Wirespeed, Privacy-Preserving P2P Traffic Detection on Commodity Switches

Hassan Khan, Syed Ali Khayam
xFlow Research Inc.
1392 Borregas Ave.,
Sunnyvale, CA 94089 USA
{hassankhan, ali}@xflowresearch.com

Leana Golubchik
Computer Science Dept.
Univ. of Southern California
Los Angeles, CA, USA
leana@usc.edu

M. Rajarajan
School of Engg. & Math. Sci.
City Univ. London, UK
London, UK
r.muttukrishnan@city.ac.uk

Michael Orr
Marvell Technology
5488 Marvell Lane
Santa Clara, CA, USA
orr@marvell.com

***Abstract*—Application layer Deep Packet Inspection (DPI) is presently the predominant method to detect and regulate p2p traffic. Even if one chooses to ignore the ethical debate on DPI and privacy, most of the existing solutions rely on traffic and payload signatures/features which are not robust against NATing, client/protocol/port changes, encryption, and obfuscation. Moreover, the processing intensive nature of these solutions requires specialized hardware to provide detection at wirespeed and consequently increases the installation and management cost of enterprise and carrier networks.**

In this paper, we propose a behavioral p2p traffic classification method which uses *only* network layer features and still provides better accuracy and speed than existing non-proprietary techniques. We evaluate existing and propose novel discriminating network layer traffic features using information divergence of p2p and non-p2p traffic. These features are leveraged in low-complexity cross-correlation and log-likelihood frameworks to classify p2p traffic in real-time. For empirical evaluations, we compare the proposed method with four prominent techniques using encrypted p2p data on our network and 3.4 million unencrypted flows from publicly-available backbone data. We then design an application to demonstrate that the proposed approach can be used to detect and regulate p2p traffic at wirespeed on OpenFlow compliant commodity hardware. In addition to wirespeed detection, the proposed method provides significant improvements of up to 25% in detection rate, 30% in false positive rate, and 240 seconds in detection delay.

I. Introduction

High market penetration of broadband connectivity in the past few years has catalyzed a fundamental change in users' traffic characteristics with peer-to-peer (p2p) file sharing content comprising 40-70% of today's Internet's traffic [1]. While p2p traffic volume is decreasing [2], p2p content[1] still represents a major traffic share. Therefore, network operators and service providers would like to detect, classify, regulate and charge for this content. Enterprise networks are also inclined to monitor, and at times block, such content to reduce the risk of information leakage. Universities are concerned about file sharing software that are bandwidth hungry and are mostly used to illegally distribute copyrighted content. Legislative, law enforcement, and intelligence agencies are interested in classifying and regulating p2p content to implement governmental traffic policies and to identify points of presence of suspicious network activities.

Over the last few years, Deep Packet Inspection (DPI) based solutions that analyze each packet up to the application layer payload have matured to provide commercial-grade performance in detecting and regulating p2p traffic at wirespeeds [3]. Simultaneous to the widespread introduction of DPI products [4]–[9], a global debate has ensued on the ethical, legal and privacy aspects of DPI. In particular, DPI's privacy implications have sparked government-level debates and deliberations in North America and the EU [12]–[15]. Interpretations of USA's Federal Wiretap Act of 1968 [35] and UK's recently enacted Digital Economy Act [36] have added further fuel to this controversy.

Even if this debate is ignored in favor of better traffic visibility and management, most of the existing DPI detectors rely on traffic/payload signatures that are not robust against client and protocol changes [16]–[18]. Furthermore, in order to provide wirespeed detection, these solutions require specialized boxes (comprising of high-end processors and Regex engines) to be integrated with the existing networking infrastructure. While universities and medium-scale businesses cannot afford these solutions, enterprise and carrier networks bear increased installation and management cost.

A new school of thought is now emerging which advocates detecting p2p traffic using behavioral features at application and/or transport layers [20]–[25],[28], [29]. Some of these solutions are also defeated in practical network configurations (such as NATing and tunneling) and under payload cloaking (using encryption and/or obfuscation). P2p software are already exploiting these weaknesses to evade detection with many popular clients now supporting dynamic random port number assignment, chunked file transfers, connection reversal, HTTP masquerading, and payload encryption [10], [11].

In view of the above problems with p2p traffic classification at application and transport layers, we argue that a practical p2p traffic detector should rely *only* on network layer features. The questions looming over payload-oblivious, behavioral classification of p2p traffic at the network layer are: (1) Can it meet the accuracy (detection and false positive rates) requirements? (2) Can it meet the detection delay requirements

[1]Throughout this paper, the term 'p2p traffic/hosts' is used for file sharing hosts. We do not attempt to analyze/detect real-time media traffic (e.g., Skype).

expected from a real-time p2p detector? (3) Can it operate at wirespeed in the enterprise and carrier networks (at 10s of gigabits per second)?, (4) Can it be incorporated in the existing networking infrastructure with reasonable installation, management and maintenance cost?

To address the above questions, in this paper we propose and evaluate a network layer p2p traffic detector. We first use information-theoretic tools to evaluate a rich set of existing network layer features which have been used for traffic classification by prior studies [37]. We note that most of these existing features do not vary significantly across p2p and non-p2p traffic. We therefore extend the existing feature set by proposing novel features which can characterize p2p traffic more accurately. Existing and novel features are leveraged in low-complexity cross-correlation and log-likelihood frameworks to classify p2p traffic in real-time.

For empirical evaluation of the proposed method, we implement the proposed detector on SDN/OpenFlow compliant commodity hardware. We then use an encrypted and NATed datset from our network and 3.4 million flows from a publicly-available, non-encrypted backbone dataset. ROC curves are used to evaluate the accuracy of our method with four prominent, non-proprietary techniques using these two datasets. Our performance evaluation for host classification shows that the proposed method renders an improvement of 15% in detection rate, 10% in FP rate, and 9.5 minutes over the second-best technique. Similarly, for the detection of p2p connections, improvements of 25% in detection rate, 30% in FP rate, and 4 minutes in detected delay are observed over existing methods. Evaluation on NATed traffic shows that the proposed p2p connection classification approach's accuracy is unaffected by NATing. Finally, we publicly release the implementation of proposed approach and our encrypted dataset to facilitate future research.

Main technical contributions of this paper are as follows:

- Modeling and evaluation of a comprehensive set of distinguishing *network layer* features that differ considerably among p2p and non-p2p traffic classes;
- Quantification of the divergence among these distinguishing features using information divergence measures;
- A fast and efficient classifier that uses 6 network layer features to detect p2p hosts with a detection rate of 97.84%, a false positive rate of 4.01%, and a detection delay of 30 seconds on our evaluation datasets; and
- A fast and efficient classifier that uses 12 network layer features to detect p2p connections with a true positive rate of 93.27% and a false positive rate of 5.57% using only 8 minutes of traffic in our evaluation datasets;
- A publicly released implementation of proposed approach as a NOX application that allows wirespeed detection of p2p traffic on OpenFlow compliant commodity hardware;
- Collection of an encrypted p2p traffic dataset that has been released publicly to facilitate future research.

