A Longitudinal Study of P2P Traffic Classification

Alok Madhukar **Carey Williamson**
Department of Computer Science
University of Calgary
Email: {madhukar,carey}@cpsc.ucalgary.ca

Abstract

This paper focuses on network traffic measurement of Peer-to-Peer (P2P) applications on the Internet. P2P applications supposedly constitute a substantial proportion of today's Internet traffic. However, current P2P applications use several obfuscation techniques, including dynamic port numbers, port hopping, HTTP masquerading, chunked file transfers, and encrypted payloads. As P2P applications continue to evolve, robust and effective methods are needed for P2P traffic identification. The paper compares three methods to classify P2P applications: port-based classification, application-layer signatures, and transport-layer analysis. The study uses empirical network traces collected from the University of Calgary Internet connection for the past 2 years. The results show that port-based analysis is ineffective, being unable to identify 30%-70% of today's Internet traffic. Application signatures are accurate, but may not be possible for legal or technical reasons. The transport-layer method seems promising, providing a robust means to assess aggregate P2P traffic. The latter method suggests that 30%-70% of the campus Internet traffic for the past year was P2P.

Keywords: Network Traffic Measurement, Peer-to-Peer, Traffic Analysis

1. Introduction

Recent measurement studies [1, 7, 10, 17, 18, 19] indicate that Peer-to-Peer (P2P) applications generate a substantial volume of Internet traffic. Understanding the Internet traffic profile is important for several reasons, including traffic engineering, network service pricing, and capacity planning. Internet traffic cannot be managed properly if it cannot be measured properly.

P2P traffic measurement is especially important for Internet Service Providers (ISPs), for several reasons. First, many P2P applications are bandwidth-intensive. Excessive network congestion could lead to dissatisfied customers and possible customer churn. Second, increasing the network capacity is expensive, and only effective on a short-term basis. P2P application traffic may soon expand to occupy the increased capacity as well, making network congestion inevitable. Third, some current Internet access technologies have asymmetric upstream and downstream bandwidths, to exploit existing access technologies while limiting operating costs for the ISPs. The underlying assumption is that Internet users download much more than they upload, as they do with the Web. However, in P2P applications, users may upload as much as they download. If a large proportion of the Internet traffic is P2P, then the underlying assumption of traffic asymmetry may be invalid.

ISPs can formulate traffic management policies based on knowledge of P2P traffic. ISPs might differentiate the services into premium and ordinary classes, applying differentiated charges based on traffic type instead of the flat pricing charges that are currently prevalent. Another possibility is for an ISP to provide a file sharing proxy [7] to reduce Internet backbone traffic and operational costs. For traffic-specific network engineering, the ISP must be able to classify Internet traffic, which highlights the need for P2P traffic detection.

There is also an important legal issue since some content found in file sharing systems (e.g., music, films, eBooks, games) may violate copyright laws. Some recent trials have held ISPs responsible for those copyright infringements; this precedent strongly encourages ISPs to detect and pursue abusing users.

Identifying P2P traffic on today's Internet is challenging, since P2P applications are evolving rapidly. Many P2P applications use one or more techniques to conceal their presence on the network. These techniques include dynamic assignment of random port numbers, the use of multiple transport-layer connections, chunked file transfers, HTTP masquerading, and encrypted payloads.

This paper provides a comparative evaluation of the effectiveness of three different P2P traffic classification techniques: port-based analysis, application-layer signatures, and transport-layer heuristics. The port-based analysis method is a classical approach based on well-known port numbers assigned by IANA (Internet Assigned Numbers Authority) [10]. Since current P2P applications transmit data on randomly assigned ports, this method is no longer accurate.

The novelty in our work is the use of a 2-year dataset to provide a longitudinal assessment of the effectiveness of P2P traffic classification techniques. We use empirical network traces from the University of Calgary Internet connection. The network traces were collected using tcpdump [10] over a period of two years since September 2003.

Our results show that port-based analysis is ineffective for classifying P2P applications: 30-70% of the Internet traffic is classified as "unknown". While the application-layer signature approach is accurate for classifying P2P applications, many reasons (e.g., legal and technical) preclude its use in practice. A modified version of the transport-layer method [10] is also tested. The results show that the transport-layer method provides a promising way to measure aggregate P2P traffic.

The rest of the paper is organized as follows. Section 2 provides background on P2P applications, and a brief summary of related work. Section 3 describes the data collection method and infrastructure. Section 4 describes the port-based analysis. Section 5 discusses the application-layer method. Section 6 discusses the transport-layer method for P2P traffic classification. Section 7 concludes the paper.

2. Background and Related Work

2.1 Background

P2P applications have been evolving at a very rapid rate. Within a span of six years (since 1999), P2P applications have evolved from first generation to second generation to third generation. One reason for the rapid evolution of these applications has been the desire to avoid detection.

The first generation [15] of P2P systems consisted of centralized systems like Napster. A centralized server was used to index files, making it relatively easy to locate the server and block it. Also, P2P applications used well-known ports to transfer data, so that it was easy for network operators to identify the P2P traffic, and consequently block the corresponding ports to discourage P2P traffic.

The second generation [15] of P2P systems includes protocols like Gnutella [5]. Gnutella was a completely distributed system, where queries were flooded to neighbors. Peers also used dynamically assigned ports to transfer data, so that it was difficult to classify P2P traffic.

Third generation [15] P2P systems are quite sophisticated. These are hybrid systems that combine ideas from centralized systems and distributed systems. There is a notion of supernodes as in KaZaA or ultrapeers as in Gnutella2. The supernodes/ultrapeers have comparatively more computing resources than other neighboring peers and are responsible for handling index files for a subset of peers. They often transmit data using randomly chosen ports. Sometimes, they disguise their traffic by using ports of other well-known applications. Also, a single large file can be downloaded simultaneously in smaller pieces from several other peers. Furthermore, a few protocols like FastTrack have started encrypting the application-layer data in the packets. These techniques make it harder to detect P2P traffic.

