Classifying P2P Activities in Netflow Records: A Case Study (BitTorrnet & Skype)

by

Ahmed Bashir

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Masters of Applied Science

in

Systems and Computer Engineering

Carleton University

Ottawa, Ontario

To my Father.

R.I.P

Abstract

This thesis addresses the problem of identifying BitTorrent and Skype activities within Netflow traces. The ability to accurately classify different types of internet traffic using Netflow traces represents a major challenge in the field of internet traffic classification since there is no payload information available with Netflow. Nowadays, P2P applications represent a large portion of the internet traffic and are becoming more difficult to classify. In this thesis a simple yet effective classification method is proposed using a set of heuristics based on the discriminating features and the nature of P2P applications.

The presented scheme has been tested with a collection of real data sets. The results of the classification have shown to be accurate when applied to data sets with challenging internet traffic. Furthermore, the results of the proposed scheme have proven to be superior to other existing approaches in terms of the accuracy of the classification.

Acknowledgement

First and foremost, I offer my utmost gratitude to my supervisors Dr.Changcheng Huang and co-supervisor Dr. Biswajit Nandy as this thesis would not have been possible without their encouragement, guidance and support from the initial to the final level of this research. I would also sincerely thank everyone who supported me in any kind of way during the completion of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

This chapter provides a brief introduction to P2P systems, explaining the basic concepts and operation mechanisms behind P2P technologies. Afterwards, it shows the importance of accurately identifying P2P traffic in different networks as nowadays P2P represents the majority of the internet traffic. The different P2P classification techniques are discussed to shed light on both their advantages and disadvantages and are used as a motivation to create a new P2P identification scheme. Finally, the main contributions and the overall organization of this thesis are mentioned in the end of the chapter.

1.1 Background

Peer-to-Peer or P2P is a type of distributed systems where different nodes form a dynamic overlay network. The participating nodes or peers have the ability to pool their resources in order to achieve the major goals of distributed systems such as scalability, resource sharing and fault tolerance.

The P2P architecture is considered to be the opposite of the client-server model. The client-server architecture has a centralized server responsible for controlling the access of shared resources within the network, giving the clients limited privileges. In a P2P network, each peer is considered to be a client and a server simultaneously, due to the absence of a centralized server. Although this may serve as an advantage, it raises many management and security issues since there is no control over the content being shared within the network. Consequently, the participating peers become prone to various threats and security violations.

Figure 1 and Figure 2 display the architectural difference between the client-server model and the P2P model.

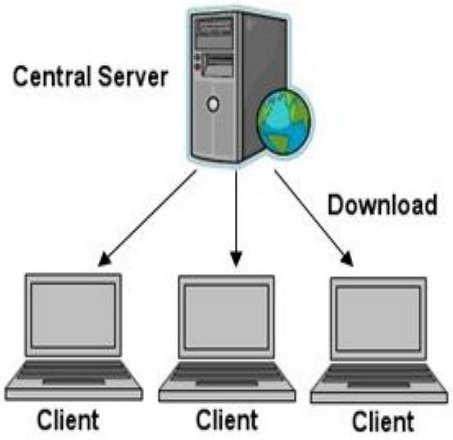

Figure 1 Client Server Architecture

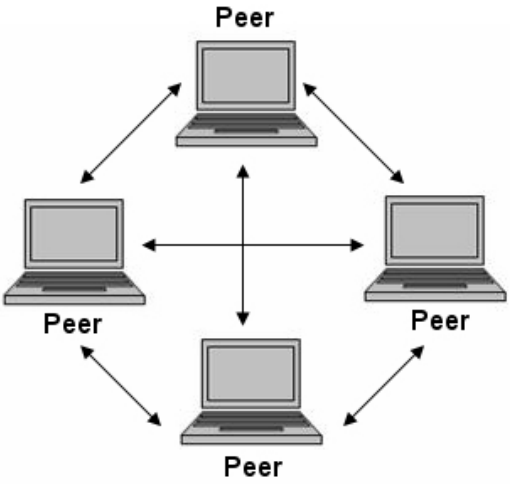

Figure 2 P2P Architecture

Despite the issues brought up by P2P, its connections represent a significant portion of the internet traffic. P2P connections enable the direct communication between peers where they can share various services and resources together. These services include file sharing, video streaming, online gaming and other activities that the client-server architecture cannot accomplish as fast or as efficient as the P2P architecture. Looking at the P2P network from a top perspective, it can be viewed as a big pool of shared resources, where every participating edge has to “give” in order to “take”.

Today P2P is considered as a standard backbone for sharing all kinds of different media throughout a network. Examples include audio, video and real time services that are time sensitive such as voice over IP (VoIP). Internet Service Providers (ISPs) and network

administrators have been dealing with the challenge of classifying P2P traffic within a network to control and limit the Quality of Service (QoS) accordingly.

1.2 Research Problem

Since the early 2000s, P2P has emerged as a dominant portion of the internet traffic. According to Ipoque; a German company responsible for developing bandwidth management solutions, P2P traffic represents 49-83% of the total internet traffic in several countries [1]. The enormous amount of P2P traffic and its rapid progression throughout the years have resulted in deteriorating network performance and congestion due to the massive bandwidth consumption of P2P applications. P2P applications are becoming more difficult to classify since they operate with a set of random ports and sometimes utilize encrypted payloads. In addition, heavy bandwidth consumption is usually a result of P2P applications such as files sharing and BitTorrent. Therefore, network analysts and administrators are concerned with accurately identifying P2P traffic and detecting such activity in their networks.

The ability to accurately classify P2P traffic finds utility in provision of QoS, network security and network planning for public network providers and campus network operators. The existing apparatuses for classifying P2P traffic utilize an array of different methodologies and techniques, each yielding a different rate of success depending on the targeted traffic to be classified. In the early days of the internet, P2P traffic could be easily classified by using port based approaches as suggested in [2-6]. Although port based approaches give high accuracy for identifying legacy applications (e.g. HTTP, FTP, NetBIOS), this technique is highly inaccurate for identifying P2P traffic as P2P applications utilize random ports to navigate through firewalls and

other network restrictions. Different payload analysis techniques [7-16] were suggested to classify P2P traffic, achieving very high rates of success. Despite its success, this approach cannot be applied to encrypted payloads or to the situation when payload information is not available. On the other hand, machine learning and data mining approaches [17-24] were suggested, where key attributes and discriminating features of P2P applications are used for classification by feeding training samples to the algorithm. The accuracy of this technique relies heavily on the quality and nature of the training samples, and the robustness of the selected features.

1.3 Research Objective

From the previously discussed problems, the main objectives of this thesis are to analyze the behavior and discriminating features of P2P applications, as well as proposing a classification model that can successfully identify and extract BitTorrent and Skype flows from Netflow traces with high accuracy.

The classification of P2P applications can be utilized to serve a variety of purposes. ISP and carriers can benefit from it in improving the QoS. By identifying Skype connections, they can be prioritized in order to enhance the calling experience, improve the call quality, and avoid delay and lag in Skype calls. In addition, the calculation of an estimate of the duration of any placed call is advantageous in providing billing information and accounting. Furthermore,, by identifying BitTorrent activities, they can be also prioritized in order to achieve faster file transfers. In addition, the classification can be used for enforcing fair internet usage policies by determining the total consumed bandwidth of each host and limiting it accordingly.

Consequently, if a user is abusing his bandwidth privileges through BitTorrent file transfers, the bandwidth allocated to the host can be limited and if needed, the BitTorrent ports can be blocked. On the other hand, network analysts and home users can benefit from the classification model by utilizing the information in determining the nature of their internet traffic, and identify different P2P activities within their networks.

Recently the focus on accurately categorizing any type of internet traffic within Netflow traces is becoming of vital importance as Netflow is the standardized tool utilized by network administrators for monitoring and collecting information regarding internet traffic within a network. This thesis is concerned with two of the most important P2P applications. BitTorrent and Skype represent the leading applications in their corresponding fields. BitTorrent is the most popular tool used for P2P file sharing over the internet according to different studies and articles [25-26], while Skype is by far the number one application when it comes to VoIP with over 350 million active users [27].

1.4 Suggested Approach

By comparing and evaluating the previously proposed approaches in P2P traffic classification, a classification scheme is proposed to identify both BitTorrent and Skype traffic using Netflow traces. The proposed scheme is based on a combination of existing approaches and a set of features and characteristics unique for BitTorrent and Skype applications.

In specific, we analysed the connection patterns and distinguishing characteristics of both BitTorrent and Skype and studied how they behave on start up, mid process, and in their closing

stages. The classification procedure takes place by converting these features and behavior patterns to statistical data and comparing them to their corresponding data fields in the Netflow traces to determine the existence of BitTorrent and Skype activities.

1.5 Summary of Contributions

The key contributions of the thesis are:

1- A proposed classification model capable of Identifying BitTorrent and Skype hosts and flows.
2- The proposed model has been tested with five different data sets and produced accurate results for both BitTorrent (91.3%-95.4% byte-wise accuracy) and Skype (100% flow accuracy, ±1-2 minutes for placed call duration accuracy).
3- The results of the proposed scheme have proven to be superior to other existing approaches in terms of accurately identifying BitTorrent and Skype flows within Netflow traces.

1.6 Thesis Organization

The remaining chapters of the thesis are organized as follows:

Chapter 2: A brief introduction and a literature survey of the current suggested approaches for general and P2P traffic classification as well as a detailed description of two of the most popular P2P applications; BitTorrent and Skype. Furthermore, an introduction to Netflow and a list of discriminating P2P flow features are presented in the end of the chapter.

Chapter 3: Introduces a list of the heuristics used for building the classification scheme for identifying BitTorrent and Skype flows. These heuristics were obtained by analysing both applications' activities and the list of unique features for both applications.

Chapter 4: Introduces the testing and experimentation phase of the classification procedure by applying the proposed scheme to real life data sets and presents the results and accuracy of the classification scheme. Additionally, comparisons with some existing approaches are presented based on testing results.

Chapter 5: Presents the conclusions and recommendations for future work.

Chapter 2 Background Knowledge

This chapter provides the background information for the material presented in this thesis. First, a survey of the existing internet traffic classification approaches is demonstrated to discuss both advantages and disadvantages. Afterwards, we display popular schemes proposed for BitTorrent and Skype traffic identification which are related to the classification scheme proposed in this thesis. In addition, a detailed description of both BitTorrent and Skype applications is presented. Finally, an introduction to Netflow and the list of P2P flow features are introduced in the end of the chapter.

2.1 Literature Survey of Traffic Classification Approaches

Recently, the network research community has been focusing on the field of internet traffic classification [2-24], by exploring different techniques, methodologies and approaches to accurately classify internet traffic. These proposed algorithms have yielded different rates of success and percentages of accuracy depending on the required traffic to be classified and the nature of the environment where these methods are applied in. The challenge lies in determining the optimal method to classify with high accuracy using minimal resources to serve different purposes such as providing quality of service for the Internet Service Providers (ISP), monitoring internet traffic, constructing different security measures, creating intrusion detection mechanisms, and applying filters to different types of internet content.

In this section, the important techniques for internet traffic classification that have been highly appraised are presented to shed light on both their advantages and disadvantages, as well as comparing them from several points of view.

The following approaches represent the top schemes and technologies used for internet traffic classification.

2.1.1 Deep Packet Inspection (DPI)

Deep Packet Inspection or DPI is a technique for classifying internet traffic by capturing raw packet data within a network and trying to distinguish between different types of traffic using different approaches such as pattern prediction, cross referencing and payload investigation.