Other important contributions of this paper are philosophical in nature:

- We dispel the common belief that network layer features alone are insufficient to provide the accuracy that is expected from p2p traffic detectors. Our proposed network layer detector is accurate, fast, and robust against evasion, and easily outperforms existing application and transport layer detectors on all of these evaluation metrics.
- We show that privacy-preserving, payload-oblivious p2p traffic classification is indeed possible. In fact, such detectors offer a pragmatic middle ground for the two extreme schools of thought on DPI-based traffic classification.

II. Background and Dataset Description

In this section we provide the necessary background of the Software Defined Networks (SDNs) and a detailed description of dataset that was used for evaluation purposes. To maintain a logical flow of thought, we discuss our SDN based implementation of the proposed approach after empirical evaluation of the proposed approach.

A. SDNs/OpenFlow

Software Defined Networking (SDN) has been proposed as a new network architecture that separates the data plane (fast packet forwarding) and the control plane (high level routing decisions). The data plane portion continues to reside on the switch, while control plane is moved to a separate controller, typically a standard server. Communication between the dataplane and the control plane is performed using a well defined protocol. This segregation of data and control planes enables innovations by researchers, operators, application/service providers, and network equipment vendors. While originally designed to allow network slicing and virtualization to support experimentation at scale on a production network by researchers, SDNs are increasingly being deployed for research and experimentation at several university campuses and within I2 and NLR backbones [38].

OpenFlow is currently the defacto protocol that provides well defined APIs for a protocol to communicate between the data plane and the control plane. When an OpenFlow compliant switch receives a packet it has never seen before, for which it has no matching flow entries, it sends this packet to the controller. The controller then makes a decision on how to handle this packet. It can drop the packet, or it can add a flow entry directing the switch on how to forward similar packets in the future. We now provide a brief description of some OpenFlow based technologies that were used in this work:

1) Open vSwitch [40]: Open vSwitch is a production quality open source software switch designed to be used as a vswitch in virtualized server environments. Open vSwitch is open to programmatic extension and control using OpenFlow protocol. Open vSwitch was designed to be compatible with the modern switching chipsets and it has been ported to the Marvell Technology's modern switching chipset [44]. The Marvell Technology's port of Open vSwitch is a fully OpenFlow 1.0 specification compliant switch and it communicates with an OpenFlow compliant controller to segregate the data plane and the control plane.

TABLE I
DURATION OF DATA COLLECTION INTERVALS

From	To	NAT
Oct 20, 2008 06:46	Oct 25, 2008 05:22	No
Oct 29, 2008 06:29	Nov 09, 2008 17:42	No
Nov 10, 2008 08:36	Nov 21, 2008 01:17	No
Jan 24, 2009 13:42	Jan 26, 2009 17:52	Yes
Jan 27, 2009 15:51	Jan 28, 2009 12:11	Yes

2) NOX [41]: NOX is a commercial grade OpenFlow controller which was designed to support networks of hundreds of switches. NOX aims to simplify the creation of software for controlling or monitoring networks by providing the underlying framework to develop controller applications. Programs written within NOX (also called NOX apps) have flow-level control of the network including forwarding, routing, user and host level access control etc.

B. Dataset Description

For this study, we use publicly available 100 Mbps Ethernet backbone trace data. Since all prominent p2p clients and protocols now support encryption to evade DPI-based p2p regulators, in addition to using a publicly-available dataset, we independently collected a dataset with labeled encrypted/obfuscated p2p traffic. We have made our dataset publicly available for repeatable performance benchmarking by future studies.[2]. In this section, we provide details of the datasets used in this paper.

C. Unencrypted p2p Traffic Dataset

For unbiased performance evaluation, we use publicly available WIDE trace data [24]. WIDE trace data was captured at a 100 Mbps Ethernet US-Japan Trans-Pacific backbone link that carries commodity traffic for WIDE member organizations. The WIDE trace was captured from 22:45 March 03, 2006 to 23:40 March 03, 2006. In 55 minutes of packet capturing, 32 million packets totaling over 14 Gigabytes were recorded. Due to privacy concerns, only the first 40-bytes of payload for each packet are available. The trace contains about 3.4 million flows. The WIDE trace is not labeled. To establish ground truth for WIDE trace, we use Karagiannis et al.'s payload classifier [18] and OpenDPI [19]. We note that these payload based classifiers might not be able to identify encrypted traffic; therefore, we only use them to identify and label unencrypted p2p traffic and non-p2p traffic (we discard the unknown traffic). Details of this payload classifier are provided in Section III-A.

D. Encrypted and NATed p2p Traffic Dataset

We collect traffic traces at the edge router of our university's network. The traffic is generated and consumed by teaching blocks, research labs and administration blocks. The traffic capturing was carried out in five intervals as shown in Table I. For the collection intervals 4 and 5, traffic was collected after a Network Address Translation (NAT) server. The goal was to observe the behavior of classifiers when transport and network layer features are masked by a NAT server. The total traffic captured in this dataset was over 170 GB.

[2]http://wisnet.seecs.nust.edu.pk/projects/ENS/DataSets.html

TABLE II
P2P APPLICATIONS' TRAFFIC STATISTICS

Client	Sessions Estb.	Traffic Vol.	Encryption
Vuze 4.0	20	685 MB	RC4
Flashget 1.9.6	62	60.7 MB	Protocol Encryption (Algorithm Unknown)
μTorrent 1.8.1	30	1.08 GB	Forced Encryption (Algorithm Unknown)
BitTorrent 6.1.2	40	1.59 GB	Forced Encryption (Algorithm Unknown)
Deluge 1.0.7	30	171 MB	Forced Entire Stream Encryption (Algorithm Unknown)
BitComet 1.07	20	57.4 MB	Forced Encryption (Algorithm Unknown)
alite 0.3.1	9	413 MB	RC4
eMule v0.49b	203	2.67 GB	Forced Encryption (Algorithm Unknown)

In our dataset, the p2p traffic belongs to the BitTorrent, eDonkey and Kademlia protocols. These protocols were chosen as a representative set because these protocols generate the largest volume of p2p traffic on the Internet [1]. During our trace collections for the BitTorrent protocol, we used multiple torrent files for transferring data from/to multiple geographical locations for each torrent session. Multiple torrent clients were used to introduce the real-world diversity in the dataset as different clients have different behavior. For instance, BitTorrent, μTorrent, Vuze, Halite, and Deluge used TCP for communication with tracker, while BitComet and Flashget used UDP. Other protocol fields and packet sizes also varied from client to client. Encryption was enabled for all the torrent sessions. For eMule sessions, option related to protocol obfuscation ("Allow obfuscated connections only") was enabled in the client to ensure logging of encrypted traffic only. Statistics for the p2p file sharing applications' traffic are given in Table II.

The non-p2p traffic contains other traffic classes including HTTP, FTP, streaming video, instant messenger, and Skype. The non-p2p trace data also includes encrypted traffic like https, ssh, gtalk, etc. The average throughput of the non-p2p trace data is 0.6 Mbps. Majority of the non-p2p data uses TCP at the transport layer.

Equipped with these datasets, in the following section we discuss the related work and evaluate the existing signature-based approaches on our encrypted dataset.

III. RELATED WORK AND EVALUATION OF SIGNATURE-BASED APPROACHES

In this section we discuss the related application, transport and network layer approaches for p2p traffic detection. We also evaluate existing application layer/signature-based approaches on unencrypted and encrypted datasets to show that p2p traffic can use encryption to evade detection.