2.2 Related Work

Sen et al. [19] developed an approach for finding P2P traffic through application-layer signatures. They examine available documentation and packet-level traces to identify application-layer signatures, and then utilize these signatures to develop filters that can track P2P traffic on high-speed network links.

Their study analyzes TCP packets in the download phase of file transfer. They decomposed P2P signatures into fixed pattern matches at fixed offsets within a TCP payload, and variable pattern matches with variable offset within a TCP payload [19].

Their study used two packet traces from different network vantage points for the experiments. The first trace of Internet backbone traffic was collected on an access network for two days in November 2003. The second trace was collected on a 45 Mbps link connecting a Virtual Private Network (VPN) with 500 employees to the Internet. The trace spanned six days of November 2003. P2P traffic was supposedly blocked on this network.

The authors evaluated the accuracy and scalability of the application-layer signature technique. The accuracy was evaluated by determining measures for false positives and false negatives. False positives measure the amount of traffic that the classifier erroneously classified as P2P. False negatives measure the amount of P2P traffic that the classifier failed to identify as a P2P application.

The results show that their technique has less than 5% false positives and false negatives, indicating that the technique is accurate most of the time. They also contend that the technique is scalable based on the number of packets examined before the application was classified. In a subsequent study, Haffner et al. [8] proposed a technique to determine application signatures automatically using machine learning algorithms. The error rate is shown to be below 1%.

Karagiannis et al. [10] provide a novel approach to identify P2P flows at the transport layer (i.e., based on connection patterns), without relying on packet payloads. The study examines packet-level traces collected on three different days (May 5, 2003, as well as January 22 and February 25, 2004) on an OC-48 Tier-1 ISP link. The first trace contained the first 44 bytes of each TCP packet, containing 0 or 4 bytes of payload, while the remaining two traces contained 16 bytes of payload for each packet.

The transport-layer method relies primarily on two heuristics. The first heuristic identifies source-destination IP pairs that concurrently use both TCP and UDP. If such IP pairs exist and they do not use specific well-known ports, then these flows are considered P2P. The second heuristic considers the structural pattern of transport-layer connections between hosts. In particular, for P2P applications, the number of distinct ports connected to a host often matches the number of distinct IP hosts connected to it.

3. Datasets

3.1 Data Collection Infrastructure and Methodology

The primary datasets used in our study come from passive network traffic measurements of the University of Calgary campus network. Our data collection infrastructure was deployed at the main campus router with the help of staff from University of Calgary Information Technologies (UCIT), the administrators of the campus network.

Since September 2003, the campus has been connected to the Internet via a 100 Mbps full-duplex Ethernet link. The traffic on that link is currently forwarded (using port mirroring) to our monitor via a 1 Gbps half-duplex Ethernet link. See Figure 1. The monitoring machine is a dual-processor Dell (1.4 GHz Pentium III processors) with 2 GB RAM and 119 GB of disk.

We use tcpdump [20] for data collection on the monitoring machine. We record the headers of all TCP/IP packets with the SYN, FIN, or RST flags set. These headers are recorded to a file, with a new file created for each 1 hour of network traffic, thus creating 24 files for each day. The recorded files are then moved off the monitoring machine to a server in the ELISA (Experimental Laboratory for Internet Systems and

Applications) laboratory in the Department of Computer Science at the University of Calgary. The files are then analyzed offline using custom analysis tools.

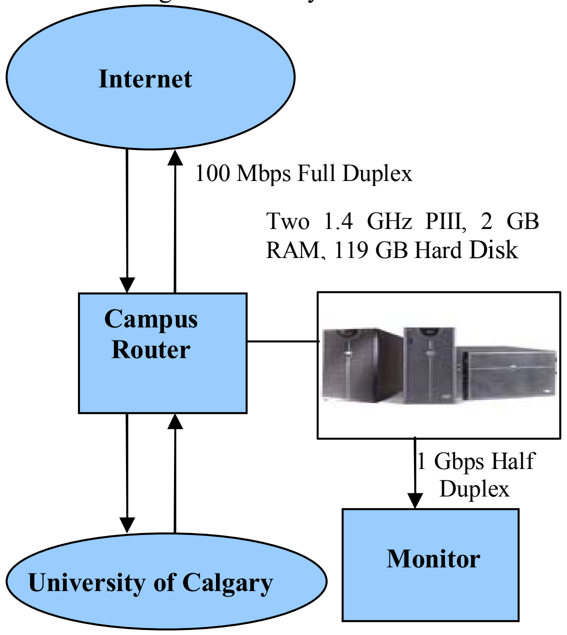


Figure 1: Data Collection Infrastructure.

Recording only TCP SYN, FIN, and RST packets is a well-known technique, and justified as follows. First, TCP has been the dominant transport-layer protocol on the Internet for over a decade, carrying over 95% of Internet traffic [21]. Second, this technique reduces trace storage requirements significantly, enabling long-term traffic studies, while still facilitating connection-level analysis [16].

3.2 Overview of Datasets

This section provides an overview of data sets that were examined for this paper. Data was collected for 2 years, with 24 trace files per day, totaling over 17,500 files. Each trace file represents 1 hour of network activity. Only TCP SYN, FIN, and RST headers are recorded.

The aggregate network traffic volumes are represented in the next three graphs, for time scales ranging from 1 day to 1 year. Traffic byte counts are determined from TCP connection-level analysis by subtracting starting (SYN) sequence numbers from ending (FIN) sequence numbers. Byte volumes are then converted into average data rates based on the trace interval analyzed.

Figure 2 shows network activity for a typical 24-hour period. Inbound traffic and outbound traffic are plotted separately. Each point represents the average data rate for a 5-minute interval. The horizontal axis shows the number of hours since midnight (0h) on Thursday, October 20, 2005.

The inbound traffic (destined to the University of Calgary) shows the expected diurnal traffic patterns, consistent with the working day. There is little traffic prior to 8am. The traffic volume increases from 8am onwards, peaking in late morning and remaining steady for most of the afternoon. The traffic gradually decreases after 4pm. The inbound traffic varies significantly during the day, ranging from 4 Mbps to 76 Mbps, and is most pronounced from 9am to 6pm.

