate with a larger number of peers, at which point they are increasingly vulnerable to host-counting strategies such as Blinc's [6].

Tests based on Failed Connections can be evaded by increasing the time between connection attempts or by ensuring that a BitTorrent peer only communicates with peers that are known to be active, a function that is theoretically already supported by the tracker. However, either change will increase download time by increasing the time required to find an active host.

Evading bandwidth detection is more challenging: users must either purchase additional bandwidth or use all their bandwidth to transfer data.

5 Conclusions

In this paper, we have demonstrated that services have well-defined behaviors which can be used to identify masqueraded peer-to-peer file-sharing traffic without relying on payload or port numbers. To do so, we have developed a collection of tests based on service behavior and shown how BitTorrent differs from SMTP, HTTP and FTP traffic. These results are used to demonstrate that BitTorrent is characteristically different from these other services, and that these differences can be detected by looking at gross flow-level attributes, without relying on deeper packet examination than what is normally required to construct the flow.

References

1. K. Claffy, H. Braun, and G. Polyzos. A parameterizable methodology for internet traffic flow profiling. *IEEE Journal of Selected Areas in Communications*, 13(8):1481–1494, 1995.
2. J. Early, C. Brodley, and C. Rosenberg. Behavioral authentication of server flows. In *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003.
3. F. Hernandez-Campos, A. Nobel, F. Smith, and K. Jeffay. Understanding patterns of TCP connection usage with statistical clustering. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2005.
4. M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *Proceedings of the 5th Annual Passive and Active Measurement Workshop*, 2004.
5. T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is p2p dying or just hiding? In *Proceedings of IEEE Globecom 2004 - Global Internet and Next Generation Networks*, 2004.
6. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005.
7. P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, and E. Zegura. Bootstrapping in gnutella: A measurement study. In *Proceedings of the 5th Annual Passive and Active Measurement Workshop*, 2004.
8. M. Kim, H. Kang, and J. Hong. Towards peer-to-peer traffic analysis using flows. In *Self-Managing Distributed Systems, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, 2003.
9. C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 2002.
10. A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement*, 2004.
11. A. De Montigny-Leboeuf. Flow attributes for use in traffic characterization. Technical Report CRC-TN-2005-003, Communications Research Centre Canada, December 2005.

12. A. Moore, D. Zuev, and M. Crogan. Discriminators for use in flow-based classification. Technical Report RR-05-13, Department of Computer Science, Queen Mary, University of London, August 2005.
13. W. Nickless, J. Navarro, and L. Winkler. Combining CISCO netflow exports with relational database technology for usage statistics, intrusion detection, and network forensics. In *Proceedings of the 14th Annual Large Systems Administration (LISA) Conference*, 2000.
14. S. Ohzahata, Y. Hagiwara, M. Terada, and K. Kawashima. A traffic identification method and evaluations for a pure p2p application. In *Proceedings of the 6th Annual Passive and Active Measurement Workshop*, 2005.
15. C. Partridge. A Proposed Flow Specification. RFC 1363 (Informational), September 1992.
16. K. Pentikousis and H. Badr. Quantifying the deployment of TCP options, a comparative study. *IEEE Communications Letters*, 8(10):647–649, October 2004.
17. J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. A measurement study of the BitTorrent peer-to-peer file-sharing system. Technical Report PDS-2004-007, Delft University of Technology, April 2004.
18. S. Romig, M. Fullmer, and R. Luman. The OSU flow-tools package and CISCO netflow logs. In *Proceedings of the 14th Annual Large Systems Administration (LISA) Conference*, 2000.
19. S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking*, 2002.
20. S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th International Conference on World Wide Web*, 2004.
21. A. Soule, K. Salamatia, N. Taft, R. Emilion, and K. Papagiannaki. Flow classification by histograms: or how to go on safari in the internet. In *Proceedings of the 2004 Joint International Conference on Measurement and Modeling of Computer Systems*, 2004.
22. C. Taylor and J. Alves-Foss. An empirical analysis of NATE: network analysis of anomalous traffic events. In *Proceedings of the 10th New Security Paradigms Workshop*, 2002.
23. K. Tutschku. A measurement-based traffic profile of the eDonkey filesharing service. In *Proceedings of the 5th Annual Passive and Active Measurement Workshop*, 2004.
24. C. Wright, F. Monrose, and G. Masson. HMM profiles for network traffic classification. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, 2004.