A. Related Work

Various application [4]–[9],[16]–[18], transport [20]–[25],[28], [29] and network layer [26], [30] approaches have been proposed to classify p2p traffic. To maintain a logical flow of thought, we defer discussion on signature based approaches to subsequent section, and we only discuss relevant transport and network layer based approaches in this section.

In a seminal work, Karagiannis et al. [20] perform non-payload based classification of p2p traffic using transport protocol (TCP and UDP) and connection patterns of <IP, Port> pairs. Karagiannis et al. extend their work in [21] and use a multilevel approach to design BLINC which uses social, functional, and application level heuristics to classify 80%-90% of p2p traffic with more than 95% accuracy. Collins et al. [23] distinguish between p2p and non-p2p flows using: 1) packet size; 2) amount of data exchanged between hosts; and 3) rate of failed connections. Constantinou and Mavrommatis [22] classify p2p traffic based on the number of peers in a connected group and connection direction. Bartlett et al. [28] detect p2p traffic based on peer coordination and failed connections, bi-directional connections and use of unprivileged ports. In [27], authors use six flow- and protocol-level features to identify p2p traffic.

While numerous techniques have been explored to detect p2p traffic at transport layer, network layer detection of p2p traffic has been largely unexplored. Ngiwlay et al. [26] propose to use connected IPs, active transfers, bi-directional active transfers, and IP-relation changes to detect 90% of torrent peers within 10 minutes. Raahemi et al. [30] propose to use protocol, TTL value, IP protocol, IP address and packet length for p2p connection detection. It should be noted that both of these techniques have been designed for and evaluated on BitTorrent based p2p traffic.

We now discuss and evaluate contemporary signature-based approaches for p2p traffic detection. We compare the contemporary transport and network layer approaches with the proposed approach in Section V.

B. Signature-based Approaches and their Evaluation

Signature-based DPI traffic classification is a well-known and conventional approach to classify p2p traffic. The main complication with this approach is that it requires a priori knowledge of signatures, protocol interactions and packet formats. However, many of the p2p protocols do not make their documentation publicly available and most of the applications which implement the protocols do not follow the standard specifications. Nevertheless, reasonably robust signatures have been identified by existing studies [16]–[19].

Karagiannis et al. [18] showed the complications in signature-based traffic classification on an OC-48 link. Sen et al. [16] developed signatures for a number of p2p protocols using pattern matching. In [17], heuristics to detect unknown applications are used in addition to application signatures to improve classification. Kim et al. [24] added new signatures to Karagiannis's payload classifier. This signature based detector was also used in an evaluation study [24] to establish the groundtruth for dataset. We evaluate OpenDPI[3] and Karagiannis et al.'s payload classifier with updated signatures on the encrypted traces that we collected to determine the sensitivity of signature-based schemes to encryption and obfuscation in p2p traffic. Before presenting the evaluation results, we provide a brief description of these approaches.

TABLE III
EVALUATION OF SIGNATURE-BASED APPROACHES (KPC = KARAGIANNIS' PAYLOAD CLASSIFIER [18]; !P2P = UNKNOWN TRAFFIC)

	Unencrypted Trace		**Encrypted Trace**			
	OpenDPI	**KPC**	**OpenDPI**		**KPC**	
	P2P	P2P	P2P	!P2P	P2P	!P2P
eMule	100%	100%	0%	100%	39.5%	60.5%
μTorrent	100%	100%	0%	100%	0%	100%
BitTorrent	100%	100%	3.8%	96.2%	5%	95%
Others	100%	100%	0%	100%	64.7%	35.3%

OpenDPI can efficiently detect over 90 protocols/ applications [19]. The signature-based technique of Karagiannis et al. [18] is able to detect 38 popular p2p applications and 21 non-p2p protocols. In addition to signatures, [18] used three heuristics to address the limitations of the available traces (e.g., some of the traces in [24] only had 16 bytes of payload data). Due to these limitations, the detector could not accurately tell whether "HTTP/1.1 503 Ser, HTTP/1.0 503 Ser and HTTP/1.1 206 Par" were p2p traffic or Web traffic. To resolve this, it was assumed that the traffic belongs to the p2p class whenever the flow is from/to source port, destination port is greater than 1000, and both ports are neither 8000 nor 8080. Similarly, if the source IP or the destination IP matched a suspected p2p source/destination host and the source and destination ports were greater than 500, the traffic was flagged as p2p.

The evaluation results of signature-based approaches on encrypted and non-encrypted p2p traffic are shown in Table III. These results show that while signature based approaches are able to detect unencrypted traffic with 100% accuracy, they are unable to classify majority of the encrypted traffic. OpenDPI is unable to detect any p2p traffic, except a 3.8% of traffic from the BitTorrent client. After analyzing the traces for those identified connections, we found that all of the identified connections belonged to the NATed trace data and none of the non-NATed BitTorrent connections were identified by OpenDPI. Karagiannis' Payload Classifier is able to detect most of the p2p traffic using its heuristics, which is why it is unable to identify the applications that generated p2p traffic (except for 5% of NATed BitTorrent connections which were detected using signature matching).

The low classification rates of signature-based approaches show that these classifiers cannot be used if payload information is cloaked using encryption. To address this shortcoming, in the following section we identify discriminating network layer features that can be used for p2p traffic detection.

[3]Open source version of ipoque's industry leading DPI engine [3]

IV. Identification of Distinguishing Features

Preliminary empirical results of the last section reveal the inability of signature based approaches to detect encrypted p2p traffic. We now proceed to design an approach which is resilient to the obfuscation techniques employed by p2p applications. To this end, we subdivide the present problem of p2p hosts and traffic classification into two sub-problems: *identification of discriminating network layer traffic* and *classification using the identified features*. This section focuses on the first problem where we compare traffic generated by p2p hosts/connections and non-p2p hosts/connections to identify distinguishing or divergent features. We note that p2p host detection is a simplified version of the p2p traffic detection, however, there are scenarios where only p2p hosts need to be identified for example, network operators and service providers might want to charge p2p hosts more. Therefore, we treat p2p host detection as a separate problem in order to design a pragmatic solution to it. In this context, we first evaluate a rich set of existing features that have been proposed for traffic classification by prior studies. Subsequently, we propose a new set of discriminant features that can be used to enhance the accuracy of p2p traffic classification.

A. Feature Identification Measure

P2p traffic is fundamentally different from non-p2p traffic because of its distributed design, heterogeneity, connection establishment using peer selection algorithms, and connection shuffling [34]. Hence, it can be intuitively deduced that some features of p2p and non-p2p traffic should diverge significantly. However, judicious selection of the best features that facilitate detection requires a mathematical measure that can quantify the divergence of a feature across p2p and non-p2p traffic. To this end, we model traffic features as discrete random variables by mapping each possible value of a feature to an integer outcome. In case of continuous valued features, we divide the feature space into equal-sized bins and assign an integer value to each bin. For each feature random variable, we compute two histograms using the p2p and non-p2p traces, respectively. By normalizing these histograms, we obtain the Probability Mass Functions (PMFs) of feature random variables.