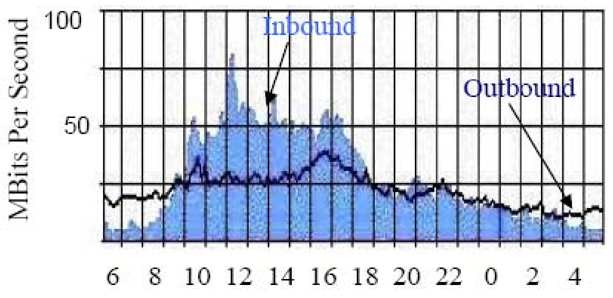


Figure 2: Network Activity for One Day (Oct 20-21, 2005).

The outbound traffic (destined to the Internet) shows different characteristics. This traffic is steadier, ranging from 12 Mbps to 38 Mbps. During the working hours, the inbound traffic exceeds the outbound traffic, implying that the campus is a net consumer of data during this period. During non-working hours, the outbound traffic exceeds the inbound traffic, so that the campus is a net producer of data during this period. Overall, during a day, the U of C campus is a net consumer of data.

Figure 3 shows a similar plot of the aggregate traffic volume, though for a longer time period (just over 1 week). Each point represents an hourly average.

Figure 3 shows that traffic volume is more pronounced on weekdays than on weekends (as expected). Inbound traffic dominates outbound traffic on weekdays, while outbound traffic dominates inbound traffic on weekends. The campus is a net consumer of data on weekdays and a net producer of data on weekends.

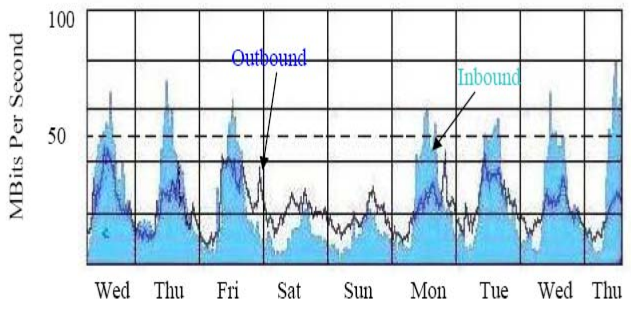


Figure 3: Network Activity for One Week (Oct 19-27, 2005).

Figure 4 represents the network activity for a one-year period (September 2004 to October 2005). Each point represents the daily average data transfer rate. At this time scale, the structure of the university academic year is evident. Network activity was most prominent during the months of Fall 2004 and the Winter 2005 semester. The average data rate ranged from 18 Mbps to 54 Mbps during these months. For most days, the inbound traffic exceeds the outbound traffic, meaning that the campus is a net consumer of data.

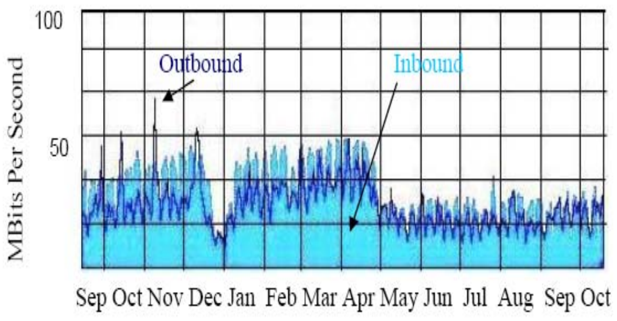


Figure 4: Network Activity for One Year (Sept 2004-Oct 2005).

Traffic volume decreased dramatically during the Christmas holidays (late December 2004). The network activity is also lower during the spring and summer months of 2005, before ramping up again in Fall 2005.

4. Port Analysis

This section presents results for the traditional approach to Internet traffic analysis, namely the use of well-known ports for traffic classification.

4.1 Port Analysis Tool

A simple tool for port analysis was developed in the C programming language. The program iterates through all the packets in the trace and creates a flow table. For each packet, the total number of packets is incremented by 1, and the total number of bytes is incremented by the IP packet length. Within the IP header, the protocol number of the packet is examined. If the protocol number is 6 (TCP), then the TCP header of the packet is examined. The TCP port numbers are extracted, and the packet is classified accordingly. Well-known TCP ports are hard-coded in the analysis tool based on the ports identified on the IANA Web site [10].

4.2 Results from Port Analysis

Figure 5 shows the results from port classification for the two-year period from September 2003 to August 2005. The horizontal axis of the plot represents time, while the vertical axis represents the percentage of TCP flows for certain popular applications, such as Web, electronic mail, and Gnutella. Each point represents the daily average, computed from 24 trace files.

Figure 5 shows that SMTP was the most dominant traffic port for the first few months of the trace, accounting for 30-70% of the TCP flows during that period. This traffic represents a local anomaly at the University of Calgary. From the IT technical support staff, we learned that this traffic anomaly resulted from misconfiguration of a campus-level mail server. The problem was rectified in January 2004.

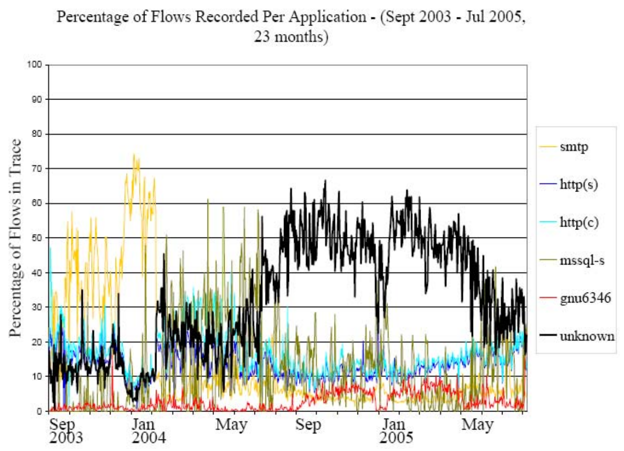


Figure 5: Port Analysis Results for a Two-Year Period (Sept 2003-Aug 2005).

