Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing

ISI-TR-627, December 14, 2006

Genevieve Bartlett[1], John Heidemann[1], and Christos Papadopoulos[2]

[1] University of Southern California/ISI bartlett,johnh@isi.edu
[2] Colorado State University christos@cs.colostate.edu

Abstract. Blind techniques to detect network applications—approaches that do not consider packet contents—are increasingly desirable because they have fewer legal and privacy concerns, and they can be robust to application changes and intentional cloaking. In this paper we identify several behaviors that are *inherent* to peer-to-peer (P2P) traffic and demonstrate that they can detect both BitTorrent and Gnutella hosts using only packet header and timing information. We identify three basic behaviors: failed connections, the ratio of incoming and outgoing connections, and the use of unprivileged ports. We show that while individual behaviors are sometimes effective, they work best when used together. We quantify the effectiveness of our approach using two day-long traces, from 2005 and 2006, showing that they are quite accurate: BitTorrent hosts are detected with an 83% true positive rate and only a 2% false positive rate, and Gnutella hosts with a 75% true positive rate and a 4% false positive rate. Our system is suitable for on-line use, with 75% of BitTorrent hosts detected in less than 10 minutes of trace data.

1 Introduction

Identifying and filtering network traffic is central to firewalls and intrusion-detection systems. The majority of these systems deployed today use ports or packet signatures to classify traffic for filtering. While fast and effective for typical traffic, these approaches are becoming less and less effective because both ports and packet contents are easy to conceal, either intentionally or accidentally.

We see three reasons for a greater need to identify network applications by packet header information alone, rather than packet payload. First, benign traffic often varies its port usage and packet contents. For example, traffic using remote-procedure calls, multiplexed protocols such as SOAP, or UDP-based protocols (like NSF or SIP) often varies port usage and communicates ports out-of-band. An increasing use of traffic encryption hides packet-contents, both with network-level approaches like IPsec, and application-level tunnels like ssh or TLS.

Second, malware and protocols that receive mixed acceptance often intentionally hide identity by varying port usage and packet contents. Protocols such as Skype and P2P file sharing often hide themselves out of concern for restrictive use policies in some networks.

Finally, even when traffic is not accidentally or actively concealing itself, ISP policy concerns sometimes prevent analysis of data packet contents. For

example, in the United States, laws about student privacy and wiretapping can be interpreted to preclude analysis of packet data contents.

The goal of this paper is to identify network applications based on their *inherent* characteristics without considering packet contents. We therefore distinguish application behaviors that are easily changed or *incidental*, from those behaviors that are *inherent* and would incur a performance penalty or require application restructuring to change. In this sense, we are investigating blind techniques to identify applications [6].

We evaluate our approach by considering two popular P2P file sharing applications: BitTorrent and Gnutella. We evaluate our detection methods with two full day traffic traces taken from a regional ISP in 2005 and 2006, and compare our detection rates to ground truth obtained by manual analysis of the data.

The contribution of this paper is the identification and evaluation of several metrics that are applicable to blind identification of multiple types of P2P file sharing applications. We show that these metrics can detect hosts running BitTorrent applications with an 83% true positive rate with a 2% false positive rate and detect hosts running Gnutella with a a 75% true positive rate with a 4% false positive rate. Of the P2P peers caught by our system, 75% required less than 10 minutes of trace data to determine P2P activity. We suggest that our approach to identifying inherent behaviors will be applicable to other protocols.

2 Related Work

There are three general areas of related work: detection based on network port usage, packet payload, and traffic behavior.

Port- and payload-based signatures are widely used today. Unfortunately, both rely on relatively ephemeral behaviors: port assignments are easily changed, and payload contents can be hidden by encryption or randomization. We therefore do not consider these approaches further.

An alternative is to detect based on network behaviors such as an application's packet trace and communications pattern, since these can be often more difficult to conceal. Karagiannis et al., identify P2P traffic from connection patterns and the concurrent use of UDP and TCP [5]. Constantinou and Mavrommatis classify P2P traffic based on connection direction and number of peers in a connected group [3]. In later work, Karagiannis et al. introduce BLINC [6], a general classification mechanism that classifies hosts based on protocol usage, port usage and connection patterns. These methods rely on behavior that is inherent to P2P applications. While our approach is similar in that it uses inherent behavior, our metrics require significantly less state than the methods used in this previous work. Further more, we quantify our on-line detection time. We differ from BLINC further in that our metrics are selected to be specific to P2P file sharing, and work by drawing out the inherent characteristics which are unique to P2P.