DPI provides different types of services such as network administration, data mining, filtering and censoring. It has been proven to be a highly accurate technique for internet traffic classification as it can reveal the contents of the packets, identifying the sender, receiver, and the payload information inside the packet. In [7-16] different DPI classification approaches are suggested. In [7-13] payload signatures are extracted and compared to a database containing different signatures of different applications. In [14-15], string matching is used where the classifier searches for specific keywords in the packet and labels the traffic accordingly. In [16] the contents of the packets are revealed by parsing the data inside the packet to identify the sender, receiver, and payload information. Despite the high success rate of DPI, its three major drawbacks prevent it from being deployed.

First, DPI consumes an enormous amount of computational resources and storage capacity. When an average user is using the internet, there is a great quantity of incoming and outgoing packets. Individually inspecting these packets can be a time consuming process. Techniques such as payload signatures [7-13] and string matching [14-15] have been developed to overcome that, by only handling certain packets that can help distinguish between different types of internet traffic. However, the computational cost remains too high. If the classification procedure is performed offline, all the packets that are required to be classified must be stored on a certain media. Furthermore, packet information and payloads may have very large sizes that consume a vast amount of storage capacity to save the packets for later investigation and classification.

The second concern regarding DPI is that it may raise security, privacy and copyright issues regarding the captured packets. DPI requires the ability to read, decipher, filter, and delay internet traffic within a network, giving the network administrator or ISPs the opportunity to mettle and investigate the user's activities over the internet. This consequently raises privacy issues and concerns. Additionally, the technology may be misused if it is utilized in an improper fashion for unethical purposes such as spying, espionage, illegal privacy intrusion and other security threats.

The final issue regarding DPI is that it fails to classify P2P applications which utilize encrypted payloads for communication between hosts. Nowadays, BitTorrent clients and Skype clients have implemented encryption capabilities as a security measure which compromises the functionality of DPI approaches.

As discussed, although DPI has a very high rate of accuracy, those three important drawbacks make the technology difficult to implement and sometimes infeasible in environments with limited resources, or when payload information is absent or encrypted.

2.1.2 Port Based Classification

In this particular approach, applications are identified by checking the operating port of the specific application and referencing it to the Internet Assigned Numbers Authority (IANA) port list.

In the early days of the internet, this method was successful for classifying applications. Nowadays, this technique has become unreliable for classifying P2P applications which do not use fixed or predefined ports when they operate, disguising their traffic by using dynamic port hopping where random ports are chosen to channel the traffic each time the service is initiated. Moreover, port masquerading techniques are used where applications can channel their traffic through other well known ports such as HTTP port 80. Another drawback of this technique is that it is not sufficient to accurately classify services and applications that run over well known ports such as HTTP video streaming or HTTP flash games as they all use the same port 80.

Although port based classification does not yield accurate results for P2P, it can be used to classify traditional services and protocols such as DNS, HTTP, FTP, ICMP and some other conventional services with high accuracy since they use default port numbers that are well known and some of them are exclusive and do not allow other applications to use those ports. This method has been used by T.Karagiannis et al in BLINC [29] where they combined port

classification with host behavior to classify different types of internet traffic. However, the results from this technique could only classify traffic from a general perspective, labelling all P2P traffic as Peer to Peer without specifying which application it is. For example, CoralReef [4] is a port based internet traffic classifier provided by the Cooperative Association for Internet Data Analysis (CAIDA) and its accuracy for identifying P2P traffic is displayed in [9] by H.Kim et al where it fails to accurately classify P2P traffic, yielding variable results ranging from 12%-75% depending on the nature of the traffic, and the amount of P2P traffic within the data sets. In [28-29], the authors compared and evaluated the efficiency and accuracy of applying port based approaches to P2P traffic classification and concluded that it could not achieve more than 70 % for P2P classification. Some tools such as Snort [5] and Bro [6] use port based approaches to classify internet traffic where they can successfully classify applications that use well known ports, but give weak results when applied to P2P traffic.

In general, despite that this method was commonly used, it has been proven to be unsuccessful for classifying P2P applications that use techniques and methods to work around this primitive approach. Although port based approaches yield inconsistent results for P2P identification, it can be combined with other approaches such as host behavior to significantly increase its accuracy.

2.1.3 Machine Learning

In machine learning based techniques, classification takes place by identifying key features that separate different types of traffic where a certain machine learning algorithm is applied to classify the traffic.

Machine learning algorithms are proposed for classifying internet traffic such as flow clustering [17], Bayesian approaches [20], and support vector machines or (SVM) [21]. N. Zander et al in [22] evaluated the performance and efficiency of machine learning algorithms when applied to different types of internet traffic and compared the accuracy of different machine learning algorithms.

Machine learning algorithms use multiple identifying features for each class as a signature for classification and compares those features to a set of trained data that are labelled with each class, and determines how close or the likelihood of that testing data belonging to a certain class or label. This approach has been used in [17-24] where classification schemes are proposed using several machine learning algorithms. However, the accuracy of the results solely depends on the quality of the given data sets that the algorithms rely on for classification.

There are two main categories in machine learning approaches. First is the unsupervised machine learning, which is used in [17-21], where an unlabelled data set is fed to the algorithm and then it derives a statistical structure or a correlation between the different data items in the set to use as a reference table. Afterwards, the data that is desired to be identified is provided where it is classified based on the model created by the algorithm using the earlier data.

The other category of machine learning approaches is the supervised machine learning, which is presented in [22-24]. In this approach, a labelled training set of data is used for referencing and cross validating. Usually the labelled training set is generated in a controlled environment where the traffic is known in order to ensure high accuracy. The learning algorithm is provided

with training sets and derives a specific function or structure that can be used to distinguish between different types of classes.

The major disadvantage of this approach is that the accuracy of the results solely depends on the accuracy and the quality of the given data sets that the algorithms rely on for classification. Since different networks operate in different fashions, it is difficult to obtain a flawless absolute training set that can be used for classifying all kinds of traffic regardless of their origin.

Another disadvantage when using this technique is that when two or more classes share similar features, it becomes difficult to accurately classify the given traffic. The algorithm will tend to be biased to one of the classes as their features may overlap if there are not enough distinguishing characteristics between them.

2.1.4 Host Behavior

Host behavior approaches are concerned with identifying activities among specific hosts in a network where each application has different types of patterns. Classification models and heuristics are built by capturing the relationships between different statistical properties of flows and the applications generating them. By analyzing the traffic patterns of different applications, flows are clustered together and are labelled to the appropriate application that corresponds to the identified pattern. Various flow features are used to construct statistical signatures for classification such as packet length, packet inter arrival time and flow duration. As mentioned, BLINC [29] uses a combination of port based classification and host behavior by exploiting connection patterns and relationships between hosts and by applying different

heuristics and schemes based on those patterns. In [28-32] the authors proposed various host behavior approaches. In specific, Zhang et al [31] proposed a host behavior classifier that exploits the connection behavior of different applications and converts them to graphical and statistical representations based on flow cardinality and directions. Afterwards, these models are used for signature matching. Additionally, Karagiannis et al [28] proposed a host behavior identification scheme that relies on network and transport layer features such as IPs, ports and the number of concurrent connections.

This approach is application specific, where classification models and heuristics must be built for every application that is required to be classified. The advantage of this technique is that its accuracy is higher when compared to other general classification schemes, since this approach uses traits and features that are application specific by exploiting the uniqueness of each service. The only drawback of this method is that applications and services evolve over time, changing their behaviors, rendering the old models incapable of accurate classification. Nonetheless, this approach is the most suitable for classifying applications that share common flow features but exhibit unique statistics and connection pattern.

2.2 Approach Selection and Comparison

The objective of this thesis is to determine the most suitable approach for classifying BitTorrent and Skype activities in network traces. After studying the current approaches suggested for internet traffic classification, it was found that DPI fails to classify applications and services that utilize encrypted payloads. On the other hand, machine learning techniques are not reliable

when the features of the applications that are required to identify overlap. Moreover, port based approaches are too primitive to be applied to the current generation of P2P traffic.

The most feasible approach for identifying P2P traffic is a host behavior approach, due to the fact that it is application specific where our main goal is to identify BitTorrent and Skype activities. By analyzing both BitTorrent and Skype connection patterns, heuristics and classification schemes are demonstrated in this thesis to accurately identify them.

The classification model is constructed by integrating different flow features unique to P2P applications and the connection patterns that P2P applications exhibit whilst they are operating, in order to achieve a solid identification mechanism capable of accurately classifying the required traffic. The existing integrated approaches such as BLINC [29] and the approaches suggested in [28-32] use host behavior features that are common for all P2P services and produce low accuracy for classifying specific P2P applications. The accuracy can be increased dramatically by utilizing features that are specific for BitTorrent and Skype applications as the features used in the mentioned approaches are general P2P features and sometimes P2P services share those features with other non P2P applications. An important condition has been added for building the classification model where all the features and attributes that are used in the classification procedure must be obtainable through Netflow, in order to easily apply this classification method to any type of flow traces, and facilitate the identification process of P2P activities even with the absence of payload information or with the presence of encrypted payloads.

2.3 Approaches for Identifying BitTorrent and Skype

This section reviews some of the approaches for BitTorrent and Skype identification that are related to this research.

2.3.1 BitTorrent Approaches Comparison

Other researches [41-46] aim to identify BitTorrent traffic within network traces. DPI approaches for classifying BitTorrent traffic are suggested in [41-42] where the packets are inspected for BitTorrent keywords. However, these approaches are infeasible if the payloads are encrypted or packet information is absent. In [45-46] machine learning algorithms are applied to traffic traces to determine the existence of BitTorrent traffic by using a labelled training set. Other host behavior approaches are suggested in [43-44] where heuristics are built for extracting BitTorrent flows from network traces by monitoring the number of outgoing flows within a certain time frame. All these suggested approaches yield accurate results for identifying the traffic generated by BitTorrent clients using the standard BitTorrent protocol [33] ranging from 85%-95%. In all these approaches, the authors assume that all BitTorrent data transfers only occur via TCP, however it has been proven that newer BitTorrent clients utilize UDP for data transfer connections. Many BitTorrent clients have dropped the original BitTorrent protocol and have implemented the uTP [59] protocol pioneered by uTorrent as it has been proven to be superior due to the dynamic packet size shaping implemented in the protocol.

2.3.2 Skype Approaches Comparison

A number of researches [50-55] have been proposed to address the issue of identifying Skype traffic. In [55], Ptacek proposed a Netflow plug-in capable of identifying Skype flows which relies heavily on using a reverse DNS lookup to determine if a host contacted the Skype server or not. The accuracy of this approach relies on the result of the DNS lookup, and sometimes IP's remain ambiguous after a DNS lookup which decreases the efficiency of this approach. Chun-Ming et al in [36] were successful in reverse engineering the Skype protocol to identify the steps that a Skype client goes through during start up and during call placement. A number of different researches [50-54] aim to identify Skype traffic using machine learning algorithms, DPI approaches, and host behavior approaches. All the proposed techniques for identifying Skype traffic are based on the older versions of Skype (2004-2009) whereas the newer versions are somewhat similar but do not operate in the same manner. The proposed approaches in [50-55] may fail in identifying the new Skype clients as they do not share the same connection patterns as the older clients. In all the proposed Skype identification schemes, the authors state that the update check for any Skype client takes place using a TCP connection and a random port. However, in the newer versions of Skype (5.x.x) the update check takes place via a UDP connection using the predefined Skype port.

2. 4 P2P Applications

This section introduces both BitTorrent and Skype applications, describing the basic concepts behind their technologies and focusing on their operational behaviors inside a network.