Another prominent network application in Figure 5 is the Web. The line denoted HTTP(c) represents the inbound HTTP client traffic from the external Internet to University of Calgary Web servers. HTTP(s) represents traffic from the University of Calgary to external Web servers. The graph shows that, on average, HTTP traffic ranged between 10% and 30% of the total traffic volume. It is the dominant traffic type in early 2004, but less prominent thereafter.

The bold black line in Figure 5 shows the "unknown" traffic: traffic that could not be classified into any known application based on port analysis. The unknown traffic was relatively low from Sept 2003-Jan 2004 (when the SMTP anomaly was present), but increased sharply in February 2004. It then remained between 10%-30% until April 2004. Unknown traffic became the most dominant traffic during the period from September 2004 to July 2005. During the academic year, 40-65% of the flows are unknown traffic.

Figure 5 also shows significant traffic on the Microsoft SQL Server port (TCP port 1433). There are two possible reasons for this. First, the traffic on this port is known to represent worm traffic [12]. Second, the IT division at the University of Calgary is building a data warehouse based on the Microsoft SQL Server, so it might be legitimate network traffic.

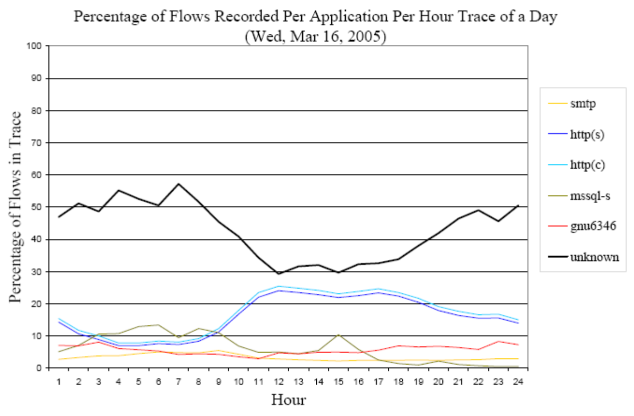


Figure 6: Port Analysis Results for One Day (Wed March 16, 2005).

Figure 6 provides a finer-grain look at the port analysis results for a typical day (Wednesday, March 16, 2005). Each point represents the percentage of TCP flows in each category, from a one hour trace. There are 24 traces for a day. The figure shows that unknown traffic is more pronounced at night than during the day. The primary reason is the strong presence of HTTP traffic during the working day from 10am to 6pm.

Figure 7 shows the port analysis results for one week of data from March 16 to 22, 2005. As in Figure 6, each point represents the percentage of flows for an hour of trace. Similar to Figure 6, Figure 7 shows that unknown traffic is more pronounced at night, whereas HTTP traffic is more pronounced during weekdays. On weekends, the proportion of HTTP traffic decreases, and the proportion of unknown traffic increases.

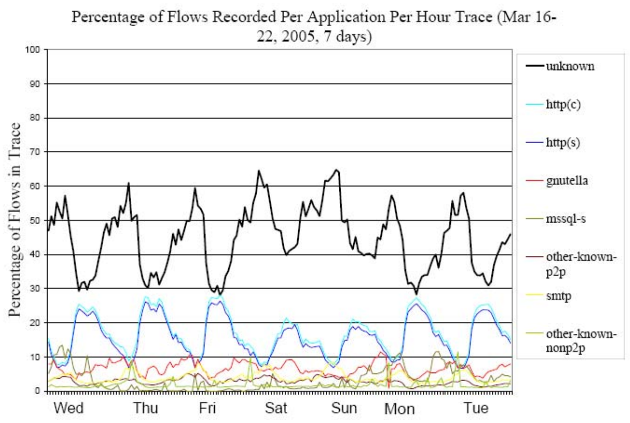


Figure 7: Port Analysis Results for One Week (March 16-22, 2005).

The foregoing graphs show that the application mix seen on the network varies with the time of day. Figure 8 and Figure 9 provide a summary of this behavior.

Figure 8 shows a noon-hour view of the total traffic, classified into ten categories, for the period March 16-31, 2005. It can be seen that unknown traffic and HTTP traffic are the most dominant, followed by Gnutella.

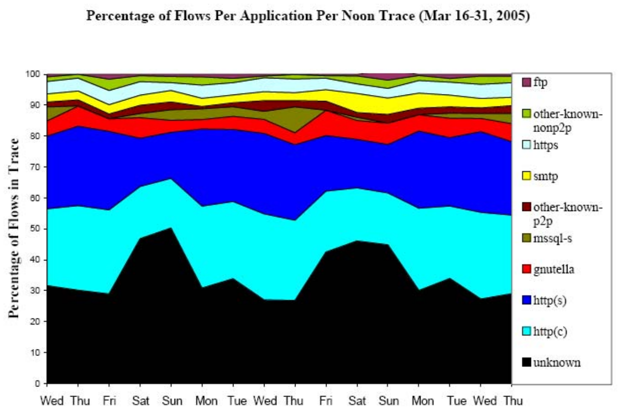


Figure 8: Noon View of Traffic Profile from Port Analysis (Mar 16-31, 2005).

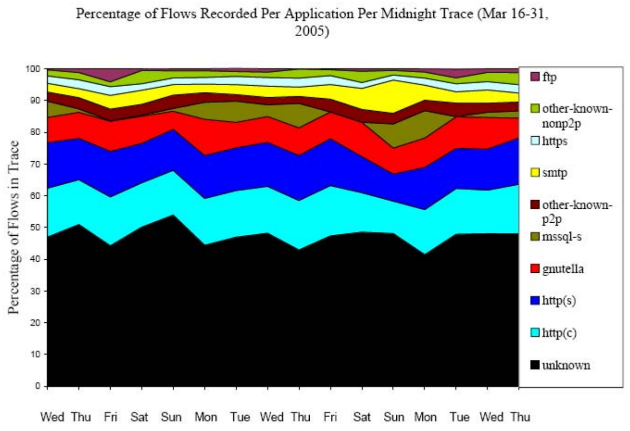


Figure 9: Midnight View of Traffic Profile from Port Analysis (Mar 16-31, 2005).