Closest to our work is that by Collins et al. [2] who distinguish BitTorrent flows from FTP, HTTP and SMTP flows between pairs of hosts. They study three metrics: packet size (looking for small control messages), amount of data exchanged between hosts, and rate of failed connections. We do not consider packet

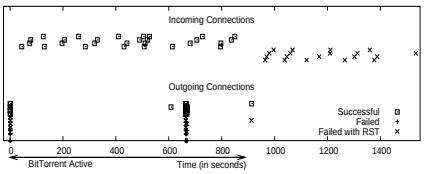


Fig. 1. BitTorrent Peer

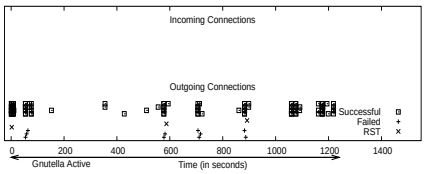


Fig. 2. Gnutella Leaf Peer

size to be an inherent metric since it is easily spoofable. The later two metrics are inherent, and we have independently determined that failed connections are an important indicator of P2P traffic. Our work differs from theirs through the addition of two other inherent behaviors (ratio of incoming-to-outgoing connections and privileged-to-non-privileged ports); by demonstrating that this approach applies to multiple kinds of P2P traffic, not just BitTorrent; and by demonstrating that our approach can operate on-line rather than post-facto.

3 Inherent Behaviors in P2P

In this section we investigate three behaviors inherent to P2P applications. In Section 4 we map these behaviors to specific metrics for detection.

Our target applications are BitTorrent and Gnutella as specific targets. Both are file sharing protocols described in detail elsewhere. For our purposes, the important characteristics of BitTorrent is that a *peer* typically contracts a tracker to find out about other peers. It then directly contacts many peers (often 20) to exchange pieces of those files. Gnutella instead uses a two-tier system of *leaf peers* and *ultrapeers*. Leaf peers typically talk only to ultrapeers, while ultrapeers communicate widely with both each other and leaf peers.

Figures 1 and 2 show new TCP connections for both BitTorrent and Gnutella. The x-axis represents time in seconds, while each connection is numbered sequentially on the y-axis. Symbols indicate when a connection is successfully started (a square) and when they fail to connect (a plus) or are terminated with a RST (an X). Figure 1 shows a BitTorrent client performing a partial BitTorrent download of a large ISO image over 15 minutes. Figure 2 depicts the connections made from a Gnutella leaf node that performs several searches and partial downloads over 1240s. Since the Gnutella peer was a leaf peer that shared no files, there were no incoming TCP connections.

3.1 Peer Coordination and Failed Connections

P2P file sharing is effective because peers share with each other directly rather than only contacting centralized servers. Since peers are end-user machines, there is considerable churn as they come and go frequently [1]. Mechanisms which track the presence of peers do so imperfectly, and this information is quickly out of date when given to a new peer. As a result, an inherent behavior of P2P sharing are many *failed attempts to contact peers* that have left the network.

At the network level, these failed contacts result in TCP RST messages from a busy or no-longer participating peer, or in multiple SYN packets attempting to start a connection and timing out. We see both these behaviors in Figure 1:

at time 0 four hosts out of ten send a reset and three hosts do not respond at all. At time 680s we observe the peer that we monitor attempt to replace a departed neighbor and there is another burst of failures. The same behavior occurs in Gnutella, where Figure 2, shows several attempted downloads (e.g. at 700s and 900s) In four out of ten of these attempted downloads, two or more sources never respond or sent a TCP reset.

This behavior is not only common to P2P traffic, but relatively uncommon among more traditional client/server applications. In client/server protocols, the servers are often well known and persistent. Failures are usually due to misconfiguration or hardware failure and there are not usually small clusters of failures.

3.2 Bidirectional Connections

P2P applications not only start connections with peers, but each peer attempts to maintain this network independently. Since peers are equivalent, this means each initiates and receives new connections. Client/server hosts instead primarily either initiate connections (clients) or receive them (servers). Thus, unlike client-server applications, an inherent behavior of many hosts in a P2P application is a balance of both incoming and outgoing connections.

We can see this in our samples, where in Figure 1, the BitTorrent peer makes 44 outgoing connections and accepts 30 incoming connections. Gnutella's two-level architecture differs, and leaf nodes initiate connections (in Figure 2 the leaf starts 5 connections to ultrapeers at time 0), but ultrapeers maintain both incoming and outgoing connections.

3.3 User Accessibility

P2P file sharing applications are typically user-level processes operating on a variety of platforms and user environments, using unprivileged ports. Thus, a P2P file-sharing connection will typically have source and destination ports above 1024, unlike server applications such as mail and web servers, which typically use well-known privileged ports.

In Figure 1, out of 44 peers contacted by the BitTorrent peer, none were listening on a privileged port. Additionally, out of the 143 unique peers suggested by the tracker, only one was listening on a privileged port. All connections in the trace used unprivileged source ports. In Figure 2, out of 119 remote peers contacted by the Gnutella peer, none were listening on a privileged port. As with BitTorrent, all connections in the trace used unprivileged source ports.

