2015 International Conference on Emerging Information Technology and Engineering Solutions

Heuristic-based Real-Time P2P Traffic Identification

Jagan Mohan Reddy, Chittaranjan Hota
BITS-Pilani Hyderabad Campus
Jawahar Nagar, Shameerpet, Hyderabad
Telangana, India 500 078
Email: p2011011,hota@hyderabad.bits-pilani.ac.in

***Abstract*—Peer-to-Peer (P2P) networks have seen a rapid growth, spanning diverse applications like online anonymity (Tor), online payment (Bitcoin), file sharing (BitTorrent), etc. However, the success of these applications has raised concerns among ISPs and Network administrators. These types of traffic worsen the congestion of the network, and create security vulnerabilities. Hence, P2P traffic identification has been researched actively in recent times. Early P2P traffic identification approaches were based on port-based inspection. Presently, Deep Packet Inspection (DPI) is a prominent technique used to identify P2P traffic. But it relies on payload signatures which are not resilient against port masquerading, traffic encryption and NATing. In this paper, we propose a novel P2P traffic identification mechanism based on the host behaviour from the transport layer headers. A set of heuristics was identified by analysing the off-line datasets collected in our testbed. This approach is privacy preserving as it does not examine the payload content. The usefulness of these heuristics is shown on real-time traffic traces received from our campus backbone; where in the best case only 0.20% of flows were unknown.**

I. Introduction

With the rapid development of P2P technology, the P2P traffic has accounted for 40-60% of Internet traffic [1] [2] [3]. The composition of Internet traffic has been changing rapidly in recent times. P2P traffic is a major contributor to the size of Internet traffic in addition to WWW, FTP, Email etc [4]. In recent years, the crack down on the use of P2P has forced P2P applications to find out alternate ways (port masquerading, relays, tunnelling, encryption etc) to hide their traffic on the network. While there is an evidence that P2P traffic is decreasing [5], it still represents a significant fraction of the Internet traffic, and is perceived as wasteful of network resources [6].

P2P traffic identification has attracted great attention recently due to its' massive adaptability within the network users, and due to its' ability to create severe security concerns. Also P2P classification has emerged as a vibrant commercial problem over the last few years. While many DPI products have emerged in response to solve this problem [7], [8] a main complexity of solving P2P classification remains the following: new, non-standardized P2P protocols and clients keep gaining popularity and consequently signatures have to be designed and updated all the time. Moreover, most P2P clients and protocols are now introducing intentional encryption and obfuscation to avoid detection. Finally, and perhaps most importantly, DPI-based products are not experiencing widespread deployments, especially in India, and other developed countries, because of their legal and ethical implications. Therefore, there is a dire need to develop P2P classifiers that are payload oblivious, but can still meet the accuracy and complexity constraints. The objective of this paper is to improve the P2P traffic identification accuracy based on the application profile or the connection patterns of the host. We can successfully distinguish P2P traffic from non-P2P traffic coming in or going out from a host using the heuristics proposed in the work. The heuristics proposed in this work achieve greater accuracy and minimal classification time for identifying a host as a P2P host.

One of the early techniques to identification of traffic at application level was based on port numbers. IANA [9] has a list of registered port numbers like DC++ uses 411, 412, and 1025-32000 port numbers for TCP connections; BitTorrent uses 6881-6889 range of port numbers for both TCP and UDP etc. This method is basic, straight forward and easy to detect P2P traffic in a network. Most P2P applications have default port numbers on which they function. With port based analysis, it is easy to identify whether traffic is P2P or not. But it has several limitations. Most of the P2P applications change their default port numbers by allowing users manually to configure whatever they like. Additionally, many newer P2P applications begin to masquerade their port numbers with well-known port numbers, such as port: 80. Hence, port-based analysis is less effective in traffic identification.

Signature-based classification techniques to monitor traffic passing through the network and examine the payload of each packet which is already a known signature of P2P application. This process is referred as Deep Packet Inspection (DPI). There are several commercial and open-source tools available to the research community to identify the P2P traffic, according to signatures and these tools map few Bytes from every packet payload written in some regular expression format. Few of these include, L7-filter [10], Juniper's Netscreen-IDP etc. These tools monitor every packet payload and raise alerts when the predefined signature is matched, e.g. the string "/announce" exists in a packet on the Torrent P2P network. With this type of classification technique we can overcome the demerits of port based classification techniques. But, when P2P applications are evolving continuously and their signatures can change, we also need to keep monitoring and update new signatures observed. However, this type of classification has limitations like, if the P2P applications use tunnels or encrypted traffic, it is difficult to detect it. This type of classification also violated privacy of users.