Figure 9 shows a view of the midnight-hour traffic, classified into the same ten categories for the same period (March 16-31, 2005). Unknown traffic is the most dominant, followed by HTTP and Gnutella.

In summary, the results from port analysis show that the unknown traffic has increased from 10%-30% in Fall 2003 to 30%-70% in Spring 2005. The port method is obviously ineffective for classifying current Internet traffic. Other methods are needed for Internet traffic classification.

5. Application Signatures

This section discusses the application-layer signature method for P2P traffic classification. This approach can be used to establish "ground truth" for other traffic classification techniques. Separate traffic traces with full packet payloads are used for the testing and validation of the signature method.

5.1 Design Issues for Signature Method

The application-layer signature method requires access to the user data payload in the transmitted IP packets. Each P2P application has a specific signature associated with the protocol, in terms of keywords, commands, options, or other identifiable content in the packets exchanged.

The signatures can be determined based on observation of well-known P2P applications. The payload analysis scheme searches for specified strings in the payload of the packet; if found, the packet is classified accordingly.

In our work, the signature method was implemented for three widely used P2P file sharing applications: Gnutella2, KaZaA, and BitTorrent. The signature method was proposed by Sen et al. [19]. In this method, we examine available documentation and packet-level traces to identify appropriate application-level signatures, and then use these signatures to classify packets in the trace file.

There are two important design issues related to the signature technique. First, P2P traffic can flow over UDP and TCP, so one must decide whether TCP packets or UDP packets (or both) are subjected to payload analysis. Since most current P2P protocols transmit their data via TCP, we focus only on signatures found within TCP traffic. Second, P2P application-layer signatures can be applied to individual TCP packets (segments) or to fully reassembled TCP data streams.

Analysis at the TCP data stream level is more robust, in that it can detect signatures that straddle packet boundaries. Furthermore, signature matching need only be done once per connection rather than once per packet, reducing analysis overhead. Since we are performing offline analyses, TCP segments are reassembled into data streams before being analyzed. That is, we apply the signatures to TCP data streams, instead of individual TCP segments. Sen et al. [19] looked for signatures in individual TCP segments.

5.2 Signatures for Popular P2P Applications

This section discusses application signatures for three popular P2P applications: Gnutella2, KaZaA, and BitTorrent.

5.2.1 Gnutella2

The Gnutella2 protocol uses TCP to establish a highly interconnected hub network topology serving dense clusters of leaf nodes. TCP connections are established between Gnutella2 nodes when they form a link. Upon the establishment of a TCP connection between two Gnutella2 nodes, a handshaking phase must be completed to negotiate the link and exchange other necessary information.

The Gnutella handshake process consists of three header blocks. The node that initiated the connection sends an initial header block, as shown in Figure 10.

```
GNUTELLA CONNECT/0.6
Listen-IP: 1.2.3.4:6346
Remote-IP: 6.7.8.9
User-Agent: Shareaza 1.8.2.0
Accept: application/x-gnutella2
X-Ultrapeer: False
```

Figure 10: Initiator Header Block in Gnutella2.

The receiver then responds with its own header block as shown in Figure 11.

```
GNUTELLA/0.6 200 OK
Listen-IP: 6.7.8.9:6346
Remote-IP: 1.2.3.4
User-Agent: Shareaza 1.8.2.0
Content-Type: application/x-gnutella2
Accept: application/x-gnutella2
X-Ultrapeer: True
X-Ultrapeer-Needed: False
```

Figure 11: Receiver Header Block in Gnutella2.

Finally, the initiator accepts the receiver's header block, and provides any final information as shown in Figure 12.

```
GNUTELLA/0.6 200 OK
Content-Type: application/x-gnutella2
X-Ultrapeer: False
```

Figure 12: Final Handshake Information by Initiator in Gnutella2.

Two important header fields sent on all connections are "Remote-IP" and "Listen-IP". The Remote-IP header contains the IP address from which the remote host is connecting. The Listen-IP header contains the IP address and port number on which the local host is listening for inbound TCP connections. It should be listening for UDP datagrams on the same port. The format of this header is "IP:PORT", eg "1.2.3.4:6346".

The User-Agent header is used to identify the client software operating on the sending node. It is sent on the first transmission, i.e., the first and second header blocks in the three block exchange.

Based on our observations, we use the simple signature string "GNUTELLA" to identify the Gnutella2 application. The Gnutella2 protocol was not studied by Sen et al. [19].

5.2.2 KaZaA

Since KaZaA is proprietary and uses encryption, little is known about KaZaA's protocol, architecture, and signalling traffic. KaZaA's signalling traffic is always encrypted. In older versions of KaZaA, the file transfer traffic is not encrypted, but newer versions of KaZaA now use encryption for the file transfer traffic as well. Files are typically sent using HTTP-like messages. A sample request is shown in Figure 13, and a sample response in Figure 14.

```
GIVE 287496918
GET HTTP/1.1
Host: *
UserAgent: KazaaClient Jul 27 2004 21:14:16
X-Kazaa-Username: *
X-Kazaa-Network: KaZaA
X-Kazaa-IP: *
X-Kazaa-SupernodeIP: *
Connection: close
Kazaa-XferUid: *
```

Figure 13: Sample Request in KaZaA.

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 2255918-3908061/3908062
Content-Length: 1652126
Accept-Ranges: bytes
Date: Tue, 16 Aug 2005 14:57:45 GMT
Server: KazaaClient Nov  3 2002 20:29:03
Connection: close
Last-Mo dified: Fri, 10 Dec 2004 23:28:10 GMT
X-Kazaa-Username: *
X-Kazaa-Network: KaZaA
```

Figure 14: Sample Response in KaZaA.

Based on our observations, the simple signature string "X-Kazaa" was used for identifying KaZaa traffic.

5.2.3 BitTorrent

BitTorrent is a popular file-downloading protocol. The BitTorrent handshake message contains the string ".BitTorrent protocol" in the beginning of the message.

A sample BitTorrent header is shown in Figure 15. Based on our observations, the simple signature string ".BitTorrent" was used for identifying BitTorrent traffic.