P2P applications have emerged as a dominant portion of today's internet traffic as they provide a wide spectrum of services such as file sharing, video streaming, and creating media hubs. The focus of this thesis is on two of the most important P2P applications. BitTorrent is considered as the most popular P2P file sharing service according to various articles and studies [25-26] as it allows clients to download a file simultaneously from different peers to achieve faster file transfers and it distributes the workload of sharing the file amongst different peers. According to a study conducted in 2011 [27], Skype is by far the most popular VoIP application used by over 350 million users all around the globe, providing free subscription access to voice and video calls.

2. 4.1 BitTorrent

BitTorrent [33] is a set of protocols and services that are used mainly for massive file sharing among peers in a P2P network. Instead of downloading a file from a single server in the client-server model, it enables the user to directly download from the peers connected in the network that have the desired file. BitTorrent uses the concept of "tit for tat", where a peer downloading a file must share in uploading it, to satisfy the objective of P2P networks of pooling resources. If a host desires to download a file, a BitTorrent client must be running on the host machine. The client first downloads a tracker file which contains information regarding the nodes that have the desired file such as their IPs and ports. All the nodes that have a certain file or parts of it form a group called a "swarm" as illustrated in Figure 3, where a peer is downloading different parts of a file from multiple peers. The BitTorrent client communicates

with the tracker, which helps the BitTorrent client to download the parts of the desired file from other nodes in the swarm.

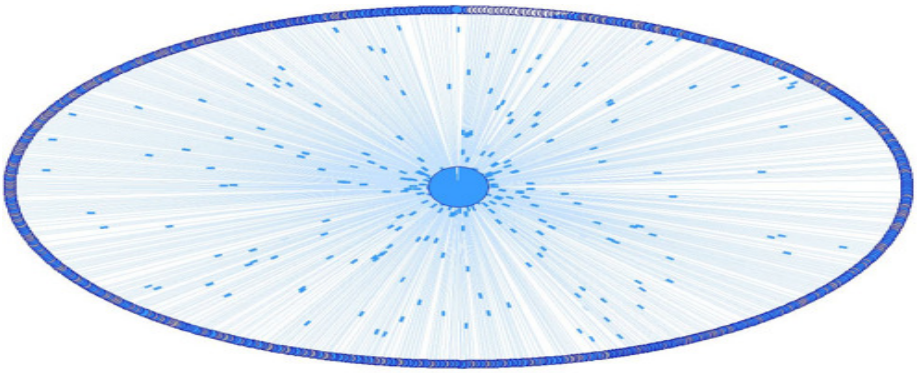

Figure 3 A Single Peer (Center) Downloading from a Swarm of Peers (Edges) in BitTorrent

BitTorrent runs above the two major known internet protocols; the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). After an extensive study of different BitTorrent clients it was found that most of them use UDP for both requests and pinging other peers, while TCP is used for data transfers. However, some clients such as uTorrent [34] use UDP for all communications between peers. This is due to the fact that uTorrent uses a UDP based micro transport protocol "uTorrent Transfer Protocol" or uTP. It was created to not meddle with open internet connection and to fully utilize the available bandwidth at the same time.

2. 4.1.1 BitTorrent Peer Location

To locate a peer, two alternative sets of mechanisms are available: tracker based vs. DHT (Distributed Hashing Table) based. In a tracker-based system, the client contacts a centrally

maintained tracker for information regarding the peers that have the desired file such as their IPs and ports. If a peer desires to download from another peer in a swarm, it contacts the peer using signalling mechanisms either in UDP or TCP.

For a DHT based system, the clients do not rely on the tracker to locate the peers within the swarm, as it implements a different technique. BitTorrent clients use DHTs for saving the contact information of different peers in order to download files without the assistance of a tracker. When DHT is enabled in a BitTorrent client, it connects to an initial bootstrap node to enter the DHT swarm. The bootstrap node provides the BitTorrent client with a set of initial peers to populate the DHT. Afterwards, the BitTorrent client queries those initial peers using UDP in order to find other peers that have the desired file. If a peer having the desired file is located, the BitTorrent client connects to the peer using TCP. Figure 4 and Figure 5 illustrate the connection pattern of BitTorrent clients for both tracker and DHT mechanisms.

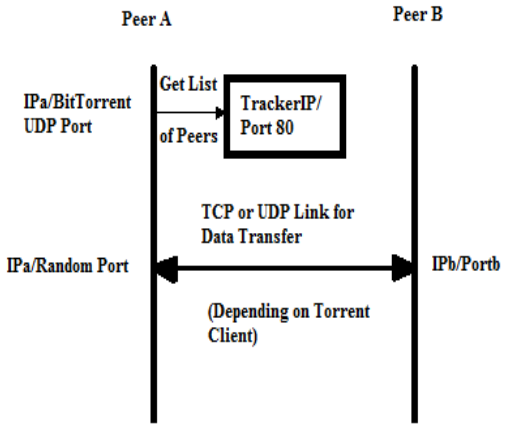

Figure 4 Peer Obtaining List of Peers from Tracker

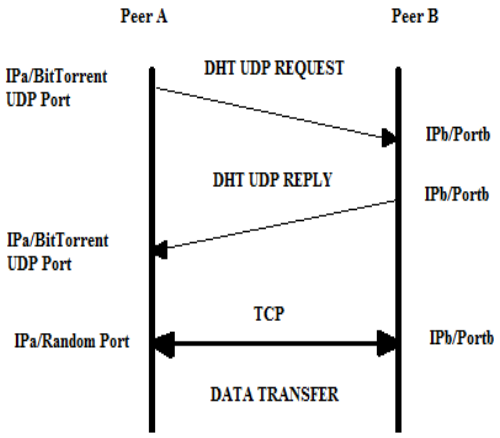

Figure 5 A Host Connecting to a Peer via DHT

2.4.2 Skype VOIP

Skype [12] is a VoIP application that utilizes different P2P elements, which enables users to perform voice and video calls between different clients running the Skype software. Its simplicity, user–friendliness, and the ability to send and receive voice and video calls among users without any subscription fees are the main reasons behind the popularity of the application. Skype offers voice and video calls with high quality when compared to the other clients such as MSN and Google Talk, providing a superior service. Another advantage that Skype has over the conventional VoIP applications is its ability to operate and function behind firewalls or NATs without setting any preferences or application privileges.

Skype implements a hybrid P2P approach where a centralized server exists, but not to manage the connection between the peers, but rather to validate the user's different stats such as the username and password. The connection between the clients in Skype takes place by using the P2P architecture and super nodes.

2.4.2.1 Start up and Login Authentication

On the start up of a Skype client, it sends out a burst of UDP packets to different super nodes until a super node replies. This process is carried out using the predefined UDP port set in the Skype client's options where the source port number must be greater than 1024. On the other hand, when a reverse DNS lookup is performed on the destination IP, the address resolves to a Skype server since these super nodes are maintained by Skype. In addition, the destination port is equal to 33033, which is an ephemeral port that is widely used by Skype super nodes. A list of all available super nodes is stored in the host's cache, which can be found in the folder where

the Skype client is installed according to the Skype operating characteristics described in [36]. On the start up of any Skype client, it must connect to one of the super nodes stored in the client's cache. If the client cannot connect in the first attempt, it tries to connect to the other super nodes in the cache. If the client cannot connect to any of the super nodes, the login phase fails, prompting an error that Skype is unable to contact the server. After a Skype client is successfully connected to a super node, the client must register with the centralized Skype authentication server to ensure a successful connection. The Skype authentication server contains a database of all the registered Skype users and their passwords and performs the authentication process in order to validate the credentials of the user.

Shortly after, the Skype client sends a request to the Skype server to determine if there is a new version of Skype and check for available updates. This request is carried over a UDP connection using the predefined Skype port on the client's side. The reverse DNS lookup of the destination IP resolves to a Skype server and the destination port is 80 or 443. Figure 6 represents the Skype startup and authenitcation process carried out by a Skype client.

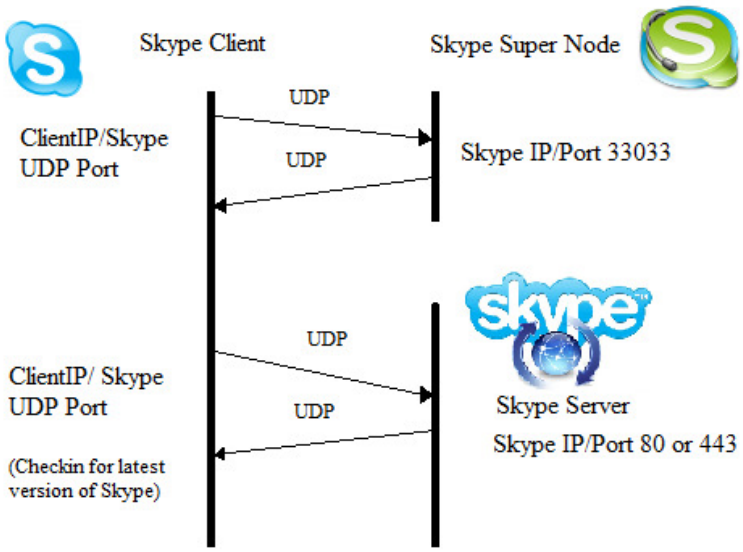

Figure 6 Connection Pattern of Skype on Start up

The connection architecture of a Skype client is illustrated in Figure 7, in which a Skype client must connect to one of the super nodes, whose IP addresses are stored in the host's cache. After the connection to a super node has been established, the Skype client must authenticate itself to the Skype authentication server to match the client's credentials, such as the username, password and available Skype credit. The process of authenticating any Skype client takes place between the Skype authentication server and the super node which the Skype client is connected to as displayed in Figure 7. The connection between the super nodes and the Skype clients represents the P2P architecture of Skype. On the other hand, the client-server segment of Skype is represented by the authentication servers that exist in the Skype architecture.

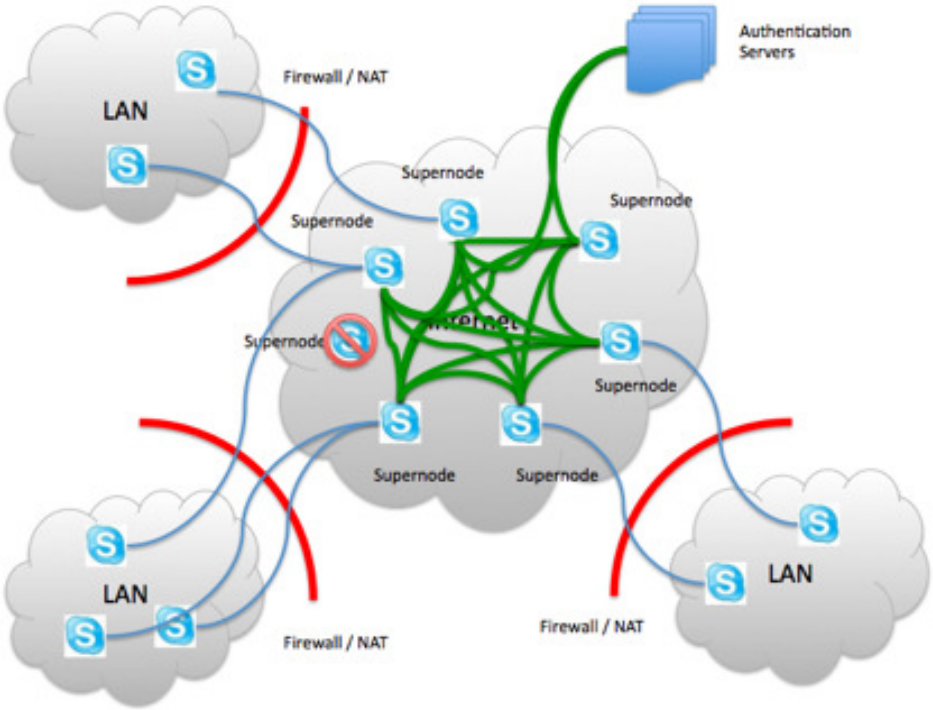

Figure 7 Skype Connection Architecture

2.4.2.2 Initiating a Call and Connection

After the Skype client has gone through the initialization processes of connecting to a super node, authenticating the user credentials with the authentication server, and checking for the latest version, the Skype client is now connected to the network and ready for call placement.