A more recent classification technique uses statistical properties of network flows whose success lies heavily on the training dataset and the Machine learning algorithms used to classify the P2P traffic [11] [12] [13]. However, ensuring accuracy and authenticity of the training sets is still an open

978-1-4799-1838-6/15 $31.00 © 2015 IEEE
DOI 10.1109/EITES.2015.16

issue, particularly for flows that go undetected.

The objective of this work is to improve the P2P traffic identification based on the application profile of the host. This technique observes the communication patterns of an end-host and it is important to understand how different applications affect hosts. We have developed heuristics, based on the application behaviour on an active host to differentiate P2P and web traffic. In this work, we captured the traffic of P2P and Web traffic in our test-bed. We extracted 5-tuples Source IP, Destination IP, source port, destination port, protocol as metrics from every network capture. Then, we analysed the patterns of P2P and Web using MySql databases. The details of our framework are explained in Section VI.

The rest of the paper is organised as follows: Section II covers related work on host based identification schemes. Section III describes differences between various P2P and nonP2P applications. Section IV describes a framework for P2P traffic identification and datasets used (generated on our testbed and datasets collected from other sources). In section V, we propose a minimal set of heuristics identified from the datasets collected and describe their behaviour to help identify a P2P host. Section VI describes our experimental results and section VII concludes the paper.

II. Related Work

Hurley *et al.* [4] proposed a set of heuristics for P2P traffic and Web traffic identification by using host behaviour. They could identify 90% of the flows accurately that go out and come in to a host. The heuristics were developed by using information like source host, destination host, connections between the hosts, and the flow activity like how many packets per flow etc.

Yan *et al.* [1] proposed P2P traffic identification scheme based on both host behaviour and flow behaviour. First, they identified whether a host is running a P2P application by matching its' behaviour with a set of predefined rules like number of ports open, number of IPs connecting, number of failed connections, etc. Next they refined this identification by comparing the statistical features of each flow in the host with several flow feature profiles like flow duration, flow volume etc. The identification accuracy was above 90%.

John *et al.* [11] proposed heuristics to classify Internet traffic based on network applications that include P2P applications along with other types of applications. The heuristics used were based on connection patterns between two hosts. Heuristics like usage of both TCP and UDP concurrently, a particular port usage, ratio of IP/port pairs, etc. were used to achieve identification accuracy of up to 99.8%.

Karagiannis *et al.* [14] proposed a traffic classification approach based on host behaviour at the transport layer. They looked at a multilevel approach to classify traffic, according to the applications that generate them. Their traffic classification approach uses host behaviour first and then social, functional, and application behaviours. At the social level, their approach identifies hosts with similar behaviour which is evident from the interactions a host makes with other hosts. At a functional level, it identifies what function a host plays in the network, i.e. either a provider or a consumer of the service, or both. At an application layer, they look for transport layer interactions to identify the traffic.

Perenyi *et al.* [15] derived a set of heuristics from the robust properties of P2P traffic collected or revealed from a traffic aggregation. They also presented a traffic analysis based on the behaviour of active users, the ratio between P2P users and normal users etc. and observed that the daily profile of P2P traffic intensity is less variable.

Naimul *et al.* [16] compared the web and P2P traffic analysis using statistical measures and models like inter-quartile (IQR), probability density function (PDF), cumulative distribution function (CDF) and complementary CDF (CCDF). They proposed three flow-level metrics: flow size, flow duration and flow inter-arrival time and three host-level metrics: flow concurrency, transfer volume and geographic distribution. They also identified that web flows are short-lived whereas P2P flows are long-lived.

The heuristics developed in this work scores over other related works in terms of the minimum number of packets required to identify the host the behaviour, and minimum number of heuristics achieve more than 99.8% accuracy for identifying P2P host.

III. P2P and NonP2P Applications

In this section, we discuss the behaviour of two groups of protocols, P2P and nonP2P. We have chosen popular P2P protocols like eMule, uTorrent, Skype and for nonP2P protocols like HTTP(S), SMTP, FTP, Dropbox. To distinguish between P2P and nonP2P, we developed a series of heuristics after understanding the behaviour of these protocol.