To quantify the difference between the PMFs derived from p2p and non-p2p traffic, we first employed the Kullback-Leibler (K-L) information divergence measure [31]. For two PMFs p and q of a discrete random variable X, K-L quantifies the difference between the two PMFs as [31]:

$$D(p\|q) = \sum_{i\in\Lambda} p(i)\log_2\frac{p(i)}{q(i)}, \tag{1}$$

where Λ is the image of X, and $p(i)$ and $q(i)$ respectively represent the probability of feature value i in p and q.

While the K-L measure provided satisfactory divergence quantification for most traffic features, we encountered three limitations: a) non-symmetry, $D(p\|q) \neq D(q\|p)$, b) lack of normalization, and c) $D(p\|q) = \infty$, if $p(i) \neq 0, q(i) = 0$ for any $i \in \Lambda$. In view of these limitations, we use a mutual information based measure. Mutual information between two discrete random variables X and Y is defined as:

$$I(X,Y) = \sum_{(i,j)\in\Lambda^2} f(i,j)\log_2\frac{f(i,j)}{p(i)q(j)}, \tag{2}$$

where Λ^2 represents a two dimensional image and $f(i,j)$ represents the joint distribution of a traffic feature in p2p and non-p2p traffic; this joint distribution was computed by pairing up a p2p connection with its next non-p2p connection.

Mutual Information is a measure of the amount of overlap between two feature distributions. It is a symmetric measure and does not require the two distributions to be continuous with respect to each other. Moreover, $I(X,Y) = H(X)$ in the limiting case of $X = Y$, while $I(X,Y) = 0$ when X and Y are independent. For the present problem, we employ the following normalized function of the mutual information measure, known as Variation of Information (VoI):

$$V(X,Y) = 1 - \frac{I(X,Y)}{H(X,Y)}, \tag{3}$$

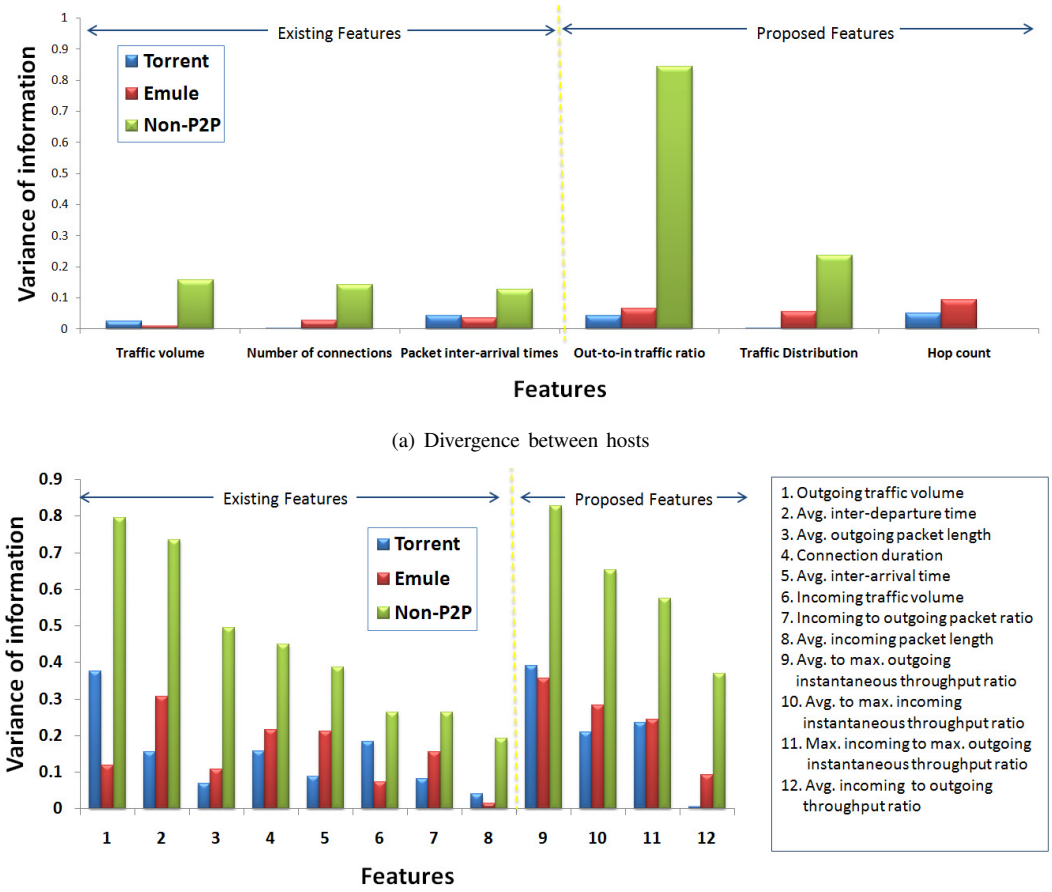
where $H(X,Y) = -\sum_{i,j\in\Lambda^2} f(i,j)\log_2(f(i,j))$ is the joint entropy of X and Y.

We now use the proposed VoI measure to quantify the contribution of existing traffic features in improving the accuracy of a p2p traffic classifier.

B. Evaluation of Existing Features

Prior studies have identified a rich set of approximately 42 distinct network layer traffic features for p2p traffic classification; see [37] and references therein. As a first step, we evaluated the individual contributions of each of these existing features for p2p traffic classification. As can be intuitively deduced, a good classification feature should vary considerably between p2p and non-p2p connections, while staying reasonably stable within a given (p2p or non-p2p) traffic class.

Fig. 1(a) shows three existing features which provided the highest VoI divergence between p2p hosts and non-p2p hosts. Note that all three features are related to packet- or connection-level traffic volumes. Also, note that the divergence is not very high, mainly because other types of traffic– in particular, video traffic –was also providing high volumes with a number of connections to the server. Fig. 1(a) shows the divergence between p2p connections and non-p2p connections for eight existing features that provided enough VoI divergence that they can be considered for p2p connection classification. This classification is invoked after a host has been classified as a p2p host and we want to detect and regulate its p2p connections, without disturbing the non-p2p connections. It can be observed that existing connection-level features, although still relying mainly on traffic volume, can provide very high divergence between p2p and non-p2p connections. Hence, if a host is classified accurately, existing connection-level features can provide good estimation of the p2p connections originating from or terminating at the host.



(a) Divergence between hosts

(b) Divergence between connections

Fig. 1. Variation of information (VoI) in distinguishing host- and connection-level features.

C. New Features for P2P Traffic Detection

We complement existing host- and connection-level features with a new set of robust traffic features. In this section, we describe these features and provide intuitive and empirical evidence to support their selection.

1) Outgoing-to-incoming Traffic Ratio: P2p applications simultaneously operate in both client and server modes. Therefore, while downloading a file, they are also serving/uploading files to the peers. This results in higher outgoing-to-incoming traffic ratio for p2p hosts. This behavior is uncharacteristic for non-p2p hosts which, due to the web's information pull model, typically exhibit a high degree of disparity between incoming and outgoing traffic rates, with incoming rates being significantly higher than outgoing rates. In our evaluation dataset, we observed that on average p2p hosts had five times higher outgoing to incoming traffic ratio than non-p2p hosts. This result is also supported by the divergence values in Fig. 1(a) which show that this feature's information divergence is approximately 5 times more than the best existing feature [Traffic volume]. Thus outgoing-to-incoming traffic ratio offers a very promising feature to improve the accuracy of p2p host classification.