```
.BitTorrent protocol..........V-
S................s.e.H.I.C.N....X.f.BitTorrent
protocolex........V-S...............exbc..LORD....
```

Figure 15: Sample Header in BitTorrent.

5.2 Validation of Signature Method

The purpose of our application-layer signature analysis is to establish "ground truth" for P2P traffic classification. The hope is that the signature method will produce traffic classification results that are consistent with (or very close to) the results determined from manual trace analysis.

Testing and validation of the signature method was done using a separate traffic trace with complete packet payloads. The signature method was tested on the validation trace only, since the 2-year trace from the University of Calgary network did not contain packet payloads.

The validation trace was collected using a local machine in the Department of Computer Science at the University of Calgary. From this workstation, a variety of Internet applications were launched. In particular, the trace traffic was generated by running three P2P applications (Gnutella2, KaZaA, and BitTorrent), while also accessing several Web sites and running an email client. The packets in the trace file were captured using Ethereal [3].

Table 1 provides summary information about the validation trace. The trace file was collected on Wednesday, August 17, 2005. The validation trace contained 25,585 packets, generated from 450 TCP flows and 513 UDP flows. The top 100 flows based on byte traffic volume were examined manually as well as using the signature method. The top 100 flows accounted for 17,462 packets (68.25% of total packets) and 12,800,611 bytes (90.52% of total bytes). Among these 100 flows, 95 were TCP flows and 5 were UDP flows.

Table 1: Description of Validation Trace

Trace Period	900 seconds
Total Packets	25,585
Total Bytes	14,141,494
Avg. packets/sec	28.4
Avg. packet size	553 bytes
Avg. Bytes/sec	15,715
Avg. Mbit/sec	0.126

The results from manual examination of the validation trace are shown in Table 2. Manual analysis identified 43 TCP flows and 2 UDP flows as P2P (denoted with '*' in the table).

Table 2: Results from Manual Analysis of Validation Trace

Application	Num TCP Flows
HTTP	47
Gnutella	36 * (plus 2 UDP) *
KaZaA	7 *
HTTPS	3
DNS	2
SSH	1
Microsoft –DS	1
Netbios	1

The signature method identified as P2P 42 of the 43 P2P TCP flows. The signature method missed classifying one Gnutella TCP flow, since no signature was found. Manual examination showed a pair of IP hosts communicating on port 6346 via TCP and UDP. It was obvious that the flow was Gnutella. The signature method did not classify any non-P2P flow as P2P. Thus, the signature method was quite accurate.

Although the signature method is quite accurate, there are obvious limitations to its usage in practice. First, privacy regulations may make it illegal to access the user payload for signature analysis. Second, signature analysis is only possible when you know in advance what you are looking for. Since new P2P protocols arise frequently, the signature analysis tool needs to be updated regularly. Third, some protocols like KaZaA and BitTorrent now use encryption, which renders payload analysis useless. These issues limit the applicability of signature analysis, and motivate the need for other methods of P2P traffic classification.

6. Transport-Layer Method

This section describes the transport-layer approach to P2P traffic classification, developed by Karagiannis et al [10].

6.1 Overview of Transport-Layer Method

Karagiannis et al. [10] proposed a novel method for identifying P2P traffic, based on transport-level connection patterns. The method does not require payload analysis. Rather, the analysis is flow-based, focusing on the connection-level patterns of P2P applications, which are a distinctive and persistent feature. While P2P applications might use random ports, or encrypt the application layer data, the connection-level patterns at the transport layer remain the same.

The transport-layer method relies on two heuristics for P2P traffic identification. These heuristics are effective in measuring aggregate P2P traffic, and can even detect new emerging P2P applications [10].

The first heuristic involves the simultaneous use of TCP and UDP by a pair of communicating hosts. If a pair of hosts is using TCP and UDP simultaneously, then most likely the traffic is P2P. UDP is prevalent in P2P systems because it provides a low-overhead method of sending queries or status messages to many peers. There are some applications like online gaming, DNS, and NFS that exhibit similar behavior, but this known traffic can be explicitly removed from consideration using a checklist for well-known ports.

The second heuristic is based on connection patterns for {IP, port} pairs. The reasoning is that if each P2P host chooses its dynamic port number at random, then it is highly unlikely for multiple P2P hosts to use the same port number. Expressed another way, for a P2P application on a given host, the number of distinct ports communicating with it will likely match the number of distinct IP addresses communicating.

During the analysis, each flow is marked either as P2P or non-P2P based on these heuristics. The classification method suggested in [10] is summarized as follows:

1. Look for source-destination IP pairs that concurrently use both TCP and UDP. If such IP pairs exist and they do not use any well-known standard ports for non-P2P applications, then consider them P2P.
2. Examine all source {srcIP, srcport} and destination {dstIP, dstport} pairs. Look for pairs for which the number of distinct connected ports matches the number of distinct connected IPs. All pairs for which this equality holds are considered P2P. If the difference between connected IPs and ports for a certain pair is large (say larger than 10), regard this pair as non-P2P.
3. Remove traffic from known applications with similar behaviour. Use packet size information to help remove false positives and false negatives.

6.2 Our Transport-Layer Method

The dataset used for our study has several limitations when codifying the techniques proposed in [10]. First, the trace does not contain UDP packets, so the effectiveness of the TCP/UDP heuristic cannot be evaluated. Second, the trace only contains TCP SYN, FIN, and RST packets, so data packet size information cannot be utilized.