A. NonP2P

These applications typically follow a client/server behaviour. The client always initiates a TCP connection and the server responds to its' request. A standard web browsing connection (either HTTP or HTTPs) is accomplished by a 3-way handshake i.e. client initiates TCP SYN flag to a web server over port 80 or 443 and then server replies with TCP $SYN - ACK$ flag. Then the client acknowledges back with ACK flag. After the 3-way handshake is completed the client requests the desired information from the server. Here, the requesting host is always a client and responding host is a server. This activity is common for most of the web applications. There are few applications which are slightly different in terms of using port numbers 80 and 443. File transfer protocol (FTP) also is a client/server protocol. FTP uses two control channels, one channel for control data that the server accepts from the client over port 21 and the other channel is utilized for data transfer from server to client over port 20. Dropbox is a cloud based application that also adopts client/server behaviour. A client host requests a Drobox server to access the files over HTTPs and these files are then downloaded into the local host machine. This protocol is different from the FTP service in a way where it can $sync$ any shared file in the Local Area Network (LAN) without connecting to the Dropbox server. It can also broadcast the network over port number 17500 if the host enables the LAN $sync$ option.

Algorithm 1: hostAnalyser

Data: Heuristics as queries
Result: returns as flows
String $query, flow \leftarrow$ null;
read the heuristics from text file;
read the database server path;
$conn \leftarrow$ getConnection(db_path);
while *query* $\neq$ *null* **do**
    result $\leftarrow$ excuteQuery(query);
    while *result* $\neq$ *null* **do**
        flow $\leftarrow$ result.getString(db_fields);
    return flow;

B. P2P

P2P applications are different from the traditional client-server architecture. The motivation behind the design of these applications is to share files amongst the collaborating peers over the IP layer. In this, a host can act as a client and also as a server to maximize their file sharing benefits. In this section we briefly discuss about the most common P2P networks like Gnutella, eMule, Skype and μTorrent. These protocols never generate any DNS queries once these are executed over a P2P network, which is a major difference between P2P and nonP2P.

$Gnutella$ was the first decentralized P2P network and it came with the concept of Super peers. In this network every host acts as a client as well as a server. To join the Gnutella network, a host must send a request for a pre-defined server namely Super-peer in the network which is already set-up for the Gnutella client. Once the Super peer accepts the peer request, then it can be the part of the Gnutella network. After which it can find the information from the other hosts. To search for a file in the network it initiates $PING$ messages to all of its neighbours. Who ever has the file will reply with a $PONG$ message over the UDP. Once the file is available in the network, the host directly communicates to retrieve the file over TCP.

$eMule$ P2P application uses eDonkey network which is based on the centralized sever concept. eDonkey network forms a logical ring network where each peer is assigned an ID based on a hash function. Once the client is part of the network it will exchange information with all the servers. Initially the client connects over TCP to log into the server. The server uses another TCP connection to perform a client-to-client handshake for accepting connections from other eMule clients and then the sever closes the second connection. eMule client and server both use UDP for keep-alive messages and for enhancing the search [17].

$\mu Torrent$ is a variant of BitTorrent client owned by BitTorrent, Inc. The client needs a $.torrent$ file to download a file which contains a list of peer URLs called seeders. These URLs are associated with a tracker server that is a centralized component. The client connects to these tracker servers over UDP or TCP. These servers only provide the information about the seeders. Once the client finds the file from the seeders, the data transfer begins from multiple peers over TCP connection. Downloading peers are called as leechers.

$Skype$ is a Voice over P2P (VoP2P) application. It uses P2P networks to discover peers. It contains three main components, i.e., Skye Client (SC), Skye Server (SS), and Super Nodes (SNs) [18].