4 Implementation

The previous section outlined three P2P file sharing application behaviors which are identifiable at the network level. In this section we translate these behaviors into specific, testable metrics and describe how they can be used to perform on-line detection. Given the metrics and corresponding tests, we re-evaluate the status of each host as new associated flows appear, looking for whether that host is P2P or non-P2P. We describe this process in detail in Section 4.3.

4.1 Translating Behaviors to Metrics

Peer coordination and failed connections: As discussed in section 3.1, coordination with other peers often corresponds to bursts of failed connections. We capture this behavior with the following ratio of *failed connections*:

$$\mathbf{M_{PC}} = \frac{failed_{out}}{successful_{out} + failed_{out}}$$

where $failed_{out}$ is the total number of new outgoing connections that fail and $successful_{out}$ is the total number of new outgoing connections that were successfully established. Values of $\mathbf{M_{PC}}$ tend to be low for normal clients and servers, medium (0.1–0.8, our thresholds) for P2P hosts, and high (more than 0.8) for hosts doing port scans.

Bidirectional connections: As discussed in section 3.2, P2P clients both initiate and receive new connections. We to capture this behavior we use the following ratio of *bidirectional connections*,

$$\mathbf{M_{BC}} = \frac{successful_{in}}{successful_{out} + successful_{in}}$$

where $successful_{in}$ and $successful_{out}$ is the total number of new, successfully established, incoming and outgoing connections. The metric $\mathbf{M_{BC}}$ will be close to 1 for servers, and close to 0 for clients, and we consider values between 0.2 and 0.9 indicative of P2P hosts.

User accessibility: As discussed in section 3.3, although the individual port number varies, P2P clients connect to unprivileged ports, while other clients connect to standard servers on privileged ports. Thus we define:

$$\mathbf{M_{UA}} = \frac{successful_{user2user}}{successful_{in} + successful_{out}}$$

where $successful_{user2user}$ is the number of successful connections which have a source and destination port in the unprivileged range and $successful_{in} + successful_{out}$ is the number of total new connections at that host which were successful. For clients and servers, the expected value for ratio $\mathbf{M_{UA}}$ is near 0. Hosts doing user-level P2P run closer to 1; we consider any value over 0.2 to indicate a potential P2P host.

4.2 Metrics to Tests

We must now map individual ratios $\mathbf{M_X}$ into binary tests that confirm or disclaim P2P traffic on a host. P2P traffic corresponds to medium values of each ratio, so we define high and low thresholds h_X and ℓ_X. In general, high values indicate non-P2P behaviors (such as port-scanning), so exceeding h_X terminates the test as non-P2P host. Low values often occur when a new host appears, so we consider values below l_X as inconclusive. Values in-between the thresholds after a warm-up number of connections positively indicate a P2P host.

While each metric by itself corresponds to a specific P2P behavior, we found individual metrics to be noisy. We therefore test multiple metrics in parallel. A

negative P2P result from any metric disqualifies a host, while a positive result from all metrics is required to flag the host as P2P.

Typically, we evaluate each metric over all connections over a sliding time window until we get either positive or negative confirmation of P2P activity. However, failed connections captured by $\mathbf{M_{PC}}$ primarily occur at the beginning of a P2P session. When combining multiple metrics, $\mathbf{M_{PC}}$ often triggers before the others, but then can be "washed out" by the time the other metrics trigger. We therefore define a "sticky" equivalent $\mathbf{M_{sPC}}$, as indicative of P2P traffic provided no metric indicated non-P2P and the host was flagged as P2P over the last window of time.

4.3 System Operation

Our system runs on top of a continuous network tracing infrastructure [4]. We transform the packet-level trace into a flow-level trace by observing only the TCP SYN and SYN-ACK packets. We identify failed connections by four or more duplicate SYNs, and compute the ratio of incoming connections and privileged/non-privileged ports by looking at all flows to a particular destination. We process the data sequentially, on-line, evaluating the metrics for each source IP address once we have 10 connections or more. If at any time the metrics indicate a positive or negative result we classify the host as P2P or non-P2P and then discard any remaining information in the time window pertaining to that host. Otherwise, we continue to acquire information about that host until we reach a conclusion, or until flows time out after the configurable sliding window of time, currently set at 20 minutes.

5 Evaluation

We next evaluate our approach to determine how detection accuracy interacts with false positive rates.

Our evaluation uses network packet traces from two network taps at Los Nettos, a regional ISP in the Los Angeles area serving both commercial and academic institutions. We collected two datasets, each about 24 hours long, August 31, 2005 and October 3 2006.We see qualitatively similar results for both traces and present only the 2006 data here due to space constraints.