During a Skype call, the Skype client utilizes both TCP and UDP connections. Normally TCP is used for exchanging messages and signalling between the connected clients and UDP is used for transporting the payload of the voice or video call. Moreover, TCP can be used as a payload transfer protocol if UDP is unavailable due to NAT port restrictions. However, if UDP is available, it will prefer it as its transport layer protocol.

If both the clients are on hosts with public IP addresses, they establish a direct TCP connection between them for sending signalling information. Additionally, a UDP connection is established

using both of their Skype predefined port numbers to exchange voice and video packets among them.

If one of the Skype clients is behind a firewall or a restricted NAT, there cannot be any direct communication between the Skype clients to carry out the calling procedure. In this case, the Skype client initiating the call sends out UDP requests to neighboring Skype super nodes to act as a relay between both ends. The Skype clients will establish a TCP connection to the super node for message exchanging and signalling, as well as a UDP connection to the super node for the exchange of voice and video packets. The super node forwards the incoming data from both ends of the Skype call to the corresponding end, bridging the connection between both clients.

If both clients are behind a NAT, a similar process of locating a super node to relay the data between them occurs. However, the media transfer takes place via a TCP connection instead of UDP in this case.

Table 2 represents a set of Skype call flows where the receiving end is using a private IP address and a super node is needed for relaying the communication between them.

Table 1 Communication Flows between 2 Skype clients

IP Protocol	Source Address	Destination Address	Source Port	Destination Port	Packets Count	Total Packets Size	Data Speed
TCP	134.117.61.46	213.146.189.238	44585	14438	8	608	
TCP	213.146.189.238	134.117.61.46	14438	44585	8	499	
UDP	134.117.61.46	213.146.189.238	16700	14438	1,969	228,758	5.9 KB/Sec
UDP	213.146.189.238	134.117.61.46	14438	16700	1,685	224,347	4.2 KB/Sec

In Table 2, both TCP and UDP connections are used, where the UDP link carried the media data between them and the TCP connection is kept alive during the session for the exchanging of messages between both ends of the call. Based on the data recorded for a 2-minute voice call between two Skype clients in Table 1, the data speed between them for the UDP connection carrying the audio and video payload is around 5.9 Kbps. This satisfies the Skype official claim that any voice or video call consumes roughly 3-16 Kbps, as stated in [36] where the Skype call parameters are discussed.

2.5 Netflow and General P2P features

In this section, we present as set of features and characteristics which are unique for P2P applications. First, we present an introduction to Netflow showing how its functionality and the information it can provide. Furthermore, we demonstrate the selected features for P2P identification. These general P2P features are combined with other host behavior patterns of P2P applications to build the proposed classification model.

In the field of traffic classification, the challenge lies in finding a set of indentifying features and characteristics that distinguish each class to achieve a high rate of accuracy and successfully label each unknown item to a specific class. Before selecting these features, certain criteria must be established to judge these features to determine whether they will prove to be valuable signs or predictors in the classification process or not.

Many parameters have been proposed to uniquely identify P2P activity within a network such as first *n* packet sizes [31] [43], the inter arrival time between packets [50] and other features.

Some of these features are impractical for application since flow collectors such as Netflow do not support such statistics. This proves the infeasibility of using these specific traits for service providers and enterprise IT networks, who rely on Netflow traces and flow information for traffic classification.

2.5.1 Netflow

Netflow [37] is a network protocol developed by Cisco Systems that provides network administrators with access to IP flow information. Netflow has become the industry standard for network traffic monitoring and is a widely used measurement solution. Different network elements (e.g. routers and switches) collect traffic statistics and are later saved to Netflow records. The gathered IP flow information is critical for network analysts in determining the utilization of network resources and identifying network anomalies.

According to Cisco's definition [37], a flow is a unidirectional sequence of packets that share the following 7 key parameters: (a) Source IP address, (b) Destination IP address, (c) Source port number, (d) Destination port number, (e) Layer 3 protocol type,(f) Ingress interface,(g) IP type of service.

Routers and switches that support Netflow monitor the incoming and outgoing packets at a certain location in the network. Afterwards, the packets are exported in the form of flows to the Netflow collector. A flow is exported to the Netflow collector, (I) if the flow is idle where no new packets are received during a certain time frame, (II) if the exporter has detected a TCP

flag indicating that the flow has been terminated (i.e. FIN, RST flag) and, (III) if the flow is active for a period of time exceeding the active timeout period.

The collection process can be accomplished by using stand alone Netflow probes instead of routers and switches. The Netflow probes are deployed at a designated location in the network and do not interfere with the packets passing through the link. The Netflow architecture using Netflow probes is displayed in Figure 8.

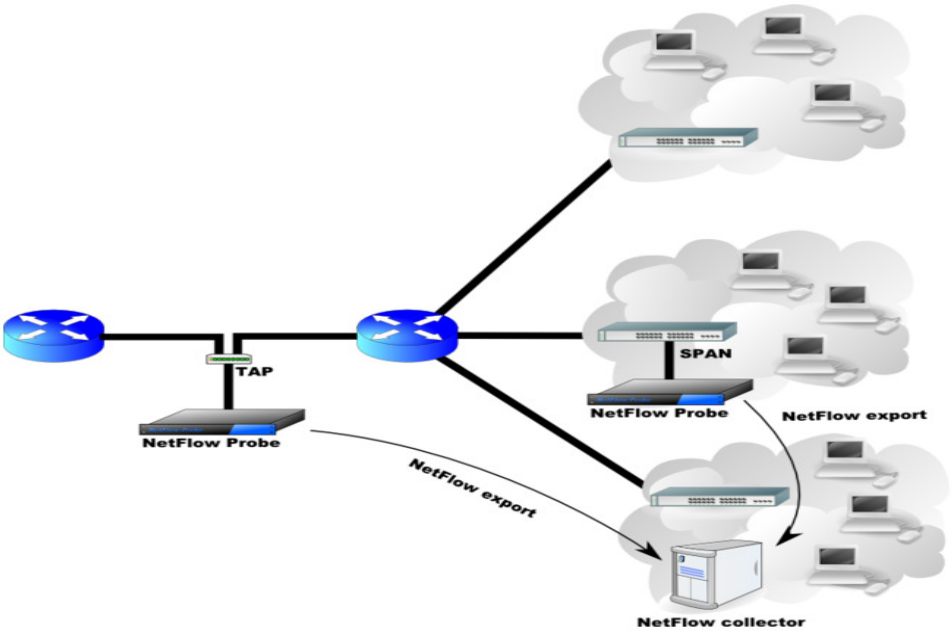

Figure 8 Netflow architecture using Netflow Probes

Each packet that is forwarded within a router or switch is inspected for the 7 key parameters mentioned earlier to determine if the packet is unique or similar to other received packets. All the packets that share the 7 key parameters are grouped together in the same flow and are stored in the cache within the Netflow collector.

According to different articles and studies [38-39], Netflow has become the standardized tool when it comes to network traffic monitoring. Besides traffic collection, Netflow offers a variety of tools that can be used for network troubleshooting, resource management, and network security.

Recently the research community has shifted in the direction of identifying different types of internet traffic using Netflow traces [32] [44] [48] as flow information is widespread and can be easily collected from routers and switches. Netflow provides detailed flow level information of the ongoing traffic without inspecting the payload of the packets flowing through the network. In addition, Netflow information is not affected by the presence of encrypted payloads, since the IP flow information is extracted from the packet's header not the encrypted payload, which makes Netflow traces obtainable even under the circumstances where the payload is absent or encrypted. According to Cisco [37], exported Netflow records can provide information regarding 104 different flow features. Some of the important features provide by Netflow are the source and destination IP/Port. The IPs/Ports can be utilized in identifying the host machines within a network through IP addresses, and the type of applications based on the port numbers. Furthermore, it can provide various statistics regarding the flows such as byte count, packet count, and duration. These statistics can be used to differentiate between flows associated with high bandwidth consumption and other flows. In addition to the 104 flow features provided by Netflow, there are 22 other fields in Netflow records that are reserved for future use by Cisco [37].

The output of a Netflow collector is a data set that contains a collection IP flows. The IETF (Internet Engineering Task Force) sought out to create a universal protocol for IP flow information and introduced the IPFIX [75] protocol. The IPFIX [75] protocol is an IETF standard for the export and collection of flow information and defines how it should be formatted. The IETF have considered different implementations of IP flow collecting technologies and elected to use Netflow v9 as a basis for creating the IPFIX [75] protocol.

2. 5.2 Features for Classifying P2P Activities

A longitudinal study was conducted using different network monitoring tools such as Wireshark [61] and NTOP [60] in order to select flow features that can be utilized to successfully identify P2P traffic. Throughout the testing phase, a number of P2P features were accounted for. However, not all of them could be used for classification since some of them are common with other non P2P applications.

The following section reviews some of the features and attributes that are used for classifying P2P activities in traffic traces.

2.5.2.1 Port Numbers

In most P2P application, a dynamic port hopping algorithm is applied where both the source and destination operate on ports with numbers greater than 1024.

The initial 1024 ports are reserved for default applications and protocols that use these ports when communicating within a network. These ports are called the “well known ports” as

operating systems and important network procedures use these ports for many important processes.

Current P2P researches [40-46] state that the majority of P2P applications utilize a number of different ports when they are operational and most of these ports fall outside the range of the well known ports. In [40-46], the authors state that in order for a flow to be classified as a P2P related flow, both source and destination ports must be greater than 1024.

Table 2 represents the percentage of P2P flows in 5 different data sets where both their destination and source port are greater than 1024.

Table 2 Percentage of P2P Flows using ports > 1024

Datasets of 1000 Peer to Peer Records	Percentage of flows where both source and destination port > 1024
Dataset 1 29rd October 2011	97.8 %
Dataset 2 5th November 2011	99.1 %
Dataset 3 23rd November 2011	98.3 %
Dataset 3 16th January 2012	98.6 %

These statistics were obtained by generating traffic in a controlled environment using different types of P2P applications such as Vuze [62], Bitcomet [63], DC++ [64] and Skype [35]. The results were validated by obtaining a list of connected peers from the P2P file sharing applications, and in the case of Skype, the traffic was generated in an isolated environment where no other applications connected to the internet.

From the previous results, it can be concluded that a vast majority of P2P applications operate on ports that are greater than 1024. This behavior can serve as a primary indicator for classifying P2P traffic.

2.5.2.2 TCP Push Flag

In a TCP packet there are two important segments: (a) the data section containing the payload of the packet, and (b) the header segment which contains different information regarding the source and destination of the packet and other details describing its nature. Within the TCP header segment, there is a field responsible for the TCP flags, which are a group of different bits. Each bit indicates a state of connection or how the receiver should handle the incoming packet. The three well known TCP flags are the SYN, ACK and FIN flags, which can be observed in any successfully established TCP connection since they are essential for the 3 way handshake implemented in TCP.