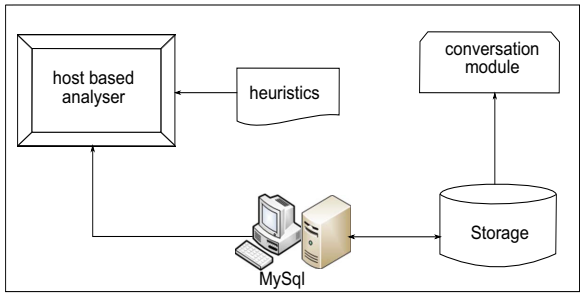

Fig. 1. Framework for P2P Traffic Identification

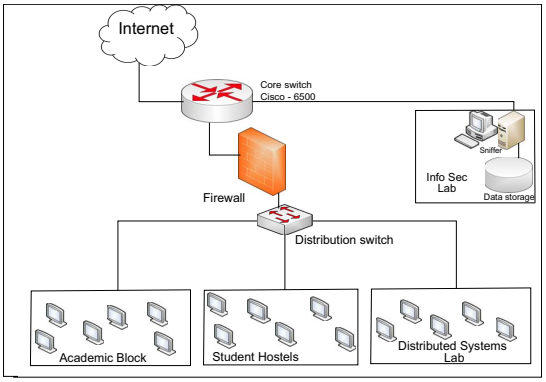

Fig. 2. Testbed for Data collection

i. Skype Client (SC): SC or ordinary node is used for host login with SS. Once it authenticates successfully with the SS, this information is made available to the SNs.

ii. Skype Server (SS): Is a traditional central server that maintains all the user's account information.

iii. Super Node (SN): SNs are the end points of SCs which are used to connect each other SCs. SNs can be designated by Skype itself. SNs are very high-end systems with a powerful CPU, enough memory and large network bandwidth.

The SCs open randomly chosen TCP and UDP listening ports. SCs bootstrap themselves by connecting to a SN over a UDP connection with a fixed packet size. Then SN opens a TCP port to exchange it's information with SCs. Once SN recognizes an SC, it allows it to login using a TCP connection. In this work we have used SN data for the purpose of classification.

IV. Framework for Traffic Identification

The objective of our proposed technique is to improve the P2P traffic identification accuracy based on the application profile of the host. This technique observes the communication patterns of an end-host and it is important to understand how different applications affect hosts which we have discussed in the previous section. We imported packet information into the MySql database. Then, we analyse the connection patterns

TABLE I. APPLICATION WISE STATISTICS

Application	Date (captured/Obtained)	Pakets	Flows	Bytes
web	14 May, 2013	10534 K	137465	2810 M
Dropbox LAN	06 June, 2013	2389 K	133	182 M
FTP (control)	08 June, 2013	1096 K	7898	98 M
Smtp	24 September, 2012	49 K	658	40 M
eMule	02 August, 2013	20984 K	179235	2310 M
Frostwire	02 August, 2013	26766 K	771204	3150 M
Skype	23 October 2013	597 K	35145	2080 M
μTorrent	02 August, 2013	24176 K	526141	2710 M
Vuze	02 August, 2013	16270 K	580154	1830 M

of P2P and nonP2P. Figure 1 shows the framework of our approach to develop the heuristics. We implemented a Java interface to query the database and developed heuristics given in Algorithm 1. In later sections, we discuss the usage of our heuristics. Then flows are constructed using 5-tuple i.e., source IP, source port, destination IP, destination port and protocol. TCP flows are separated by FIN, ACK or RST flags, whereas UDP flows are terminated based on Timeout of 600 seconds. We have implemented a Java module to construct flows which is given in Algorithm 2. The first for loop iterates over the list of packets. If the packet is TCP with FIN or RST flags set, then the TCP flow is added to the conversation.

In this section, we provide details of the datasets used in this research and datasets obtained from other sources. We independently collected network traces from our campus LAN that is connected to the Internet by a 155 Mbps STM link. We collected nonP2P (HTTP, HTTPs, and SMTP) applications traffic and P2P applications traffic (Torrents, Gnuetlla) using Wireshark [19] tools. Our dataset generation was performed on the testbed which is shown in Figure 2. Due the privacy concerns, we captured only first 130 Bytes of each packet. We captured around 250 GB of data on our tetsbed. FTP dataset was obtained from Lawrence Berkeley National Laboratory (LBNLab) [20] which has no payload information and the IP addresses were anonymized. Part of the P2P traces used in our work was obtained from the University of Georgia (UGA) [21]. Statistics of nonP2P and P2P application traffic in our datasets is shown in Table I.