5.1 Detection Accuracy for BitTorrent

We first look at detection accuracy to verify that our approaches do successfully identify P2P traffic.

To establish ground truth, we classify some hosts as *known BitTorrent hosts* first by identifying flows on the default BitTorrent tracker port (6969). We then manually verify that the destination was a BitTorrent server by contacting it ourselves within several hours of the trace collection.

The Known BitTorrent section of Table 1 shows the 130 hosts we identified. We first observe that each individual metric is successful at detecting the majority of known BitTorrent hosts (85–92%), and that $\mathbf{M_{UA}}$ detects the most. Our 2005 trace (not shown here due to space) shows similar qualitative results.

Among the individual metrics, $\mathbf{M_{UA}}$ appears to be the best, with both low false positives and false negatives. However, this advantage is an artifact of our

metric:	$\mathbf{M_{PC}}$	$\mathbf{M_{BC}}$	$\mathbf{M_{UA}}$	$\mathbf{M_{sPC+BC}}$	$\mathbf{M_{sPC+UA}}$	$\mathbf{M_{all}}$
Total unique hosts: 9,656						
P2P hosts : 290						
Known BitTorrent hosts: 130						
True Positives	110 (85%)	114 (88%)	120(92%)	108 (83%)	109 (84%)	108 (83%)
False Negatives	20 (15%)	16 (12%)	10 (8%)	22 (17%)	21 (16%)	22 (17%)
Known Gnutella hosts: 160						
True Positives	123 (77%)	109 (68%)	155 (97%)	93 (58%)	120 (75%)	91 (57%)
False Negatives	37 (23%)	51 (32%)	5 (3%)	67 (42%)	40 (25%)	69 (43%)
Other Hosts : 9,366						
Likely non-P2P: 4,075						
False Positives	530(13%)	1,018(25%)	n/a	81(2%)	n/a	n/a
True Negatives	3,545(87%)	3,057(75%)	n/a	3,994(98%)	n/a	n/a
Discarded as likely-P2P: 608						
Unclassified hosts: 4,683						
Flagged as P2P	702 (15%)	1,639(35%)	1,592(34%)	140(3%)	187(4%)	70(1%)
Not flagged as P2P	3,981(85%)	3,044(65%)	3,091(66%)	4,543(97%)	4,496(96%)	4,613(99%)

Table 1. Summary of results of BitTorrent detection for 2006 Data Set.

ground truth—since non-BitTorrent traffic (described in the next section) always includes some privileged ports, this metric is artificially perfect. Unfortunately, it will also consistently classify *any* applications that use only non-privileged ports as P2P, including direct user-to-user chat programs and games. Our simple definition of ground truth places these applications in the unknown category and therefore excludes them from this analysis. However, we suggest that UA is useful in $\mathbf{M_{all}}$ because it helps eliminate the few client-server applications that would be detected as a false positive by $\mathbf{M_{sPC+BC}}$.

Finally, we observe that the combined metrics $\mathbf{M_{sPC+BC}}$ and $\mathbf{M_{all}}$ perform almost as well as the stand-alone metrics at detecting true positives (83% and 84% compared to 85–92%), and the combined metrics perform much better in reducing false positives (2% instead of 13–25%).

5.2 Understanding false positive rate

Even if the system performs well at detecting P2P hosts, it will not be useful if it also falsely tags many non-P2P hosts as P2P. We therefore evaluate the false positive rate of individual and combined metrics.

Although it is easy to confirm the presence of known P2P traffic to a host, it is significantly more difficult to prove absence of P2P traffic. To establish a rough body of non-P2P hosts, we first remove all known P2P hosts from our population and select half of the remaining hosts. (We will use the other half in Section 5.4.) While these hosts include no known P2P hosts (those running on well known P2P ports), there may be some hosts using non-standard ports. We assume these non-standard ports are non-privileged, so we therefore discard 608 hosts that have only non-privileged-to-non-privileged ports as "potential P2P". We label the remaining hosts as *likely non-P2P*. This decision is conservative since we know that it is possible to tunnel P2P traffic over well-known ports (such as web port 80).

We use this set of likely non-P2P hosts to look for false positives in our metrics. We expect that individual metrics will have some number of false positives: port scanners and misconfigured machines or servers can accidentally trigger $\mathbf{M_{PC}}$, and some services that have bidirectional traffic (such as DNS) and user machines that host some servers can trigger $\mathbf{M_{BC}}$. Note that we cannot consider $\mathbf{M_{UA}}$ with this methodology because our definition of likely non-P2P distorts this metric.