Another important flag is the PUSH flag. When a certain host sends a stream of packets to another client, the data is temporarily stored within the TCP queuing buffer. When the data inside the TCP buffer reaches a certain limit, the stream of data is pushed to the receiving side. On the other hand, when the receiving end receives any TCP packets, they are automatically stored in the incoming TCP buffer before the data is sent to the application layer. Once a segment that has the TCP push flag bit set to 1 arrives in the buffer, the cluster of packets within the TCP buffer is pushed to the corresponding application that the packets are meant to reach. Examples of the usage of the TCP flag can be seen in streaming video clips online where the data needs to be quickly processed to avoid any play back interruptions and maintain a smooth stream. The purpose of the TCP push flag is to efficiently save time by combining related segments together and processing them simultaneously, instead of individually handling each packet.

The importance of the TCP push flag and its relationship with P2P activity is discussed by M.P. Collins et al in [47], as in downloads, file transfers, VoIP and video conferencing applications, the stream of data sent from the host is large and on the receiving side the TCP buffer has a certain limit of packets it can hold. Thus, it is essential to frequently push the data to the specified application to empty the TCP buffer and guarantee the seamless flow of the packets to the application. Due to the nature of the TCP push, it is present in all the online applications listed above and can be used as a discriminating feature for P2P applications that have a large stream of data.

2.5.2.3 Incoming and Outgoing Flows

While a P2P application is active, the machine on which the application is running on sends out a large amount of UDP requests to neighboring nodes or peers, in order to establish connections between them. Although there is no exact prediction for the number of outgoing UDP probing requests that the application will transmit, the number is usually high as there is no guarantee that all the probed peers will reply. This technique is used as a safety measure to ensure that the host will connect to a number of peers, by sending out a large quantity of requests.

Usually in BitTorrent clients, the number of UDP probing requests is higher when compared to any other P2P application. This can be explained by the nature of BitTorrent applications where the downloading process is performed by acquiring different chunks and pieces of the desired file from various peers. Thus, it is necessary to be able to establish multiple simultaneous connections with other nodes to maintain the flow of data.

Figure 9 is a time chart of a captured session for 1 hour, showing the number of outgoing flows, where the host was performing different types of BitTorrent client downloading.

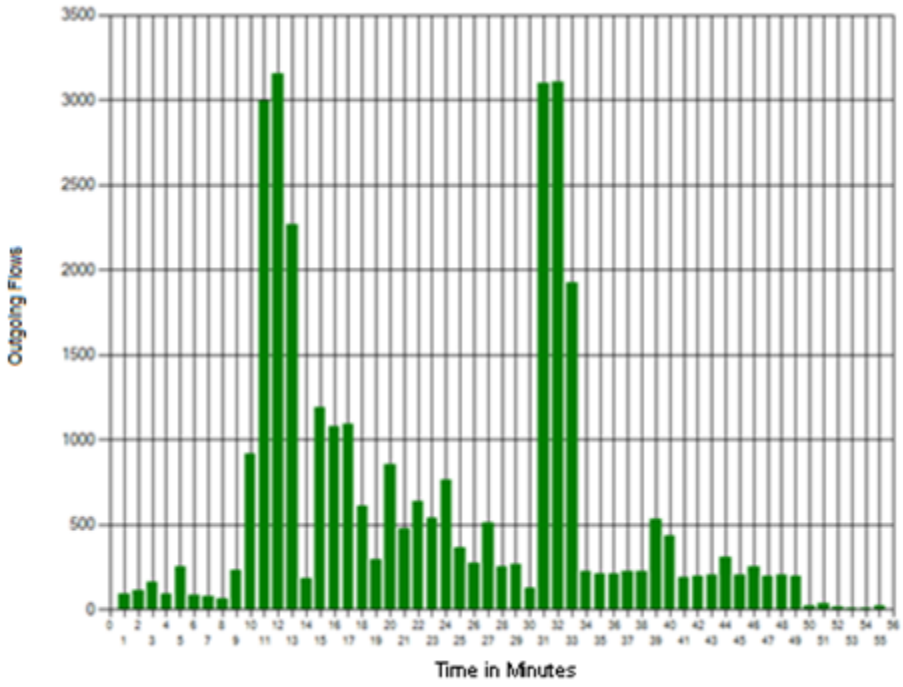

Figure 9 Number of Outgoing Flows vs. Time

As presented in the histogram, the number of outgoing flows starts at a relatively low rate for the first few minutes. However, starting from minute 10 to 13, a huge spike occurs in the number of outgoing flows, indicating the high probability of P2P application usage where the number of outgoing UDP requests tends to be high during the lifetime of the application.

Figure 10 represents the number of flows incoming to the same host for the same data set.

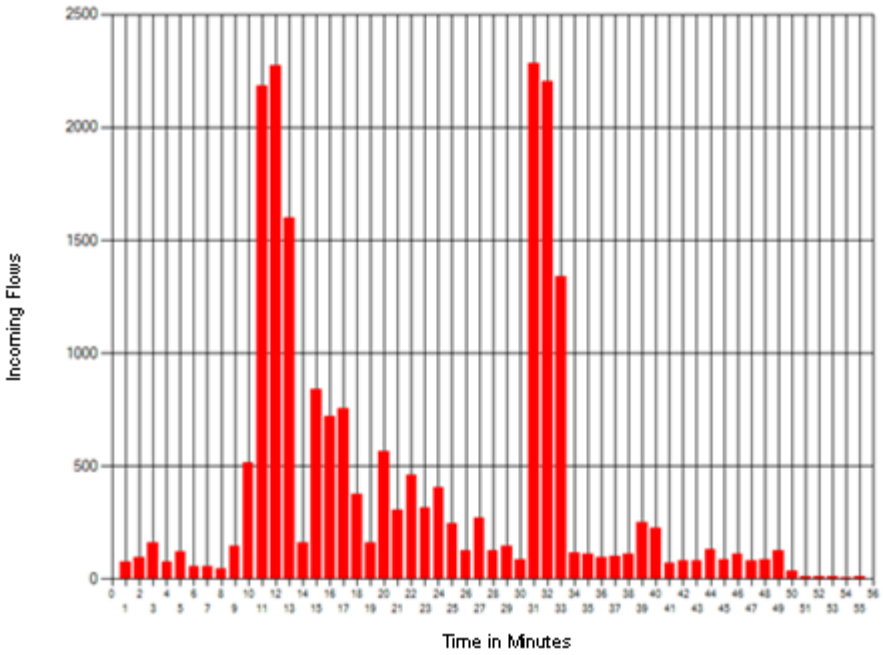

Figure 10 Number of Incoming Flows vs. Time

By observing both histograms, the number of incoming flows is relatively less in comparison with the outgoing flows. By observing the peak points in both histograms, a difference of roughly 1000 flows can be noticed. From this behavior, it can be deduced that P2P applications have a high rate of failed connections, where the client will send out a large number of requests but connects to a small number of peers when compared to the number of sent requests. This feature is used as an essential identifying attribute in the P2P classification mechanism suggested in [48] by LiJuan et al where a host is identified as a P2P participant if the number of outgoing flows divided by the number of incoming flows exceeds a certain threshold.

2.5.2.4 Maximum Transmission Unit (MTU)

Maximum Transmission Unit or the MTU of a certain protocol is the absolute limit of the size of the protocol data unit that can be sent from a source to a destination.

The MTU represents the maximum size of the packet that can be transferred over a selected medium without fragmentation or dividing the packet into multiple segments. Applications and services that require a large amount of bandwidth usually have packets that have relatively large sizes. Usually these services generate packets with massive payloads and consume a great deal of bandwidth.

This is due to that these packets are associated with massive payloads and contain more data and information. These large sized packets are produced by bandwidth consuming services such as downloading files, video streaming, P2P file sharing and online gaming. Since these applications and programs require a very high transmission capacity, network administrators and analysts pay attention to them and are highly interested in flows and traces that contain very large sized packets.

The MTU for TCP and UDP packets running on Ethernet is approximately 1500 bytes. The previously stated applications and services utilize the maximum available bytes in order to transmit their packets which contain large payloads. Ideally these applications should utilize the available MTU, but normally the maximum size that is used for data transfers is from 1300-1400 bytes since it depends on the size of information in the packet's header.

Table 3 represents the percentage of packets that have a size greater than 1300 bytes that were found in network flows. The flows were obtained in a controlled testing environment by generating internet traffic in order to validate the previous assumption. All traces and results were gathered from flows that were longer than two minutes in duration and with a size greater than 5 MBs.

Table 3 Percentage of Large Sized Packets in Different P2P Applications

Application or Service	Percentage of Packets with Size > 1300
Youtube Video	94.3%
uTorrent (P2P file sharing)	97.6%
HTTP File Download	97.2%
World of Warcraft (Online Gaming)	87.4%

The results in Table 3 indicate that all applications that require high bandwidth generally have a high percentage of packets that have a size greater than 1300 bytes due to the fact that most of the transmitted packets within those flows carry a substantial amount of data in their payloads. This behavior is typical for those types of online applications as they need to utilize the largest possible packet size to transmit their data. The MTU and packet sizes are used as key features for the proposed classification scheme serving as a threshold for identifying and separating the flows that are required to classify and small sized flows which are not of that much significance in the research as they do not consume as much bandwidth as the other flows.

Chapter 3 Classification Heuristics and Proposed Scheme

In this chapter, the process of identifying potential BitTorrent and Skype hosts in Netflow traces is introduced. A heuristic based approach has been selected due to its high accuracy for identifying different types of internet traffic as displayed in the literature survey section in Chapter 2. Heuristic based approaches give definite results and accurate classification, provided that the heuristics are based on unique connection patterns and sophisticated application specific features. On the other hand, theoretical based approaches such as machine learning do not produce definite results, since algorithms are used for classification which is not reliable if the features of the applications overlap and give inaccurate results when the characteristics are similar.

The classification procedure is a heuristic based method which is carried out using a two phase approach. The objective of the first phase is to identify the IP address and the UDP port of the host suspected of BitTorrent or Skype activity. Network flows are examined using multiple heuristics to determine the existence of BitTorrent or Skype activity. The IP address and UDP port of BitTorrent and Skype clients are identified once their activities have been detected in network flows.

The second phase is concerned with extracting all the flows associated with the detected BitTorrent or Skype client. The IP address and the UDP port obtained from the first phase are utilized in different queries in order to identify all the flows associated with the IP/Port tuple.

First, the individual heuristics obtained from analyzing both BitTorrent and Skype applications are displayed. Next, we present the classification scheme proposed to detect the flows that utilize the IP address and UDP port of a potential BitTorrent or Skype participant, which incorporates the individual heuristics and integrates them together to build the classification model. Finally, we display the heuristics used for extracting all the associated flows, once the IP address and the UDP port of a host suspected of BitTorrent or Skype activity has been identified.

3.1 Detection of Potential BitTorrent or Skype Hosts

In this section, we introduce the heuristics used to determine if a flow containing a host's IP and port are suspected of potential BitTorrent or Skype activity. The displayed heuristics are based on the connection patterns observed by monitoring and analyzing BitTorrent and Skype applications. One key constraint for our feature selection is that all our features must be available through Netflow's IP flow information. Numerous vendors other than Cisco provide an equivalent technology capable of exporting IP flow information such as Juniper's jFlow [73] and sFlow[74]. The exported IP flow information from those other technologies is similar to Netflow in terms of the statistics they can provide regarding IP flow information such as IPs, ports, byte count and packet count. This proves the feasibility of applying the classification model on any type of flow information regardless of the technology used to capture it. The following parameters are used in the classification process: (a) IPs, (b) Ports, (c) Byte count, (d) Packet count, (e) Duration. These parameters can be easily obtained from any type of IP flow information.