V. PROPOSED HEURISTICS

In this section we explain the P2P traffic heuristics based on the host behaviour that were observed from the dataset collected. Heuristics devised to contain some well-known port number information as well. We have also derived tunable thresholds to trigger application of our heuristics. We classify the P2P traffic from the proposed heuristics and we also identify the False Positives (FP). Our work in this paper differs from other related works in terms of using lesser number of packets in the flows. These heuristics are tested in real-time with one minute interval by running both P2P and nonP2P applications in the campus backbone. We have filtered few other types of traffic seen on the campus LAN, i.e., services like NETBIOS, NTP, DHCP, SSDP, and Link local from both TCP and UDP protocols. The heuristics used are described as below:

Algorithm 2: Conversation

```
Conv c;
HashSet< Conv > cset;
Map< String, Conv > ConvMap;
Conv: Conversation
getTstamp: gets the time stamp of the packet.
getSig: gets the flow signature.
getSIP, getDIP: gets the Source and Destination IP
addresses.
getSPort, getDPort: gets the Source and Destination
Port numbers.
Data: getConv(ArrayList < Packet > list) input is
      a list of packets.
Result: Set< Conv > convSet; conversations in
        convSet.
begin
    cset ← new HashSet< Conv >()
    ConvMap ← new HashMap< String, Conv >()
    for Packetp ∈ list do
        if ConvMap.containsKey(p.getSig()) then
            if p.isTcp() and
            (p.Tcpflag.RST||p.Tcpflag.FIN) then
                ConvMap.get(p.getSig()).addPacket(p)
            else if p.isTcp() ≠ ∅ then
                if p.getTstamp()-
                ConvMap.get(p.getSig()).last ≥
                udptimeOut then
                    c = Conv(p.getSIP(), p.getDtIp(),
                    p.getSPort(), p.getDPort(), p.isTcp())
                    c.addPacket(p)
                    cset.add(c)
                    ConvMap.put(p.getSig(), c);
                else
                    ConvMap.get(p.getSig()).addPacket(p);
            else
                ConvMap.get(p.getSig()).addPacket(p);
        else
            c ← Conv(p.getSIP(), p.getDIP(),
            p.getSPort(), p.getDPort(), p.isTcp())
            c.addPacket(p); cset.add(c);
            ConvMap.put(p.getSig(), c);
    for Conv co ∈ cset do
        co.updateValues(); co.freeSpace();
    return cset;
```

A: **P2P TCP/UDP protocols:** This heuristic is based on the fact that the P2P applications like Gnutella, Skype, etc. use both TCP and UDP protocols. In most of the scenarios the TCP is used for data transfers whereas UDP is used for signalling messages. There may be an FP with this heuristic with UDP like default LAN services. However nonP2P applications make communication parallel over TCP.

B: **P2P Source Port and Destination Port:** In general all most all the P2P applications allow change in port numbers or the application itself will use a random port

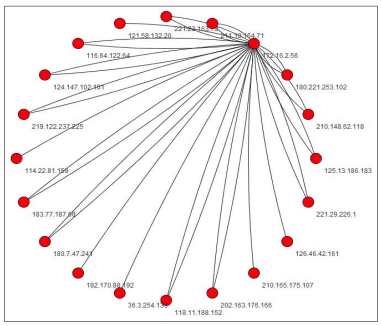

Fig. 3. Behaviour of Heuristic B

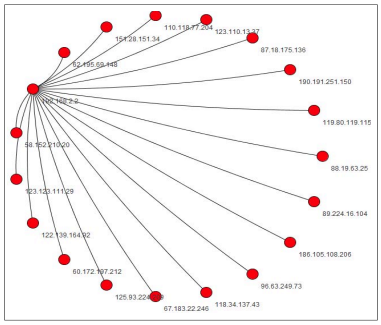

Fig. 4. Behaviour of Heuristic C

to signal the peer in the overlay. However, the P2P application has the following property. Both source IP and destination IP communicate with the same port over UDP protocol. This property is unique for UDP, but not for TCP. All flows to and from these hosts are classified as P2P.

C: **P2P UDP/TCP port pairs:** This heuristic is derived from the above one (B). The fact that, P2P applications use UDP for signalling or control messaging, any host can communicate using a default port or an ephemeral port over UDP on the destination side, the same port can be assigned to TCP for data transfer. This property is unique in nonP2P applications. All the TCP and UDP flows directed to and from a host are classified as P2P hosts. This heuristic can be applicable for both source IP and destination IP pairs, if both of them are involved in sharing of files in the overlay network. For example, a host (source/destination) communicates on port $XXXX$ over UDP and for data transfer the other end opens same port $XXXX$ to TCP. This heuristic when used classified Dropbox traffic wrongly as P2P traffic. This is because of the Sync behaviour of Dropbox.