The likely non-P2P section of Table 1 shows these results. Individual metrics show moderate-to-high false positive rates (13–25%). Because the number of likely non-P2P hosts is so much larger than the number of known P2P hosts, these false positive rates imply 5–10 errors for every true positive. Such high false positive rates mean that an individual metric is impractical without additional confirmation. Examination of specific traces suggest that many $\mathbf{M_{PC}}$ failures are due to false identification of port scans as P2P. We examined a few cases of $\mathbf{M_{BC}}$ failure; they were typically due to user hosts that also run small server applications.

Our hope is that combining multiple metrics can reduce the false positive rate, and $\mathbf{M_{sPC+BC}}$ shows a false positive rate of only 2% rather than 13–25%. This success is because the false positives are triggered by different circumstances. Combining all three metrics in $\mathbf{M_{all}}$ eliminates all false positives, but as described above this is an anomaly due to our definition of likely-non-P2P.

From our evaluation of true and false positives we conclude that the combined metrics are essential to get good accuracy and few false positives. The combined metrics show only a few percent reduction (2–5%) in detection accuracy for BitTorrent (although a larger drop for Gnutella, 19%, when using all metrics), while the percent of false positives is cut in four.

5.3 Effectiveness for Gnutella

Since our detection methods are based on behaviors of P2P applications in general, and not specific to the BitTorrent protocol, we expect that our system is capable of detecting hosts running other P2P applications. To test this claim we next evaluate our approach on Gnutella hosts.

We establish Gnutella ground truth as all hosts that contact known Gnutella ultrapeers. We track Gnutella ultrapeers by joining the Gnutella network repeatedly on the day of trace collection and recording lists of the suggested ultrapeers.

Some protocol differences between BitTorrent and Gnutella affect our metrics, however. We expect that metrics $\mathbf{M_{PC}}$ and $\mathbf{M_{UA}}$ will perform well at detecting Gnutella, but because of Gnutella's two-tiered architecture, $\mathbf{M_{BC}}$ will not perform nearly as well.

The *known Gnutella* section in Table 1 shows that $\mathbf{M_{PC}}$ alone detects 77% of the Gnutella hosts. However, as discussed in Section 5.2, this metric alone has a high false positive rate and so we need combined metrics to reduce false positives. We see that $\mathbf{M_{all}}$ is still fairly effective at detecting Gnutella, detecting 57% of the known Gnutella hosts.

We observed that $\mathbf{M_{BC}}$ does not work well with Gnutella because only ultrapeers are bidirectional, not Gnutella leaf nodes unless a leaf peer is uploading.

We observe that $\mathbf{M_{sPC+UA}}$ detects nearly as many true positives as $\mathbf{M_{PC}}$ alone (75% vs. 77%), but significantly decreases false positives. For networks where Gnutella is very prevalent, this metric may be preferable to $\mathbf{M_{all}}$.

5.4 Estimating previously undetected P2P hosts

Our above analysis isolated traffic into known-P2P and likely-non-P2P categories to study the accuracy of our approaches. We next look at unclassified traffic to estimate how many hosts appear to be P2P file sharing but escaped identification as known-P2P.

To estimate P2P traffic in unclassified traffic, we start with the half of hosts not considered above. These hosts exclude all traffic to known trackers, so they all would be unclassified by detection schemes using known sites. We then run our detection algorithms on these hosts and examine the hosts flagged as P2P.

The *unclassified* section of Table 1 shows our estimate of P2P traffic in this sample of hosts. Our combined metrics flagged 1.5% (70 hosts) of the unclassified hosts as running P2P applications. Our analysis of the false positive rate of $\mathbf{M_{sPC+BC}}$ suggests that at least half of these are true positives. Although we do not know the likely true positive rate for $\mathbf{M_{all}}$, it should be greater since $\mathbf{M_{all}}$ reduces further the number of flagged hosts by including metric $\mathbf{M_{UA}}$.

To confirm that some of these 70 hosts have P2P traffic we looked at what ports they use. Of these 70 hosts, 17 made connections to remote hosts on default BitTorrent ports (6969, 6881–6888) and 15 made connections to remote hosts on the default Gnutella port (6346), strongly suggesting that we successfully found true P2P traffic. (If the host had contacted a known tracker or ultrapeer we would have already classified it as known-P2P.) We conjecture that some of the other hosts were doing P2P sharing on non-standard ports, although we could not confirm that at the time. Finally, our analysis of these unclassified hosts sheds some light on the benefit of adding $\mathbf{M_{UA}}$ to form $\mathbf{M_{all}}$—this addition reduces the number of hosts identified as potential P2P in half (70 vs. 187). Of the 17 hosts just described none are eliminated, suggesting (but not proving) that $\mathbf{M_{all}}$ does not reduce the true positive rate.