3.1.1 P2P Flow Identification

Since Skype and most BitTorrent clients nowadays use UDP for DHT probing, it is essential to identify the UDP port associated with the BitTorrent client and the Skype client in order to extract all the flows related to the P2P client. This is carried out by selecting all the outgoing UDP flows from a host that utilize the same source port and has sent out more than a 100 requests to different destinations. The number 100 is chosen based on experimentations and testing results displayed in Figure 11 and 12. The number of 100 outgoing requests utilizing the same UDP port has been proven to be an empirical threshold that can distinguish between P2P traffic and other types of internet traffic, as in both applications the host will periodically send out UDP requests to announce its presence in the network.

Some viruses and malicious software tend to have a large number of outgoing requests. However, several other application specific heuristics are introduced in this chapter to distinguish between the mentioned P2P applications and viral behavior.

Figure 11 and Figure 12 represent the number of outgoing flows from both BitTorrent and Skype applications on a machine for 10 different test cases where both applications are idle (no downloading for BitTorrent and no calling for Skype) for duration of 10 minutes. This helps to narrow down the ports that are suspected of P2P activity, where the host will rapidly send out a burst of UDP requests from the same source port. The result of this step returns a list of all the ports that are suspected of P2P behavior. Afterwards, it is necessary to distinguish between different P2P clients and applications, in our case BitTorrent applications and Skype clients.

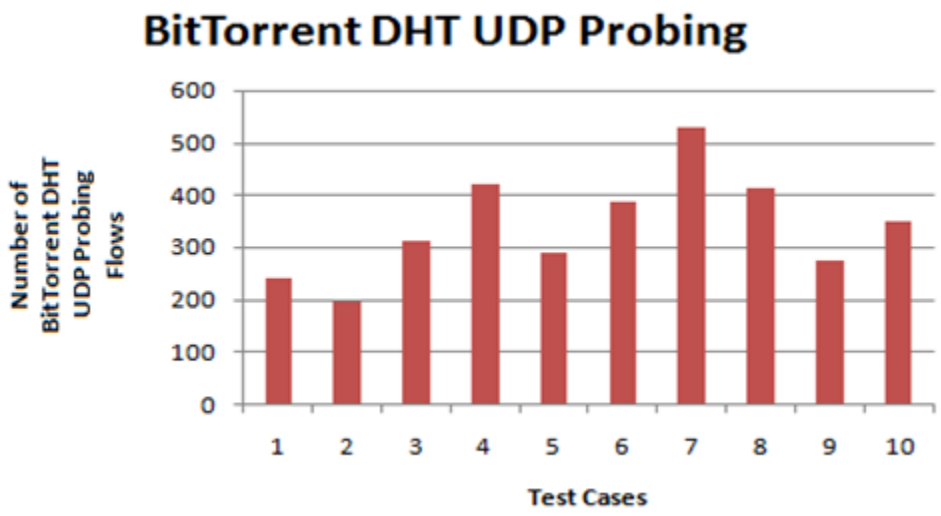

Figure 11 BitTorrent UDP Probing Flows

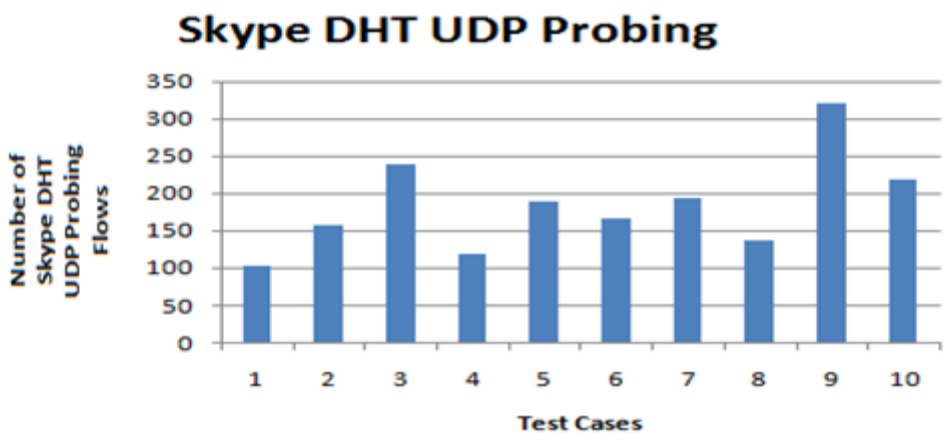

Figure 12 Skype UDP Probing Flows

3.1.1.1 Distinguishing between BitTorrent and Skype Flows

The first metric that is used to differentiate between BitTorrent and Skype is the behavior of the DHT UDP probing in both cases. After analysing the behavior of both clients the following heuristic was obtained:

The percentage of DHT UDP probing flows that have 1 packet and a size greater than 100 bytes is significantly more than the ones having a size less than 100 bytes in BitTorrent DHT UDP probing, while the opposite is true in the case of Skype DHT UDP probing.

The threshold of 100 bytes was selected based on monitoring the DHT UDP probing flows having 1 packet for duration of 12 hours and documenting the packet size distribution of Skype and BitTorrent clients for three different test cases. Table 4 represents the distribution of the packet sizes for the DHT UDP probing flows having 1 packet of Skype clients, while Table 5 represents the distribution of the packet sizes for the DHT UDP probing flows having 1 packet of BitTorrent clients.

Table 4 Packet Size Distribution for DHT UDP Probing Flows having 1 Packet of Skype

Skype DHT UDP Packet Size Distribution

	Data Set 1			Data Set 2			Data Set 3		
Packet Lengths	**Packet Count**	**Distribution Percentage**	**Threshold**	**Packet Count**	**Distribution Percentage**	**Threshold**	**Packet Count**	**Distribution Percentage**	**Threshold**
0-50	998	19 %	71 %	657	14 %	73 %	261	9 %	69%
50-75	2089	42 %		1869	43 %		1103	38 %	
75-100	545	10 %		724	16 %		638	22 %	
100-125	408	9 %	29%	621	11 %	27 %	352	12 %	31 %
150-200	590	12 %		326	7 %		445	16 %	
200-400	363	8 %		458	9 %		105	3 %	

Table 5 Packet Size Distribution for DHT UDP Probing Flows having 1 Packet of BitTorrent

BitTorrent DHT UDP Packet Size Distribution

	Data Set 1			Data Set 2			Data Set 3		
Packet Lengths	Packet Count	Distribution Percentage	Threshold	Packet Count	Distribution Percentage	Threshold	Packet Count	Distribution Percentage	Threshold
0-50	548	2 %	13 %	496	2 %	15 %	527	1 %	12 %
50-75	1921	7 %		1354	5 %		1854	6 %	
75-100	1097	4 %		1418	8 %		1336	5 %	
100-125	7684	28 %	87 %	5982	23 %	85 %	6314	22 %	88 %
150-200	9879	36 %		6534	25 %		7859	31 %	
200-400	6311	23 %		10053	38 %		9873	35 %	

The following queries are used to obtain the percentage of UDP probing flows that have 1 packet and size greater than 100 and less than 100.

Select COUNT(*) from flows where IPV4_SRC_ADDR='Host Machine' AND L4_SRC_PORT='BT_PORTS' AND PROTOCOL=17 AND IN_PKTS=1 AND IN_BYTES <100

Select COUNT(*) from flows where IPV4_SRC_ADDR='Host Machine' AND L4_SRC_PORT='SKYPE_PORTS' AND PROTOCOL=17 AND IN_PKTS=1 AND IN_BYTES >100

From the results in Table 4 , the distribution of the DHT UDP probing flows having 1 packet for Skype are heavily concentrated within the 0-100 bytes range where they represent 71%, 73% and 69% of the total distribution for each testing case respectively. From the results in Table 5, the distribution of the DHT UDP probing flows for BitTorrent is focused in the 100-400 bytes range, representing 87%, 85%, and 88% of the total distribution for each testing case respectively. From the results displayed in Table 4 and Table 5, it can be concluded that the

majority of DHT UDP probing flows having 1 packet in BitTorrent clients falls in the 100-400 bytes range. On the other hand, the majority of DHT UDP probing flows having 1 packet in Skype clients falls in the 0-100 bytes range.

The difference in the packet size distribution in each testing case is a result of different factors, such as the number of contacted peers via DHT UDP probing, the available bandwidth during the testing duration, and if the applications were idle or active; whether the host was engaged in an active download or not in case of BitTorrent clients, and if any calls were conducted or not in case of Skype clients.

According to the BitTorrent DHT protocol [49], the most frequently used queries are 'ping' and 'find node' due to the nature of BitTorrent clients since they periodically send out bursts of requests in order to connect to different peers. Both of these queries have a size greater than 100 bytes according to [49]. In Skype the most frequently used UDP requests are for pinging and probing other peers. The sizes of these queries are between 18 to 120 bytes according to [36]. Other large sized UDP requests are used during call establishment, call teardown, user searching and update checks, which do not occur as frequently as peer probing and pinging.

Since the distribution percentages of both applications do not overlap in the 0-100 bytes range and the 100-400 bytes range and are easily separated from the results in Table 4 and Table 5, the difference between them can be exploited and the packet size distribution can be utilized as a primary indicator to distinguish between the DHT UDP probing for BitTorrent and Skype.

From the previous results, Skype clients and BitTorrent clients can be identified by monitoring the DHT UDP probing flows having 1 packet in both cases. In addition, the mentioned heuristic

summarized in Figure 13 can be applied to determine which range does the DHT UDP probing distribution fall in, and label that UDP port as a potential P2P suspect.

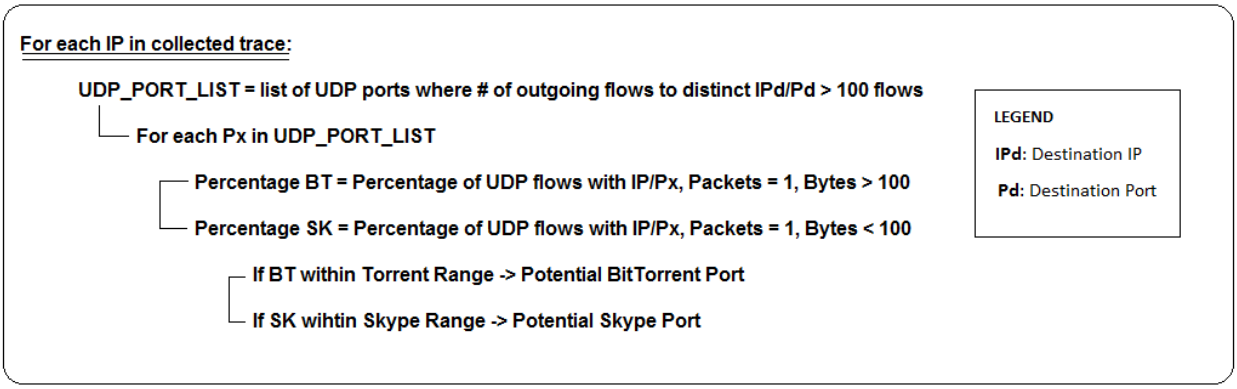

Figure 13 Probing Heuristic Summary

3.1.2 Identifying BitTorrent Flows Contacting Trackers

In order to download a file using BitTorrent, the BitTorrent client contacts all the trackers within the .torrent file using a UDP/destination port 80 connection. The request is carried out using the IP address of the machine running the BitTorrent client and the UDP listening port predefined in the BitTorrent client's settings.

Whenever a BitTorrent client initiates a download process, it acquires the list of peers that are participating in the swarm which have the desired file as illustrated in Figure 4, Chapter 2. The BitTorrent client will use the source port that is predefined in the client's options to communicate with the tracker during the initialization process. The port used to contact the tracker is the same port used by the BitTorrent client for DHT UDP probing.