Fortunately, the TCP SYN, FIN, and RST headers are sufficient to provide information regarding connection-level patterns, so that the IP-port pair heuristic can be evaluated. The design of our transport-layer method differs from the original method suggested in [10], for the reasons stated previously. The description of our approach follows.

For each day of traffic to be analyzed, do the following:

1. Concatenate in order the 24 trace files for the day.
2. Build a CompleteFlowTable (all flows) for all packets observed in a 15-minute sliding time window.
3. Scan the CompleteFlowTable and remove all flows that are using standard ports of known non-P2P applications. Set the P2P flag for flows using standard ports for known P2P applications. Call the resulting table the OtherFlowTable.
4. Create a table for all {IP, port} pairs in OtherFlowTable.
5. Scan the trace file and count the number of distinct destination IP hosts and the number of distinct destination ports for each {IP, port} pair. If the number of distinct ports and IP hosts match, then flag the {IP, port} pair as P2P. Call such pairs P2Ppairs.
6. Flag all the flows in OtherFlowTable corresponding to P2Ppairs as P2P. Create a new table containing all of the flows flagged as P2P in OtherFlowTable. Call this table the P2PFlowTable.
7. Calculate the percentage of P2P flows for the given time window from the number of flows in P2PFlowTable and CompleteFlowTable.

6.3 Validation of Transport-Layer Method

The transport-layer method was tested using the same validation trace described in Section 5.3. The purpose is to show that the transport-layer method is as effective as (or almost as effective as) the application-layer signature method. Recall that manual analysis of the validation trace identified 43 TCP flows and 2 UDP flows (among the top 100 flows) as P2P. The transport-layer method processed all flows in the validation trace, though the results presented here are just for the top 100 flows, for consistency.

On the validation trace, our transport-layer method identified all 45 P2P flows. There were no false positives and no false negatives, indicating that our method is reliable.

For completeness, we also evaluated separately the effectiveness of the IP-port pair heuristic and the TCP/UDP heuristic, using manual analysis of the top 100 flows in the validation trace. On its own, the IP-port pair heuristic (without knowledge of ports) identified 49 TCP flows and 2 UDP flows as P2P. It identified as P2P 6 TCP flows that were not P2P. This represents a false positive percentage of 13%. The misidentified flows were 3 HTTP, 1 SSH, 1 Netbios, and 1 MS-DS (Microsoft Denial of Service). On its own, the TCP/UDP heuristic identified only 6 flows as P2P. It missed 39 flows, for a false negative percentage of 87%. Manual analysis showed that the IP host pairs participating in the missed flows did not use UDP for any data transfers.

These results show the importance of considering multiple aspects of the traffic characteristics, including known ports. The results provide confidence in the effectiveness of our modified transport-layer method for P2P traffic classification.

6.4 Results for Transport-Layer Method

Figure 16 shows the results from our transport-layer analysis of the University of Calgary dataset. Each point represents the daily average percentage of P2P flows in the aggregate traffic.

Figure 16 shows that the proportion of P2P traffic flows ranged from 10-30% between September 2003 and January 2004. Between July 2004 and July 2005, the aggregate P2P traffic has ranged from 30-70%. On average, this traffic accounts for 38% of the byte traffic volume seen on the network.

The results in Figure 16 show a significant increase in P2P traffic from 2003-2004 to 2004-2005. This trend is structurally similar to that observed for the "unknown" traffic for port classification in Figure 5. That is, the proportion of P2P flows identified by transport-layer analysis seems to increase in a manner similar to that for the proportion of unknown traffic in the port-based analysis.

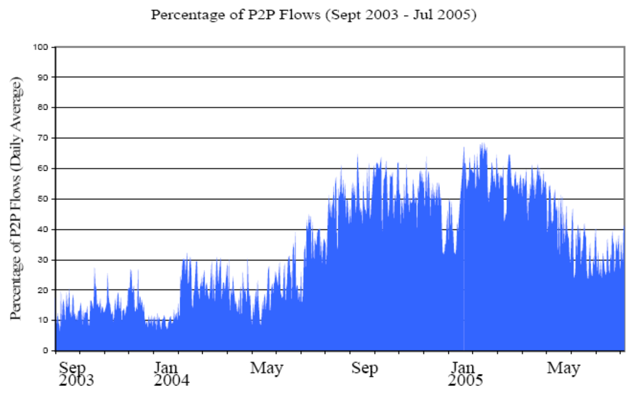
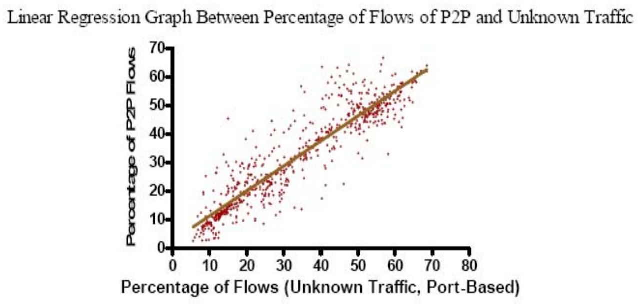


Figure 16: P2P Flows for Sept 2003-Jul 2005 (Daily Average).

To explore this trend further, and quantify the relationship, we use regression analysis. Figure 17 uses a scatter plot to show the apparent linear relationship between the percentage of unknown flows from port-based analysis (x axis) and the percentage of P2P flows from transport-layer analysis (y axis). There is one data point for each day of trace data analyzed.



Best-fit values	
Slope	0.8739 ± 0.01374
Y-intercept when X=0.0	2.554 ± 0.5258
X-intercept when Y=0.0	-2.922
1/slope	1.144
95% Confidence Intervals	
Slope	0.8470 to 0.9008
Y-intercept when X=0.0	1.523 to 3.584
X-intercept when Y=0.0	-4.218 to -1.696
Goodness of Fit	
r²	0.8536
Sy.x	6.360
Is slope significantly non-zero?	
F	4046
DFn, DFd	1.000, 694.0
P value	< 0.0001
Deviation from zero?	Significant
Data	
Number of X values	696
Maximum number of Y replicates	1
Total number of values	696
Number of missing values	0

Figure 17: Comparison of P2P Traffic and Unknown Traffic.

The relationship observed in Figure 17 is strongly linear, as indicated by the least-squares regression line in the graph. The R-squared value for the goodness of fit is 0.8536. We interpret this result as strong statistical evidence that much of the unknown traffic in our trace is P2P traffic.

Summarizing our results, a visual comparison of Figure 5 and Figure 16 provides strong circumstantial evidence that the unknown traffic is P2P traffic. However, no definitive proof is possible, since we do not have packet payloads in our campus-level trace.

Our results, along with the validation tests, suggest that the transport-layer method can reliably estimate the aggregate P2P traffic. The validation results in Section 6.3 show that the transport-layer method (applied to packet headers) produces results consistent with the signature method (applied to payloads). Furthermore, the validation results in Section 5.3 show that the signature method produces results consistent with "ground truth" from manual analysis. This reasoning and the statistical data in Figure 17 suggest that much of the "unknown" traffic in our datasets is P2P traffic.