D: **P2P UDP ports:** This heuristic exploits port based classification where the peers in the P2P network use well-know ports like port 80 and 443 over UDP to communicate outside the servers which can bypass the firewall. In general, these ports are being used with TCP connections for retrieving web contents. If any host uses these port numbers to and fro, that host is marked as P2P and all these flows are classified as P2P. Our experimental

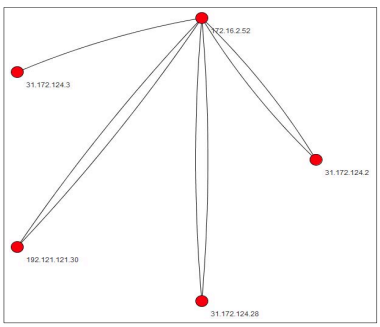

Fig. 5. Behaviour of Heuristic D

results reveal that the P2P hosts use these ports quite often in their lifetime.

E: **P2P Unique IPs and Unique Ports:** When the P2P client initiates a connection in the network it signals multiple peers. At this point, it opens multiple port numbers to get the information about other peers over UDP and this will continue till the desired information is obtained. In our analysis, if a host is a P2P host, it must use TCP and UDP and the number of unique destination IPs are less than the number of unique ports and all flows are classified as P2P. On the other-hand nonP2P applications typically use multiple connections with a web server over TCP. Typically, FTP and SMTP servers accept multiple connections over the same port, but geographically, the servers communicate distinct IPs and distinct port numbers over TCP, which are marked as nonP2P hosts, i.e. the host must use TCP and the number of unique IPs is less than or equal the number of unique ports and the ratio of the TCP packet count is one.

F: **P2P TCP-UDP percentage:** The ratio of TCP and UDP define the number of packets that a host communicates on both TCP and UDP divided by the total number of packets transmitted. The fact that P2P uses both transport protocols TCP and UDP, which is not the case with nonP2P applications. We observed that the value of TCP percentage and UDP percentage is less than one and every flow should contain at least two packets.

VI. Validation and Experiment Results

In this section we validate the performance of our heuristics to identify P2P hosts in real-time. To validate our approach, we implemented the algorithms in Java using jNetPcap library [22]. The experimental results obtained on the dataset are given in this section. We created flows from network traces for both TCP and UDP. Our validation dataset contains 2091879 flows of P2P and 146154 flows of nonP2P. The heuristics that we derived are applied to these data to identify P2P and nonP2P traffic.

To verify the correctness of our heuristics, we constructed graphs using JUNG library [23] from the dataset. Figure 3 describes our second heuristics which says that P2P hosts use the same port numbers over UDP and some of the P2P hosts use bidirectional communication (e.g. 124.147.102.101,

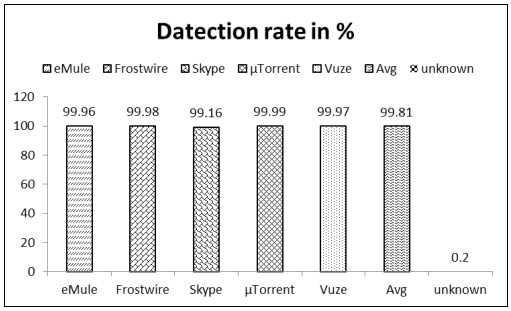

Fig. 6. Detection Rate of P2P traffic in Flows

221.29.226.1 and so on). Similarly, our third heuristics where same port numbers being used over TCP and UDP is as shown in Figure 4. Figure 5, describes the P2P hosts that use default port numbers 80 and 443. By running through heuristics from A-F, the result is promising for identifying P2P hosts while eliminating misclassification of the nonP2P hosts. The average detection rate is found to be more than 99%. The amount of unknown traffic is around 0.2%. The total number of unclassified flows, for Vuze is about 0.01%, μTorrent is about 0.01%, Skype is about 0.84%, Frostwire is about 0.02% and eMule is about 0.04% as shown in Figure 6. Total 594 flows remain unclassified as P2P traffic out of 2091879.