The flows where the BitTorrent client contacts the tracker can be found by scanning all the flows that utilize the BitTorrent suspected ports and are requests to port 80 via UDP. Once a list

of all the IPs is obtained, a reverse DNS look up can be performed on the IPs. It can then be observed that almost all of them contain a keyword indicating torrent activity, for example “torrent, tracker, demonoid, etc.

NOTE: This may not apply to all torrents as some trackers do not contain these keywords and remain ambiguous via a reverse DNS look up. Nonetheless, this is an important feature as this process happens before the downloading occurs. Thus, the timestamp of the UDP port 80 request can be checked and validate whether it occurs before the connection to peers or not. The DNS lookup is only used for validation as it is not critical for the classification procedure. The presence of a UDP flow outgoing from the suspected BitTorrent port to a destination port equal to 80 is sufficient since most applications that utilize port 80 operate on TCP rather than UDP.

Table 6 shows the list of domains when the reverse DNS lookup was performed on one of the IP addresses that the BitTorrent client contacted via UDP and port 80. Based on the results of the DNS lookup, 17 out of 23 domain names contain tracker keywords, which indicate potential BitTorrent activity.

Table 6 Table Displaying the Result of a DNS lookup of one of the Tracker IPs

Domains	
Showing 23 out of 23 total domains	
tracker.publicbits.com	t3.pow7.com
pow7.com	t2.pow7.com
tracker.pow7.com	tracker.jamendo.com
tracker.irc.su	tv.tracker.prq.to
tracker.prq.to	tracker.publicbt.org
tracker.publicbt.com	tpb.tracker.prq.to
tracker.csze.com	tracker.ilibr.org
t1.pow7.com	tracker.token.ro
tracker1.publicbt.com	secure.pow7.com
ipv4.tracker.harry.lu	tracker.blazing.de
10.rarbg.com	dances.with.wolves.tracker.prq.to
eztv.tracker.prq.to	

3.1.3 Identifying Skype Flows Contacting the Skype Server

In the initialization process of a Skype, a Skype client (version 5 or higher) contacts a Skype super node via a UDP connection where the source port is the Skype assigned port and the destination port is 33033, and shortly after a UDP request is sent from the Skype client via the predefined Skype port to an IP address with a port number equal to 80 or 443.

During the start up of Skype, the client sends 2 requests via UDP. The purposes of these requests are to connect to a Skype super node during the initialization process and to contact the Skype server to check for the latest version of Skype. These requests are described in [56] by Adami D. et al where both these requests take place during the start up phase of any Skype client. Although in [36] Chun-Ming et al state that the checking for updates takes place via a TCP connection using a random port number, it has been found that in the newer versions of Skype (5.x.x) the update check takes place via a UDP connection using the predefined Skype port as the connection pattern mentioned in [36] is for older Skype clients (2007).

When a Skype client is launched, it initializes a connection to the IP of one of the super nodes saved in the client's host cache, and a port number equal to 33033. The reverse DNS lookup of the super node's IP returns a Skype domain as the super nodes responsible for connecting the clients on start up are maintained by the Skype server. The start up connection pattern of Skype clients can be used as an identifying feature to separate Skype activity from other P2P behavior and is illustrated in Figure 6, Chapter 2. The second part regarding the checking for updates can be observed by extracting the outgoing UDP flows that have the suspected machine's IP and Skype port and are contacting an IP with either port 80 or 443.

A list of the ports suspected of P2P activity is obtained by using the probing heuristic mentioned earlier. The UDP flow with the earliest time stamp outgoing from the suspected port is checked to determine if the flow is a request to an IP with the mentioned Skype port (33033). Afterwards, if there is a UDP flow outgoing from the suspected port to a destination port equal to 80 or 433, then the suspected port is labelled as a Skype port and any related flows are Skype flows.

The following query is used to extract all IPs within the network that contacted a Skype super node, hence having a Skype client up and running on their machine.

```
SELECT IPV4_SRC_ADDR, IPV4_DST_ADDR, FIRST_SWITCHED from flows where
L4_SRC_PORT>1024 and L4_DST_PORT=33033 and PROTOCOL=17
```

After obtaining the list of IPs suspected of contacting a Skype super node with a destination port equal to 33033, the following query can be used to determine whether the suspected host

has an outgoing UDP flow to a destination with port 80 or 443, which is the update check process of any Skype client's start up phase.

SELECT IPV4_DST_ADDR from flows where IPV4_SRC_ADDR='Host Machine' AND (L4_DST_PORT=80 and L4_DST_PORT=443) and FIRST_SWITCHED>= UDP_FIRST_SWITCH and PROTOCOL=17

For the results of both queries, a reverse DNS lookup for the destination IP can be performed to gain further validation and confirm whether this IP is associated with any of the Skype servers or not. However, the result of the reverse DNS lookup is not critical for the classification process since the mentioned connection pattern is unique for Skype clients.

NOTE: This feature is only applicable to Skype clients version 5.x.x or higher, and the Netflow collector must be up and running before the Skype client is launched on the user end to ensure capturing the Skype client's start up connection pattern.

3.1.4 Identifying BitTorrent Heavy Hitting Flows

A set of BitTorrent client flows will contain at least one TCP 'heavy hitter' where the TCP push flag is enabled , the duration of the flow is greater than 5 seconds, the incoming bytes is greater than 524288 bytes (0.5 MBs) and the average bytes per packet is greater than 800 bytes.

Since BitTorrent deals with online file transfers and downloading different chunks and pieces of the desired file from various peers, it consumes an enormous amount of bandwidth during the process. On the other hand, Skype does not require much bandwidth for singular voice or video calls. Going back to the Skype characteristics described in [36], it is observed that a one hour

voice and video call consumes around 50 MBs. After testing in a controlled environment, a Skype video and voice call consumed 47.2 MBs which is roughly equal to the range stated by Skype.

The term 'Heavy Hitter' according to A. Mahanti et al's definition in [57] is described as a flow where the consumption of bandwidth is relatively high. Since downloads require frequent packet processing as discussed in the previous chapter, the conditions for a flow to be considered a heavy hitter varies from one application to another and are determined by various features such as the presence of the TCP push flag. In addition, the duration of the flow is greater than 5 seconds, the number of incoming bytes is greater than 0.5 MB, and the average packet size is greater than approximately half the value of the MTU of both TCP and UDP (around 800 bytes per packet). The reason behind the last condition is due to the collectors setting where a maximum interval is set for each flow. If the flow is divided into multiple flows, the true value of the average packet size cannot be obtained. Another reason for reducing the threshold is that the MTU for Ethernet is 1500 bytes. However, not all links have the same MTU values as it depend on the physical media type and the MTU configuration which may be tweaked to different values. If a packet comes across a link with a MTU value less than its size, the packet is dropped and the sender is notified in order to reduce the packet size and accommodate the MTU value of the link. Furthermore, most BitTorrent clients have shifted towards implementing the uTP protocol. uTP offers a dynamic packet size shaper, where the sizes of the packets are adjusted dynamically according to the status of the link. The initial size of the packet is 1500 bytes. If the connection is slow or congested the packet size is reduced until a suitable value is reached according to the uTP specification [59]. This technique is used

in order to not clog up slow links and maintain a seamless flow of data even with a slow connection. All of these factors can result in smaller packet sizes in BitTorrent file sharing, leading to smaller bytes per packet value.

These thresholds were chosen according to the parameters set in [57].

To verify the validity of this heuristic, a P2P subset of the traffic data used in [24] by Moore et al is obtained to check for the existence of heavy hitters in P2P flows. After processing the data set, the total incoming bytes from all the P2P flows was 28825724 bytes. And by applying the heavy hitter's extraction heuristic with the specified thresholds, the total incoming bytes from the heavy hitters was 27833393 bytes which represents 96.5 % of the total P2P traffic. This indicates the huge impact of heavy hitters to the contributed bytes when compared to other non heavy hitting flows.

NOTE: The mentioned 'heavy hitter' features are only valid for the BitTorrent case, as these features are different depending on the type of application or service.

3.2 Host IP and Port Identification and Summary

The obtained heuristics are in general enough to efficiently classify whether a host and port are BitTorrent or Skype related. For example: for an unknown host, a list of suspected UDP ports is obtained by selecting flows where the protocol is UDP and the source port has sent more than 100 requests to different destinations (probing heuristic) . Afterwards, all the probing flows associated with each port are individually inspected and the percentage of 1 packet flows having size greater than 100 bytes (probing heuristics) is determined. The percentage that lies

in the BitTorrent range or the Skype range is calculated based on the test results displayed in Table 4 and Table 5. Furthermore, the first UDP probing flow that occurred (time wise) is inspected. If the destination IP resolves to a Skype server and the destination port is equal to 33033 (super node connection), and if the host has an outgoing UDP flow using the same port to an IP with a destination port equal to 80 or 433 (update check) then the port is classified as a Skype port. Otherwise, the BitTorrent heuristics are applied to determine whether the associated flows contain heavy hitters and if the host has an outgoing UDP flow from the suspected port to port 80 (communication with tracker).

Figure 14 represents a pseudo code of the proposed classification algorithm and Figure 14 displays a flowchart summarizing the proposed scheme where all the heuristics are combined and integrated together.

```
List<IPs> IPList= Gather IPs of Known Hosts in Network;

    while (IPList not Empty)
    {
    List<Ports> Suspected_ports=Get Suspected P2P Ports;
    foreach (Port P in Suspected_ports)
        {
        Percentage Per =Get_Percentage(1_PKT_FLOWS,BYTES>100,P);
        if(Per Within Skype_Range)
            {
            First_Flow=Get_First_Flow(Time,P);
            Assco_Flows=Get_All_Flows(P);

            if(First_Flow.DstPort=33033)
            {
                if(Assco_Flows.Contains(Flow(Udp,DstPort 80 || 443)))
                {
                P.Label="Skype Port";
            }   }
            }
        else if (Per Within Torrent_Range)
            {
            First_Flows=Get_First_10Flows(Time,P);
            Assco_Flows=Get_All_Flows(P);

            if(First_Flows.DstPort=80 || Assco_Flows.Contains(Heavy_Hitters))
            {

                P.Label="Torrent Port";
            }
            }
        else
            {
            P.Label="Other";
            }
        }
    }
```