2) Traffic Distribution: In a typical p2p application, only a fraction of connections have active file transfers at varying rates with peak transfer rates close to the mean due to peer clustering. In case of non-p2p applications, either applications share their bandwidth fairly (which results in a flat distribution) or some applications consistently consume more bandwidth than the others (with huge variance among connections). This observation is shown pictorially in Fig. 2 for 15 connections on two p2p and two non-p2p hosts. Clearly, non-p2p traffic's histogram is noticeably flatter than the p2p traffic histogram. Therefore, the traffic distribution feature should be able to distinguish p2p and non-p2p hosts. This is the reason that the traffic distribution feature provides almost twice the amount of divergence than was observed by the best existing host-level feature.

3) Geographical Diversity: Due to the widespread usage of common Content Distribution Networks (CDNs) by many web services (e.g., webservers, search engines, streaming video servers, FTP servers, etc.), we observed that non-p2p hosts generally use network services that are situated relatively close to their ISP's core network. On the other hand, p2p applications connect to peers which are placed at geographically distributed locations. As a consequence, inter-connection TTL

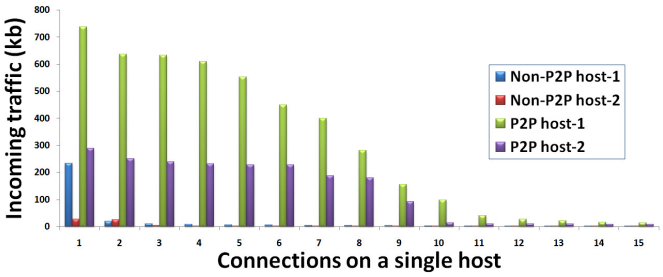


Fig. 2. Incoming traffic distributions of p2p non-p2p hosts.

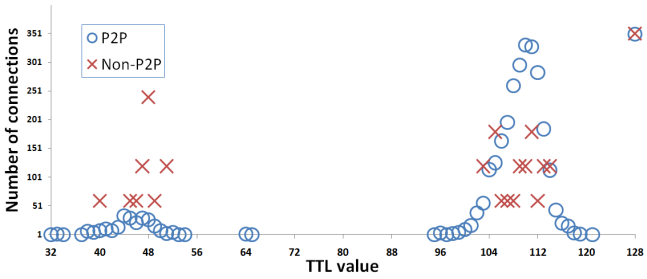


Fig. 3. Distribution of IP TTL values for p2p and non-p2p hosts; large number of connections having TTL value 128 have been truncated for clarity.

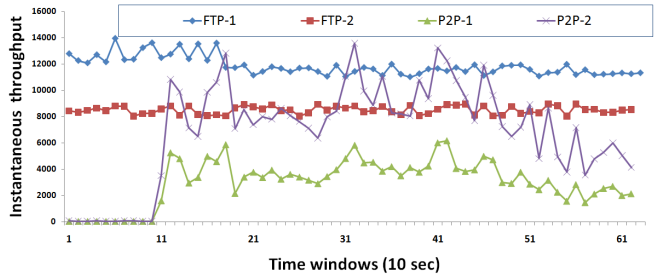


Fig. 4. Instantaneous throughput of p2p (BitTorrent) and non-p2p (FTP) file sharing connections.

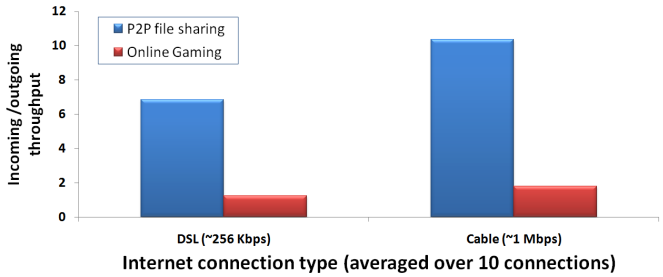


Fig. 5. Incoming-to-outgoing throughput of p2p (eMule) and online gaming connections.

values observed in p2p traffic fluctuate much more rapidly than the inter-connection TTL values for non-p2p hosts. In order to show that geographical diversity is a good feature to identify p2p hosts, Fig. 3 plots the TTL values versus the number of connections for over 5000 p2p and 5000 non-p2p connections. This result confirm that the TTL value for the p2p hosts is more scattered than the non-p2p hosts. Fig. 1(a) shows that, while the TTL feature offers a divergence comparable to existing features. We defer discussion on possible attacks on this feature to Section VII.

4) Instantaneous Throughput: As a precursor to p2p host detection, we also identified a set of distinguishing network layer features for p2p connection identification on a p2p host. In particular, we observed that the underlying peering principles of p2p file sharing protocols result in a per-connection throughput behavior which is significantly different from non-p2p traffic. For instance, Fig. 4 shows the instantaneous throughput of p2p (BitTorrent) and non-p2p (FTP) file sharing connections over 61 windows of 10 seconds each. The obvious difference between the throughput behaviors is observed because p2p applications employ several mechanisms for peer selection such as connection shuffling and optimistic chocking/unchoking. Furthermore, the relationship among peers continuously changes due to high churn rate in p2p networks. Consequently, the incoming and outgoing throughputs among p2p connections are quite dynamic. On the other hand, bandwidth hungry client-server applications like FTP have relatively stable throughput as compared to p2p applications. Thus, instantaneous throughput based features offer a good discrimination between p2p and non-p2p connections, as is also shown quantitatively in Fig. 4.

5) Incoming-to-Outgoing Throughput: Majority of p2p users use DSL/cable modems and have a tendency to choke the upload bandwidth so that their other Internet activities remain unaffected. We show this phenomenon by calculating the outgoing to incoming throughput ratio of 10 cable modem users and 10 DSL users. Fig. 5 compares the throughput of p2p file sharing with another application that requires high incoming-to-outgoing throughput, namely online gaming. It can be seen that in comparison with online gaming users, p2p file sharing users have a higher throughput ratio because of upload bandwidth choking. As a consequence, this feature can be used to distinguish p2p file sharing connections.

D. Summary

We are now equipped with a robust set of privacy-preserving, network layer features that can be used to detect p2p hosts and connections. In the next section, we leverage these features in low complexity statistical frameworks and use ROC curves to evaluate the accuracies offered by these features on the evaluation datasets.

V. Classification using Distinguishing Features

In this section, we leverage the distinguishing features identified in the preceding section to detect p2p hosts and connections. We propose a simple and low-complexity statistical classifier that compares the learned statistical models of traffic features with the features observed in real-time using cross-correlation and log-likelihood tests. The output of these tests is used to classify the real-time traffic. We use $\approx 10\%$ of the datasets for training (the training set), and the rest is used for classification (the test set).

We employ two low-complexity statistical measures to leverage the distinguishing traffic features identified in the preceding section. For both measures, we first learn the p2p file sharing features' PMFs using the training set. The features observed in the unknown trace are then compared against the p2p traffic's PMFs using cross-correlation and log-likelihood tests described next.