5.5 Detection speed

As well as being accurate, we wish to detect P2P hosts quickly. To estimate detection time we consider known P2P hosts. We identify the first contact with a known tracker or ultrapeer as the "start" time and then determine how much later we classify that host as P2P. This start time corresponds to an unrealizable, idealized detection system based on a perfectly known P2P network, and so it represents a conservative estimate of our detection time.

Figure 3 shows CDFs of detection time for both BitTorrent and Gnutella for our two traces. As is shown, about 75% of the time we can identify a P2P client in less than ten minutes, and about one-fifth of the time we can decide within a minute. Given that P2P applications often run for tens of minutes, we

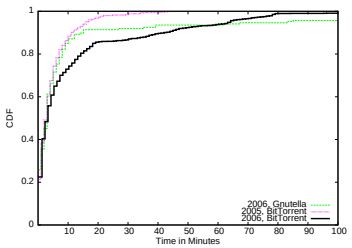


Fig. 3. Time until detection

believe these detection times are more than sufficient for on-line identification.

To understand the cause of the delay we looked at how the combined metrics operate. For BitTorrent, $\mathbf{M_{PC}}$ triggers quickly, but $\mathbf{M_{BC}}$ is much slower. These timings are consistent with the behaviors they track, since failed connections occur when a new peer starts up and actively probes other peers, while bidirectional communication happens only later as other peers learn about the target host and connect to it. In the case of Gnutella, $\mathbf{M_{PC}}$ often does not trigger till several minutes after contacting ultrapeers. We believe this slower trigger is because ultrapeers are more reliable than typical BitTorrent peers, and with Gnutella $\mathbf{M_{PC}}$ is triggered only when a peer attempts to contact remote resources and download files.

In addition to the delay described above, our current implementation batches packet traces into 2–6 minute segments. Thus actual delay in our current implementation is up to 16 minutes 75% of the time. This batching is due to our data collection system [4] and could be greatly reduced or eliminated by integrating trace collection with metric evaluation.

5.6 Parameter sensitivity

Our system has several parameters that affect operation, including the size of the sliding time window, minimum number of connections considered, and thresholds. Due to space limitations we provide a detailed evaluation of these factors in Appendix B.

6 Conclusions

We have shown that one can map inherent P2P behaviors into metrics that allow on-line detection of P2P hosts. We showed that a combination of metrics allows for high accuracy with low false positives with the majority of hosts detected in less than 10 minutes.

Acknowledgments and Data Availability

We would like to thank Los Nettos for facilitating and granting access to trace collection. We would also like to thank Mark Baklarz and Steve Sutor of USC for discussions on current practices for tracking P2P users at USC.

This material is based on work supported by the United States Department of Homeland Security contract number NBCHC040137 ("LANDER"). It is also supported by the National Science Foundation (NSF) under grant number CNS-0626696, "NeTS-NBD: Maltraffic Analysis and Detection in Challenging and Aggregate Traffic (MADCAT)". All conclusions of this work are those of the authors and do not necessarily reflect the views of the sponsors.

Our data sets are made publicly available through the Predict project as `USC-LANDER-p2p_detection-20050831` and `USC-LANDER-p2p_detection-20061003`.

References

1. M. Bawa, H. Deshpande, and H. Garcia-Molina. Transience of peers and streaming media. In *Proceedings of the ACM HotNets I*, pages 107–112, Princeton, NJ, USA, October 2002.

2. M. Collins and M. Reiter. Finding peer-to-peer file-sharing using coarse network behaviors. In *Proceedings of the European Symposium On Research In Computer Security*, Hamburg, Germany, September 2006.
3. F. Constantinou and P. Mavrommatis. Identifying known and unknown peer-to-peer traffic. In *IEEE International Symposium on Network Computing and Applications (NCA)*, pages 93–102, Cambridge, MA, USA, July 2006.
4. A. Hussain, G. Bartlett, Y. Pryadkin, J. Heidemann, C. Papadopoulos, and J. Bannister. Experiences with a continuous network tracing infrastructure. In *Proceedings of the ACM SIGCOMM Workshop on Mining network data Mine Net*, pages 185–190, Philadelphia, PA, USA, August 2005.
5. T. Karagiannis, A. Broido, M. Faloutsos, and kc claffy. Transport layer identification of p2p traffic. In *Proceedings of the ACM SIGCOMM Workshop on Internet Measurement (IMC)*, pages 121–134, Taormina, Sicily, Italy, October 2004.
6. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM Conference*, pages 229–240, Philadelphia, PA, USA, August 2005.