6.4 Limitations

Although the transport-layer method looks promising, there are several limitations and caveats to bear in mind.

One limitation is that port masquerading cannot be detected. The transport-layer method uses a checklist of standard ports for filtering known traffic. If P2P traffic occupies a port that is assigned to a different application (e.g., SMTP on port 25), then those flows cannot be classified as P2P.

A second limitation is that the IP-port heuristic is ineffective in the trivial case of one IP host communicating with another IP host on a single port. Many network applications, P2P and non-P2P, share this connection level pattern. Other heuristics may be needed to remove false negatives.

Finally, P2P applications continue to evolve. While the transport-layer method proposed in [10] has been effective in the recent past, and might be effective currently, there is no guarantee of its effectiveness on the next generation of P2P applications. Other traffic classification techniques may be required. For example, early work by Erman et al. [4] on cluster-based analysis looks very promising.

7. Summary and Conclusions

This paper studies traffic classification of Peer-to-Peer (P2P) applications. Accurate knowledge of P2P traffic is desirable for several reasons, including traffic engineering and network capacity planning. P2P applications on the Internet have evolved rapidly, making identification of P2P traffic challenging.

The paper compared three methods to classify P2P applications: port-based analysis, application-layer signatures, and transport-layer heuristics. The study used empirical network traces from the University of Calgary Internet connection for a two-year period. To the best of our knowledge, this is the first longitudinal study of the effectiveness of P2P traffic classification techniques.

Our results show that classic port-based analysis is ineffective, and has been so for quite some time. The proportion of "unknown" traffic increased from 10-30% in

2003 to 30-70% in 2004-2005. This result provides motivation for other methods to classify P2P traffic.

While application-layer signatures are accurate, this technique requires examination of user payload, which may not always be possible. Furthermore, encryption may soon render application-layer signature methods ineffective.

Transport-layer heuristics offer a novel method that classifies the P2P traffic based on connection-level patterns. Our results show that the transport-layer method can give useful information regarding aggregate P2P traffic.

As P2P applications keep evolving, new challenges will arise for P2P traffic classification. Since traditional methods like port-based analysis are now obsolete, and application-layer signatures are not always possible, transport-layer analysis is perhaps one of the best-available current approaches for P2P traffic classification. Even better methods will likely be required in the near future.

8. Acknowledgements

Financial support for this research was provided by TRLabs (Telecommunications Research Labs) in Calgary, iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, as well as NSERC (Natural Sciences and Engineering Research Council) and CFI (Canada Foundation for Innovation) in Canada. The authors thank Martin Arlitt for his assistance with trace collection and storage, and Gwen Houtzager for assistance with trace processing in the ELISA lab. The authors are very grateful to UCIT (University of Calgary Information Technologies) for providing access to the campus network traffic data. The authors thank the anonymous MASCOTS 2006 reviewers for their constructive comments on an earlier version of this paper.

9. References

[1] E. Adar and B. Huberman, "Free Riding on Gnutella," Technical Report, Xerox PARC, August 2000.

[2] BitTorrent, http://www.bittorrent.com, 2005.

[3] Ethereal, http://www.ethereal.com, 2005.

[4] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classification Using Clustering Algorithms," *Proceedings of ACM SIGCOMM Minenet Workshop*, Pisa, Italy, September 2006.

[5] Gnutella, http://www.gnutella.com, 2005.

[6] Gnutella2, http://www.gnutella2.com, 2005.

[7] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File Sharing Workload," *Proceedings of the 10th ACM Symposium on Operating Systems Principles (SOSP-10) 2003*, pp. 314-310, NY, USA, October 2003.

[8] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated Construction of Application Signatures," *ACM SIGCOMM Workshop on Mining Network Data (MineNet 2005)*, pp. 107-202, Philadelphia, PA, USA, August, 2005.

[9] Internet Assigned Numbers Authority, TCP/UDP Port Numbers, http://www.iana.org/assignments/port-numbers, 2005.

[10] T. Karagiannis, A. Broido, M. Faloutsos, and K. Klaffy, "Transport Layer Identification of P2P Traffic," *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC 2004)*, pp. 121-134, Italy, October 2004.

[11] KaZaA, http://www.kazaa.com, 2005.

[12] B. McWilliams, *'SQLsnake' Worm Blamed for Spike in Port 1433 Scans*, Newsbytes, May 21, 2002, http://online.securityfocus.com/news/429, 2005.

[13] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer to Peer Computing," Technical Report, HP Labs, Palo Alto, March 2002.

[14] Napster, http://www.napster.com, 2005.

[15] A. Oram, *Peer-To-Peer: Harnessing the Power of Disruptive Technology*, First Edition, O'Reilly, March 2001.

[16] V. Paxson, "Growth Trends in Wide-Area TCP Connections," *IEEE Network*, Vol. 8, No. 4, pp. 8-17, July 1994.

[17] S. Saroiu, K. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proceedings of the Multimedia Computing and Networking (MMCN 2002)*, pp. 18-25, California, January 2002.

[18] S. Saroiu, K. Gummadi, and S. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Springer-Verlag Multimedia Systems*, Vol. 9, No. 2, pp. 170-184, 2003.

[19] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic using Application Signatures," *Proceedings of the 13th International World Wide Web Conference*, pp. 512-521, NY, USA, May 2004.

[20] tcpdump, http://www.tcpdump.org, 2005.

[21] K. Thompson, G. Miller, and R. Wilder, "Wide-area Internet Traffic Patterns and Characteristics," *IEEE Network*, Vol. 11, No. 6, pp. 10-23, November/December 1997.