A. Classification using Cross-Correlation Test

For classification, we first use the cross-correlation measure [32] which quantifies the similarity between two vectors. To use this measure, we treat the ordered list of distinguishing features as a discrete random process, $\vec{Y}$, where each constituent random variable of the process corresponds to a distinct feature random variable. From the p2p traffic traces, we know the PMFs of each constituent random variable. We treat the distinguishing feature vector $\vec{X}$ from the unknown trace as a realization of this random process. To find the crosscorrelation of the unknown realization with p2p trace realizations, we use the p2p features' PMFs to generate an ensemble of *n* realizations of the random process, say $\vec{Y}_i, i = 1, 2, ..., n$. We then generate a process realization $\vec{Y}$ by taking the ensemble average of each feature; i.e., for a feature indexed at *k*, $\vec{Y}[k] = \frac{1}{n} \sum_{i=1}^{n} \vec{Y}_i[k]$. We compute the cross-correlation of $\vec{X}$ with each $\vec{Y}$ as:

$$O(\vec{X}, \vec{Y}) = \sum_{k \in \phi} \vec{X}[k]\vec{Y}[k], \tag{4}$$

where ϕ is an ordered set of distinguishing feature random variables. High values of cross-correlation imply that the unknown data's features closely match the statistics observed in the training set. Low values of cross-correlation imply that the data potentially comprises non-p2p traffic.

B. Classification using Log-Likelihood Test

Another measure that we use for classification is the log-likelihood of the random process realization observed in the unknown trace. Specifically, assuming independence across features, the likelihood that the unknown realization $\vec{X}$ has been derived from the learned (p2p trace) random process $\vec{Y}$ is

$$L(\vec{X}|\vec{\vec{Y}}) = \log \prod_{k \in \phi} \Big(\Pr\{\vec{Y}[k] = \vec{X}[k]\} \Big) \tag{5}$$

$$= -\sum_{k \in \phi} \log \Pr\{\vec{Y}[k] = \vec{X}[k]\}$$

where ϕ is an ordered set of distinguishing feature random variables. As in the correlation measure, higher values of this measure imply a potential p2p host or connection, while non-p2p hosts/connections should have lower likelihood values.

TABLE IV
EMPIRICAL EVALUATION OF PROPOSED P2P TRAFFIC CLASSIFIER ON NATED TRAFFIC

	Detection Rate	**False Positive**
NATed trace	94.62%	4.87%
without NAT	93.27%	5.57%

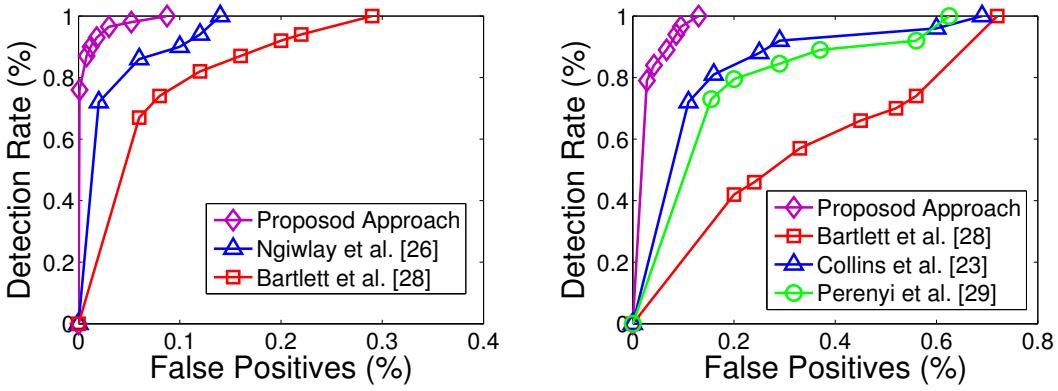
C. ROC-based Performance Evaluation

We plot Receiver Operating Characteristics (ROC) curves for our proposed classifier for detecting the p2p hosts and connections. We classify a host/connection as p2p if either cross-correlation or log-likelihood classifiers flag it as p2p. We change the detection threshold over a range of values and plot the detection rate and false positive rate for each threshold. All performance evaluation is performed using the WIDE dataset and our encrypted dataset described earlier. Fig. 6(a) shows the results of the proposed classifier for host detection using 30 seconds of traffic. Fig. 6(a) also compares the detection accuracy of the proposed approach with that of Ngiwlay et al. [26] and Bartlett et al. [28]. We evaluate [26] and [28] over 10 minutes of trace data (original papers show that an acceptable detection accuracy is achieved by using 10 minutes of trace data). It should also be noted that [26] only works for BitTorrent traffic, and therefore the results for [26] in Fig. 6(a) are for BitTorrent traffic only; i.e., eMule traffic was excluded when [26] was evaluated.

It can be seen from Fig. 6(a), that by using only 30 seconds of trace data, our classifier provides a detection accuracy of 97.84% and a false positive rate of 4.01%. Performance comparison shows that the proposed method renders an improvement of 15% in detection rate, 10% in false positive rate, and 570 seconds over [26]. Similarly, our proposed method provides an improvement of 30% in detection rate, 20% in false positive rate, and 570 seconds over [28].

Fig. 6(b) shows the results of the proposed classifier for connection detection. The proposed classifier provides a detection rate of 93.27% and a false positive rate of 5.57% on the test set. A comparison of the proposed connection-based approach with [23], [28], [29] is also provided. It should be noted that the results for [28] are reported on 10 minutes of trace data for each connection (borrowed from the original paper), while for [23], [29] all the trace data for each connection was used as these are not real-time approaches. Fig. 6(b) shows that for the detection of p2p connections, an improvement of 60% in detection rate rate, 30% in false positive rate, and 240 seconds in detection delay over [28] is observed. Similarly, 20% improvement in detection rate and false positive rates is recorded over [23]. Comparison with [29] shows an improvement of 30% in detection accuracy and 25% in false positive rate. It is evident from Fig. 6(b) that the proposed approach provides much higher detection accuracy for a significantly lower detection delay than [23], [28], [29].

We also evaluate our proposed approach on NATed traffic. For traffic coming from a NAT server, we identified connections using source IP, destination IP, source port, and



(a) Host detection (b) Connection detection

Fig. 6. ROC based performance evaluation of proposed host and connection detection approaches.

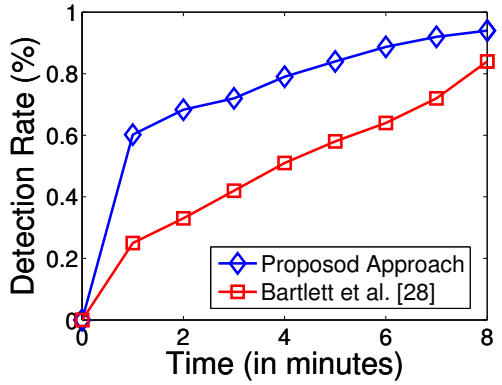


Fig. 7. Time until detection (comparison with [23], [29] cannot be provided as [23] is an offline detection method and [29] requires atleast 10 minutes of trace data for classification)

destination port. It should be noted here that we only use the port numbers to identify connections and do not use them for traffic identification. Table IV shows the results of the proposed classifier for the NATed traffic. The results from Table IV clearly show that the proposed approach remains unaffected when NAT or a proxy server is used.