APPENDIX

A Evaluation of Second Dataset

To further determine the effectiveness of our approach, we perform a second evaluation of our metrics using our second data set from 2005. We expect that the results from the 2005 data set are similar to the results obtained from the 2006 data set. Our results over the 2005 data set are summarized in Table 2.

metric:	$\mathbf{M_{PC}}$	$\mathbf{M_{BC}}$	$\mathbf{M_{UA}}$	$\mathbf{M_{sPC+BC}}$	$\mathbf{M_{sPC+UA}}$	$\mathbf{M_{all}}$
Total unique hosts: 10,415						
P2P hosts : 251						
Known BitTorrent hosts: 251						
True Positives	210 (84%)	219 (87%)	225 (90%)	201 (80%)	204 (81%)	201 (80%)
False Negatives	41 (16%)	32 (13%)	26 (10%)	50 (20%)	47 (19%)	50 (20%)
Other Hosts : 10,415						
Likely non-P2P: 4,378						
False Positives	525 (12%)	1,313 (30%)	n/a	175 (4%)	n/a	n/a
True Negatives	3,853 (88%)	3,065 (70%)	n/a	4,203(96%)	n/a	n/a
Discarded as likely-P2P: 704						
Unclassified hosts: 5,082						
Flagged as P2P	711 (14%)	1,677 (33%)	1,575 (31%)	101 (2%)	152 (3%)	50 (1%)
Not flagged as P2P	4,371 (86%)	3,405 (67%)	3,507 (69%)	4,981 (98%)	4,930 (97%)	5,032 (99%)

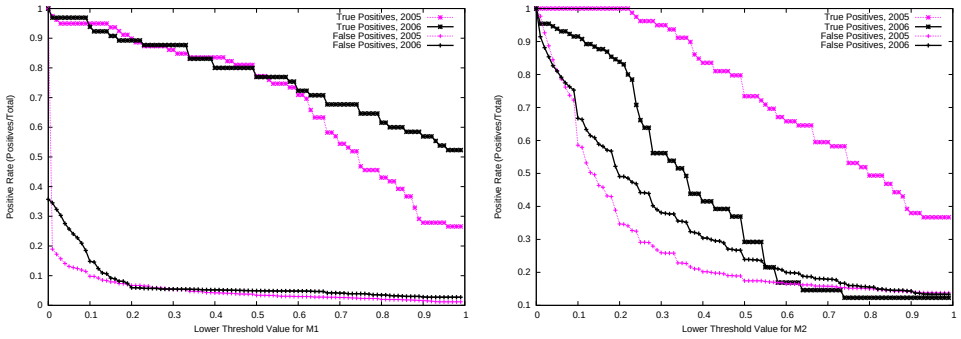
Table 2. Summary of results of BitTorrent detection for 2005 Data Set.

A.1 Detection accuracy for BitTorrent in 2005 data

Following the same methodology as described in section 5.1, we examine our 2005 data and show in this section that our detection accuracy is similar across both data sets.

The Known BitTorrent section of Table 2 shows the 251 hosts we identified in our 2005 data set as running BitTorrent. As expected, our detection accuracy with the 2005 data set is similar to our 2006 data set. Individual metrics are successful at detecting the majority of BitTorrent hosts (84–90%). Combined metrics perform almost as well (80–81%) as individual metrics over the 2005 data set.

These results are comparable to the results we obtained using our 2006 data set. As seen in the 2006 data set, combining metrics slightly decreases the true positive rate, but significantly decreases the false positive rate (up to 26%) as shown by the combined metric $\mathbf{M_{sPC+BC}}$. Again, false positive rates involving metric $\mathbf{M_{UA}}$ are not included in the table because our definition of likely non-P2P hosts distorts the false positive rate for metric $\mathbf{M_{UA}}$.



(a) $\mathbf{M_{PC}}$ True and False Positive Rates (b) $\mathbf{M_{BC}}$ True and False Positive Rates

Fig. 4. Effect of $\mathbf{M_{PC}}$ and $\mathbf{M_{BC}}$ Lowerbound Thresholds

B Result Sensitivity

We use several fixed constants in our system: the metric thresholds, the sliding time window size and the minimum number of connections considered before a decision can be made. Ideally, varying these constants will have little to no effect on our results. In this appendix, we will demonstrate how varying our constants affects our results.