VII. Real-Time Analysis

Our objective is to identify P2P and nonP2P flows even if a host uses both applications simultaneously. In this section we discuss the effectiveness of our heuristics to identify a host as a P2P or nonP2P in the form of traffic flows based on a specific time period. For this purpose we consider, a flow to have at least two packets in UDP. For TCP only packets carrying payload are used. We assumed, if any host uses the default port number 80 over TCP then it is treated as nonP2P traffic. We experimented measuring the heuristics for different values for a time window of duration W from 1 to 2 minutes. For the time window $W = 1$, we captured both P2P (torrents) and nonP2P (youtube) originating from a host. From the captured file we had 238 flows of 210 are UDP flows and 28 are TCP flows. Out of 238 flows our heuristics classified 218 flows as P2P. Similarly, for the time duration of 2 minutes we had 379 flows out of 289 flows are UDP and 90 of them are TCP. The heuristics A-F classified 341 as P2P.

VIII. Conclusion

In this work we proposed a new set of heuristics to identify a host as a P2P host based on its connection patterns. The proposed heuristics do not require any payload signatures. Our experimental result shows that the proposed rules are able to classify the P2P hosts effectively and suggested heuristics are promising. The dataset used as realistic in nature and we verified our approach in a real-time scenario too. We also presented the comprehensive behaviour analysis of our P2P hosts. Further, our approach has minimal heuristics which can be deployed easily in real-time. The unclassified traffic is about 0.2% of the P2P traffic. However, our approach can only identify broad P2P applications rather than different P2P applications. Our future work will focus on further investigation of fine-grained P2P traffic classification.

Acknowledgment

This work was supported by a grant from Tata Consultancy Services (TCS) under research scholar program, India.

References

[1] J. Yan, Z. Wu, H. Luo, and S. Zhang, "P2p traffic identification based on host and flow behaviour characteristics," *Cybernetics and Information Technologies*, vol. 13, no. 3, pp. 64–76, 2013.

[2] "ipoque inc." 2008. [Online]. Available: http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf

[3] "Sandvine report on p2p," 2008. [Online]. Available: https://www.sandvine.com

[4] J. Hurley, E. Garcia-Palacios, and S. Sezer, "Host-based p2p flow identification and use in real-time," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 2, p. 7, 2011.

[5] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 75–86, 2011.

[6] J. Seibert, R. Torres, M. Mellia, M. M. Munafo, C. Nita-Rotaru, and S. Rao, "The internet-wide impact of p2p traffic localization on isp profitability," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 6, pp. 1910–1923, 2012.

[7] "Procera pl10000 dpi solution." [Online]. Available: http://www.proceranetworks.com/products/packetlogic-hardware-platforms/pl10000.html

[8] "Cisco sce 8000 series service control engine." [Online]. Available: http://www.cisco.com/en/US/products/ps9591/tsd-products-support-series-home.html

[9] "Iana." [Online]. Available: http://www.iana.org

[10] "l7-filter." [Online]. Available: http://l7-filter.sourceforge.net

[11] W. John and S. Tafvelin, "Heuristics to classify internet backbone traffic based on connection patterns," in *Information Networking, 2008. ICOIN 2008. International Conference on*. IEEE, 2008, pp. 1–5.

[12] J. M. Reddy and C. Hota, "P2p traffic classification using ensemble learning," in *Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop*. ACM, 2013, p. 14.

[13] P. Narang, J. M. Reddy, and C. Hota, "Feature selection for detection of peer-to-peer botnet traffic," in *Proceedings of the 6th ACM India Computing Convention*. ACM, 2013, p. 16.

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[15] M. Perényi, T. D. Dang, A. Gefferth, and S. Molnár, "Identification and analysis of peer-to-peer traffic," *Journal of Communications*, vol. 1, no. 7, pp. 36–46, 2006.

[16] N. Basher, A. Mahanti, A. Mahanti, C. Williamson, and M. Arlitt, "A comparative analysis of web and peer-to-peer traffic," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 287–296.

[17] Y. Kulbak, D. Bickson *et al.*, "The emule protocol specification," *eMule project, http://sourceforge. net*, 2005.

[18] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," *arXiv preprint cs/0412017*, 2004.

[19] "Wireshark," 2012. [Online]. Available: http://wiki.wireshark.org/Tools

[20] "Lbnl/icsi enterprise tracing project," 2005. [Online]. Available: http://www.icir.org/enterprise-tracing/download.html

[21] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," *Journal of Information Security and Applications*, 2014.

[22] "jnetpcap," 2013. [Online]. Available: http://jnetpcap.com

[23] J. OMadadhain, D. Fisher, S. White, and Y. Boey, "The jung (java universal network/graph) framework," *University of California, Irvine, California*, 2003.