In addition to having high accuracy, our proposed approach has low detection delay as compared to other approaches. To estimate detection delay, we identify the first data packet between two hosts as the "start" time and then measure elapsed time until we classify that host as p2p. Fig. 7 shows the sensitivity of the detection accuracy to the time delay in minutes. It can be seen from this figure that if only four minutes of trace data for each connection is used, we are able to identify about 80% of p2p connections. Similarly for eight minutes of trace data, about 94% of p2p connections are detected. These results show that the detection delay of the proposed approach is much better than the online detection technique of [28].

VI. Implementation of Wirespeed P2P Traffic Detection on an OpenFlow Testbed

In the last section, we have answered two of the questions that we raised in the Section I. We have demonstrated that a p2p detector can be designed using the network layer features only which can meet the accuracy and detection delay requirements expected from a real-time p2p detector. In this section, we answer the remaining two questions by implementing and deploying the proposed detector in an OpenFlow enabled network.

A. OpenFlow Testbed Setup

OpenFlow allows researchers to innovate and test their novel ideas in a production network. Consequently, over a dozen OpenFlow enabled networks have been deployed at campus networks and this number is growing rapidly [42]. Marvell Technology has created a nationwide OpenFlow network experimentation testbed at NUST, Pakistan. The NUST OpenFlow testbed is used by thousands of students, researchers and faculty members of NUST. This testbed is connected to PERN [http://www.pern.edu.pk] - the largest academic RD network in Pakistan - which connects over 60 public and private sector universities in Pakistan. PERN also has direct connections to TEIN2 (Trans-Eurasia Information Network) [http://www.tein2.net/], the TransPAC2 network [http://www.transpac2.net/], and the Internet2 network [http://www.internet2.edu/].

The OpenFlow testbed at NUST was created using Marvell Technology's four 24x1Gbps and two 48x10Gbps switches. These switches were loaded with a modified firmware running Open vSwitch v. 1.2 [40]. For controllers, the OpenFlow testbed at NUST uses Linux Ubuntu 10.04 machines running NOX controller with Intel Core2Duo 2.0 GHz processors and 3GB of RAM. These controller machines are running NOX OpenFlow Controller [41]. The research labs of the engineering school and the Network Operation Centers (NOCs) are equipped with OpenFlow enabled switches and connected to NOX controllers. This OpenFlow testbed at NUST provides

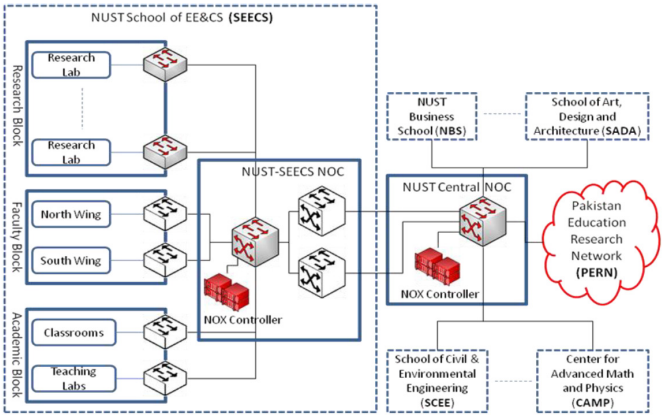


Fig. 8. The OpenFlow Testbed at NUST

researchers complete flow-level control of the research lab's networks including forwarding, routing, and user and host level access control through NOX. Fig. 8 shows the OpenFlow testbed at NUST.

B. Designing an Application for Wirespeed Detection

Most of the modern programmable switches have a conventional high speed data forwarding plane and a slow speed control plane. The control plane controls the flow level behavior of the data plane by writing rules in a policy table in Ternary Content Addressable Memory (TCAM). At the arrival of each ingress packet, a lookup is performed in the policy table and if packet's flow matches a flow specified in a rule, the corresponding action (such as Forward, Drop and etc) specified in that rule is executed. Although the slow path of the switch provides complete control over a packet, the slow path introduces significant delays and it cannot be used to process every packet of every flow. In order to operate at wirespeed, only the first packet of a flow is sent to the controller through the slow path.

For our proposed detector to operate at wirespeed, we have to extract the features listed in Fig. 1 for every flow using the first packet only. However, the first packet can only provide value of two features (number of connections and hop count). Fortunately, controllers can query OpenFlow enabled switches for per flow and per port statistics. Some useful per flow statistics that can be obtained from the openFlow enabled switch include duration (in seconds and nano-seconds), packet count and byte count [43]. Furthermore, this information is stored separately for bi-directional flows using hardware counters in the switches. It should be emphasized here that the proposed approach uses the hardware counters of the switches and OpenFlow is just an enabling technology. Since the hardware counters are available on most of the switches, the proposed approach can be implemented on such switches with some effort. We now design proposed detector as a NOX application using per flow statistics and the first flow packet only.

Our NOX application works by recording a new flow entry whenever switch encounters a new flow and sends information of new flow to the controller. It also writes a rule to the switch to forward the new flow. It then queries the controller for statistics for all the flows that it has recorded after every 30 seconds (one query contains flow stats for multiple flows). Using the statistics in the current 30 second window and previous 30 second window, it updates the values of the features for every flow and it then executes the cross-correlation and log-likelihood tests to determine whether a flow is a p2p flow or not. If a flow is found to be a p2p traffic flow, it modifies the rule at the controller to drop packets for that particular flow. Similarly, it stops requesting flow statistics for a flow that it classifies as a non-p2p flow. We run our NOX application on the NUST-SEECS NOC controller to classify p2p traffic at wirespeed. We have also released our NOX implementation of the proposed approach.

Our testbed setup and NOX application design allows us to answer the remaining two questions that were raised in the Section I. Since the proposed classifier only operates on the features that can be populated based on the hardware counters of switches, it does not affect traffic passing through the fast path. Consequently, it is able to operate at wirespeed. Finally, since OpenFlow enabled switches can coexist with the legacy L2 switches and firmware of a large number of modern switches can be updated to enable OpenFlow on those switches (NEC, Pronto Systems, Marvell, Toroki, HP and a few other vendors provide/ are planning to provide OpenFlow enabled firmware for their switches[4]), the proposed classifier can be incorporated in the existing networking infrastructure with a reasonable installation, management and maintenance cost.

VII. Mimicry Attacks and Countermeasures

To defeat the proposed approach, mimicry attacks can be launched where a p2p host's traffic must have sufficient number of features that match those of a typical non-p2p host's traffic. By studying the divergence measures between p2p and non-p2p traffic types (given in Fig. 1), one can determine which features of p2p applications should be modified to make it appear as non-p2p application. We now discuss mimicry attacks on proposed features and provide countermeasures.

- A p2p host can defeat the proposed outgoing to incoming traffic ratio feature by uploading to and downloading from disjoint set of peers. However, uploading to and downloading from disjoint set of peers is contrary to the peer-to-peer model of p2p protocols which assists in thwarting the leechers.
- To defeat the incoming traffic distribution among connections feature, p2p host will need to control the inflow of traffic from neighboring peers. Due to varying upload bandwidth and lack of control over network activity of neighboring peers, such a distributed coordination requires careful design.