B.1 Sensitivity to Threshold Selection

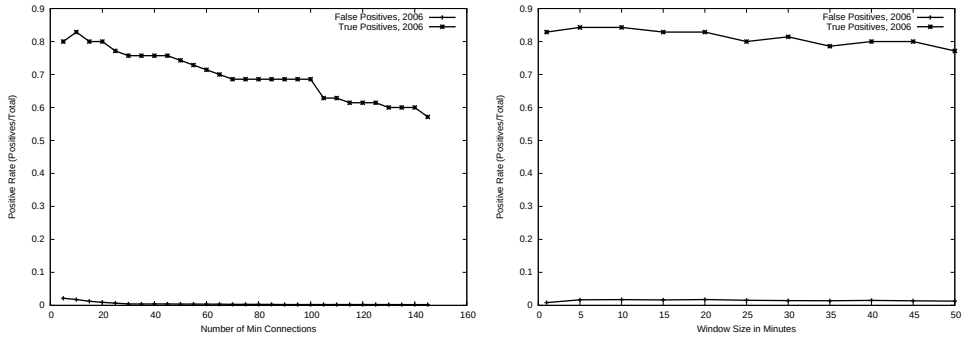
Our first set of constants are the lower and upper threshold bounds for each of our metrics. We have set our threshold values fairly permissive, allowing a large range of values to trigger a metric. In this section we explore how the lower bound thresholds affect the distinguishing ability of metrics $\mathbf{M_{PC}}$ and $\mathbf{M_{BC}}$. We do not consider metric $\mathbf{M_{UA}}$ in this discussion due to the lack of good ground truth in determining false positives.

We expect that there is no single cutoff point for each metric at which the metric performs ideally. There are two separate reasons a single cutoff point is problematic.

First, our methods are designed to detect a variety of types of peers. The variety of behaviors leads to a variety of metric values. For example, a BitTorrent peer joining a peer group with a high rate of churn will have a much higher value for $\mathbf{M_{PC}}$ than a Gnutella leaf peer performing a search in which only a few sources are non-responsive.

Second, P2P behaviors at a host may be somewhat masked by other ongoing activities at a host. If a user is surfing the content on several reliable web servers while running a P2P peer, the successful HTTP connections will lower the value of $\mathbf{M_{PC}}$. Similarly, if a user follows several bad links to web servers which are down, the failed connections to these servers will raise the value of $\mathbf{M_{PC}}$ independently from the host's P2P activity.

As shown in Figure 4(a), metric $\mathbf{M_{PC}}$ is a relatively strong distinguisher after a lower bound cutoff of 0.2. This is not true for metric $\mathbf{M_{BC}}$ as seen in



(a) Varying Minimum Connections (b) Varying Sliding Window Size

Fig. 5. Effects of Chosen Constants

Figure 4(b). This is partly because there are many non-P2P activities which can trigger $\mathbf{M_{BC}}$ including other user-to-user programs or a mix of server and client activities.

As shown in section 5.2, it is the combination of metrics which reduces the false positive rate. Combining $\mathbf{M_{PC}}$ with $\mathbf{M_{BC}}$ reduces the false positive rate 8–26%.

B.2 Sensitivity to Window Size

Our approach computes metrics based on connections in a sliding time window. During our evaluation of our method we set our sliding time window size to be 20 minutes.

We expect that our results are not directly dependent on the size of the sliding window since our metrics are updated per flow and not computed over the entire time window. However, too large of a time window may cause undesirable merging of two distinct dynamic hosts which during the window share a single IP. To minimize the potential problem of dynamic hosts, we do not examine window sizes greater than 50 minutes.

To confirm the independence of our detection ability on our sliding window size, we examined window sizes from 1 to 50 minutes, varying the window size by 5 minutes at a time. Figure 5(b) shows the effect of the window size the true positive rate and false negative rate.

As expected, there is no significant increase in the false positives at any specific window size. There is, however, a very slight decline in the true positive rate as the window size increases. This is due mostly to $\mathbf{M_{sPC}}$ not triggering because activities at a host earlier during the time window "wash out" bursts of failed connections which occur later in the time window.

B.3 Detection Sensitivity to Minimum Connections Threshold

As with our sliding window size, the minimum number of connections we consider before attempting to make a decision is a fixed number. We wish to show that

our choice of requiring at least 10 connections before a decision is made reduces the false postive rate without significantly reducing the true postive rate.

We expect that there is a range of threshold values for the minimum number of connections considered for which the results are roughly the same; however, at some point the minimum number of connections considered does greatly affect the results. If too few connections are considered, the likelihood of a host being falsely identified as P2P is expected to increase since metrics can be easily triggered over a small number of connections. If too many connections are considered, metrics are likely to be "washed out" by other activities at the host.

We examined from 5–150 minimum connections. Figure 5(a) summarizes the effect of minimum connections considered on the false positive rate and true positive rate. There is a slight increase in false positives at fewer than 10 connections, which is consistent with our expectations.

The true positive rate drops steadily above 20 connections and significantly above 100 connections. Partly, the drop in true positives is due to metrics being diluted by non-P2P activities at the host as more and more connections are considered in the decision. The overall steady decline in true positives is due to peers never making the required minimum connections. Requiring too many connections before a decision is made excludes peers which are relatively idle and never make the minimum number of connections during the time window.

Setting our minimum connection threshold to 10 avoids eliminating relatively idle peers while still reducing the false positives seen when too few connections are considered.