[4]For a list of hardware/commercial switches supporting OpenFlow please see http://www.openflow.org/wp/openflow-components/

- In order to defeat the geographical diversity feature, a p2p host should only communicate with peers at geographically co-located locations. However, such a p2p application would limit the number of peers it can communicate with. Furthermore, in order to defeat TTL distribution as a measure of geographical diversity, a p2p host will have to request its peers to communicate with it using a TTL distribution which is commonly used by non-p2p hosts. Such an attack on TTL distribution is possible and in order to circumvent such an attack, a complex measure such as IP prefix matching can be used as a representative of geographical diversity.
- Instantaneous throughput measure based features can be defeated by maintaining constant throughput with neighboring peers. However, as p2p hosts do not have control over the instantaneous outgoing throughput of neighboring peers, circumventing this feature will require a significantly complex and careful design.

Based on these countermeasures, p2p host/application which tries to circumvent the proposed features would compromise on the desirable characteristics which a p2p application should possess (provided in [34]) to ensure its wide spread use.

VIII. Conclusion

In this paper, we provide evidence to dispel the common belief that network layer features alone are insufficient to provide the accuracy that is expected from p2p traffic detectors. We proposed a network layer detector which, in addition to being accurate, fast, and robust against evasion, outperformed existing application and transport layer detectors on diverse evaluation metrics. We also deployed the proposed detector in an OpenFlow enabled network and show that the proposed approach can be detect p2p traffic at wirespeed with reasonable installation, maintenance and management cost.

Acknowledgments

We thank Hyun-chul Kim for providing Karagiannis et al.'s payload classifier and WIDE trace data for performance evaluation. We thank Marvell Technology for providing us with the OpenFlow testbed. We thank Hasham Asghar at NUST, Pakistan for experimental setup. We also thank Alix L. H. Chow and John Heidemann for their valuable comments.

References

[1] ipoque Internet Study Report 2008/2009, http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.
[2] G. Maier, A. Feldmann, V. Paxson, M. Allman "On Dominant Characteristics of Residential Broadband Internet Traffic", In IMC, 2009.
[3] C. Rossenhvel, "Peer-to-Peer Filters: Ready for Internet Prime Time?", Internet Evolution Evaluations, 2009
[4] ipoque: Bandwidth Management with Deep Packet Inspection, http://www.ipoque.com/.
[5] Cisco SCE 8000 Series Service Control Engine, http://www.cisco.com/en/US/products/ps9591/tsd_products_support_series_home.html.
[6] Procera PL10000 DPI solution, http://www.proceranetworks.com/products/packetlogic-hardware-platforms/pl10000.html.
[7] Sandvine: PTS 24000 Policy Traffic Switch http://www.sandvine.com/products/policy_traffic_switch.asp.
[8] Arbor e100 - DPI Based Technology http://www.arbornetworks.com/en/arbor-ellacoya-eseries-dpi-based-technology.html.
[9] SS8 AXS-2500 Inspection and Surveillance device for service providers, http://www.ss8.com/products-interception-surveillance-AXS-2500.php.
[10] Avoiding traffic shaping using Message Stream Encryption, http://wiki.vuze.com/w/Message_Stream_Encryption.
[11] eMule Protocol Obfuscation, http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=848.
[12] ipoque White Paper: Deep Packet Inspection Technology, Applications & Net Neutrality, http://www.ipoque.com/userfiles/file/DPI-Whitepaper.pdf.
[13] P. Burrows, "Comcast's p2p Conversion: I'll Believe It When I See Results", Business Week, March 28 ,2008.
[14] Canadian Government's Office of the Privacy Commissioner - Deep Packet Inspection, http://dpi.priv.gc.ca/.
[15] Electronic Privacy Information Center, http://epic.org.
[16] S. Sen, O. Spatscheck, D. Wang, "Accurate, Scalable In-network Identification of p2p Traffic Using Application Signatures", In WWW, 2004.
[17] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-service Mapping for QOS: A Statistical Signature-based Approach to IP Traffic Classification", In IMC, pages 135-148, 2004.
[18] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos, "Is p2p dying or just hiding?", In IEEE Globecom, 2004.
[19] OpenDPI, http://www.opendpi.org/
[20] T. Karagiannis, A. Broido, M. Faloutsos, kc claffy, "Transport Layer Identification of p2p Traffic", In IMC, 2004.
[21] T. Karagiannis, K. Papagiannaki, M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark", In IMC, 2005.
[22] F. Constantinou, P. Mavrommatis, "Identifying Known and Unknown Peer-to-Peer Traffic", In IEEE International Symposium on Network Computing and Applications (NCA), 2006.
[23] M. Collins, M. Reiter, "Finding Peer-to-Peer File-Sharing using Coarse Network Behaviors", In ESORICS, 2006.
[24] K. Hyun-chul, kc claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices", In Proceedings of ACM CoNEXT, 2008.
[25] S. Sen, J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks",In IMW, 2002.
[26] W. Ngiwlay, C. Intanagonwiwat, and Y. Teng-amnuay, "Bittorrent Peer Identification based on Behaviors of a Choke Algorithm", In 4th ACM Asian Conference on Internet Engineering, 2008.
[27] Y. Hu, D. Chiu, J. C. S. Lui, "Profiling and identification of p2p traffic", Computer Networks 53(6): 849-863, 2009.
[28] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing", In IEEE Global Internet, 2007.
[29] M. Perenyi, T. Dang, A. Gefferth, and S. Molnar, "Identification and Analysis of Peer-to-Peer Traffic". Journal of Communications, 1(7):36-46, November-December 2006.
[30] B. Raahemi, W. Zhong, J. Liu, "Peer-to-Peer Traffic Identification by Mining IP Layer Data Streams Using Concept-Adapting Very Fast Decision Tree", In ICTAI'08, 2008.
[31] S. Kullback, and R. A. Leibler, "On Information and Sufficiency", Annals of Mathematical Statistics 22: 79-86, 1951.
[32] P. Brockwell and R. Davis, Introduction to Time Series and Forecasting, Springer-Verlag, 1996.
[33] B. Cohen, "Incentives Build Robustness in BitTorrent", In P2PECON, 2003.
[34] J. Liang, R. Kumar, and K. W. Ross, "The Kazaa Overlay: A Measurement Study", Computer Networks Journal, (Elsevier), 2005.
[35] United States Code: Title 18,2511, "Interception and disclosure of wire, oral, or electronic communications prohibited, " http://www4.law.cornell.edu/uscode/html/uscode18/uscsec1800002511----000-.html.
[36] United Kingdom Digital Economy Act 2010, '" http://www.statutelaw.gov.uk/content.aspx?activeTextDocId=3699621.
[37] T. T.T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," IEEE Communications Surveys and Tutorials, (10)4:56-76, 2008.

[38] Programmable Mobile Internet 2020, http://pomi.stanford.edu
[39] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: Enabling Innovation in Campus Networks", ACM SIGCOMM Computer Communication Review, 38(2):69-74, April 2008.
[40] Open vSwitch, An OpenFlow Switch, www.openvswitch.org
[41] NOX, An OpenFlow Controller, www.noxrepo.org
[42] OpenFlow, www.openflow.org
[43] OpenFlow Switch Specification Version 1.0.0 , http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
[44] Marvell Introduces OpenFlow Enabled Switches , http://investor.marvell.com/phoenix.zhtml?c=120802&p=irol-newsArticle&ID=1561824&highlight=