Figure 14 Pseudo Code of Classification Scheme

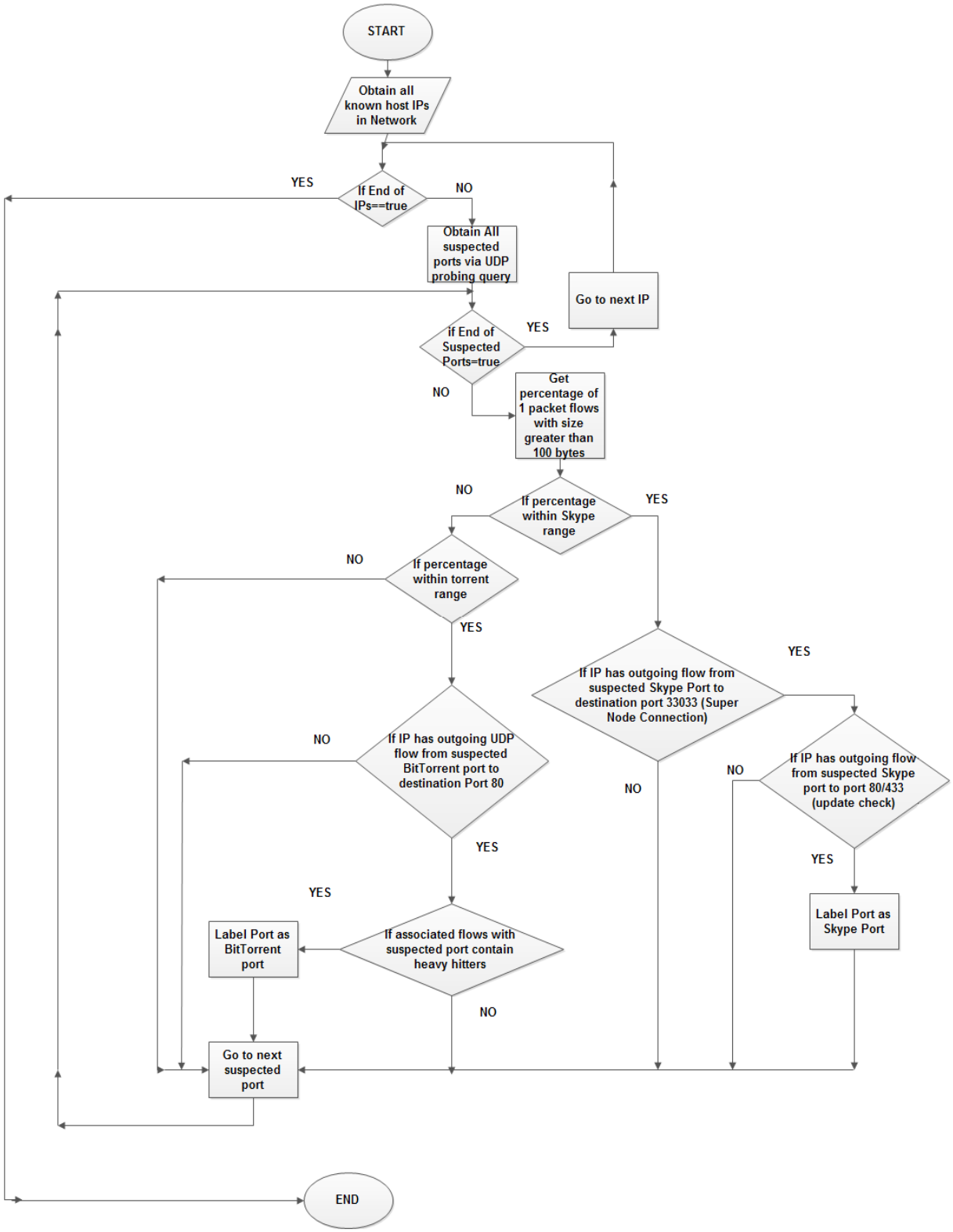

Figure 15 Flowchart of Classification Scheme

3. 3 Traffic Detection and Extraction Heuristics

In this section, we present the second phase of our classification scheme. First, we introduce the heuristics necessary for extracting all the flows associated with both BitTorrent and Skype clients. After the host's IP and port which are suspected of BitTorrent or Skype activity have been identified using the proposed classification scheme summarized in Figure 14 and Figure 15, the following heuristics are applied in order to extract all the flows that are BitTorrent or Skype related.

3.3.1 BitTorrent Traffic Heuristics and Extraction

BitTorrent traffic can be observed in three major segments: (I) Traffic from peers contacted via DHT, (II) UDP traffic from peers contacted via trackers, and (III) TCP traffic from peers contacted via trackers.

In order to contact peers, a BitTorrent file sharing clients can use (I) trackers, (II) DHTs, or (III) a combination of both mechanisms. According to a study conducted in [69] by Matteo et al, the majority of BitTorrent client users (41%) rely solely on DHTs to connect to peers, while 34% of BitTorrent client users rely on both trackers and DHTs to manage the file exchange between peers . On the other hand, only 25% of BitTorrent client users rely only on trackers to contact peers. The study by Matteo et al in [69] is based on the fact that recently law enforcement agencies have been shutting down trackers due to the distribution of copyrighted material, such as movies and music. As a result, BitTorrent clients have been forced to use DHTs in order

to locate peers. Nonetheless, there still exits a few public trackers that have not been shutdown that are still currently active and very popular among file sharers.

BitTorrent traffic can be classified according to the transport protocol utilized for data transfers. BitTorrent clients that are based on the original BitTorrent protocol [33] developed by Bram Cohen use TCP for data transfers and UDP for control and signalling. Nowadays, the majority of BitTorrent clients have shifted towards replacing the original BitTorrent protocol with uTP [59]. uTP is based on the original BitTorrent protocol and was first introduced by uTorrent [34]. uTP proved to be superior to the original protocol as it provides a vast array of options and parameters that can be tweaked to accommodate the connection speed and the nature of the network. uTP uses UDP for both the data transfers and messaging.

The following section describes each of the traffic categories and presents the necessary queries to extract them.

3.3.1.1 Incoming Traffic via DHT Contacted Peers

Nowadays, BitTorrent clients have a built in DHT option that can be enabled or disabled depending on the user`s preference. DHTs offer the BitTorrent users the ability to function and download without the help of the tracker or a .torrent file where the client can scan and find peers even if the tracker is down or dead as it helps maintain the swarm without the need of a tracker. DHT probing seeks and connects to other peers by using UDP probing to find the available neighboring peers. If they are ready for connection, a TCP link is established between them.

The following query is used to select all peers that are connected to the host, where DHT was used in order to connect to the peers rather than the traditional tracker method.

SELECT SUM(IN_BYTES) from flows where PROTOCOL=6 AND IPV4_DST_ADDR='Host Machine' AND L4_DST_PORT>1024 AND IPV4_SRC_ADDR IN (SELECT IPV4_DST_ADDR from flows where PROTOCOL=17 AND IPV4_SRC_ADDR='Host Machine 'AND (L4_SRC_PORT='SPCTD_PORTS') AND PROTOCOL=17) AND IN_BYTES>1500

The TCP flows incoming to the known host are selected where both the source and destination ports are above 1024, the source IP has been contacted via UDP probing by the suspected BitTorrent port, and the incoming bytes are greater than 1500 which is the MTU for TCP to exclude TCP connections with a small bytes count that are not data transfers. The reason behind the 1500 byte threshold is to distinguish between incomplete TCP flows that are very small and do not utilize the MTU value (1500 bytes) and the data transfers that occur via TCP that utilize the available MTU for packet transmission where the incoming bytes is greater than 1500 bytes. The method of extracting DHT connected peers is summarized in Figure 16.

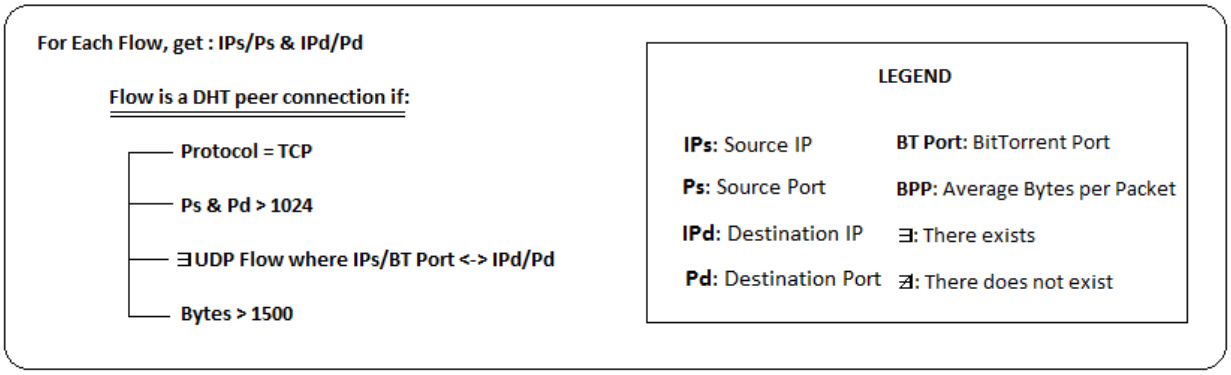

Figure 16 DHT Connected Peers Extraction

Usually the traffic from peers connected via DHT is relatively low when downloading a file from a popular tracker. If the file being downloaded is a popular file with multiple active trackers and peers engaged in the swarm, the trackers will more likely be used. On the other hand, DHT is mainly used as a fallback mechanism when the tracker is dead or not active.

3.3.1.2 Incoming Traffic from Peers via Trackers (UDP)

uTorrent [34] is by far the most popular and frequently used BitTorrent client based on measuring statistics [58] from 2011. The reason behind that is its simplicity and speed. uTorrent does not follow the standard protocols of BitTorrent file transactions as it utilizes the uTP protocol which was introduced to not interfere with existing internet connections and take advantage of the available bandwidth without slowing down any established connections. uTP uses UDP as its transfer protocol for exchanging data between peers, thus it is important to capture traffic taking place via UDP. A large number of other BitTorrent clients have started implementing the uTP protocol as it has been proven to be superior to older BitTorrent protocols. In addition, some BitTorrent clients use both TCP and UDP as a transport layer. Consequently, it is important to differentiate between DHT probing flows and data transfer flows when it comes to UDP.

The following query is used to obtain all the BitTorrent flows that use UDP as a transport protocol.

```
SELECT SUM(IN_BYTES) from flows where PROTOCOL=17 AND IPV4_DST_ADDR='Host Machine' AND l4_DST_PORT='SPCTD_PORTS' AND L4_DST_PORT>1024 AND L4_SRC_PORT>1024 AND IN_BYTES>1500
```

In this query, the UDP flows that are incoming to the known host are selected where the source port is greater than 1024 and it is incoming to the suspected BitTorrent UDP ports identified using the classification scheme displayed in Figure 15 and 16. In addition, a condition has been added to only extract the flows with a byte count greater than 1500 bytes; which is the MTU for UDP. The reason behind the last condition is to exclude the DHT UDP probing flows, which have a small byte count when compared to the UDP flows where the bulk data transfer occurs.

The method of extracting tracker connected peers using UDP as a transport layer is summarized in Figure 17.

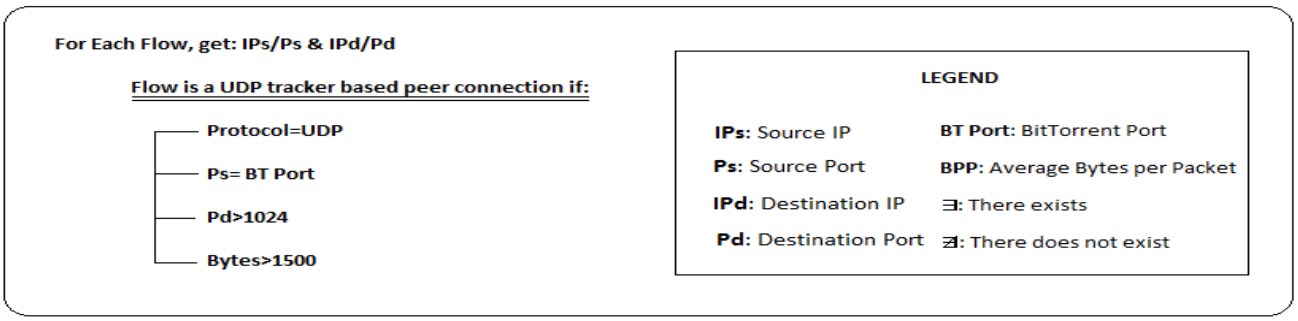

Figure 177 UDP Tracker Peers Extraction

3.3.1.3 Incoming Traffic from Peers via Trackers (TCP)

As demonstrated, peers that were contacted via DHT can be easily extracted by scanning the flows and searching if the TCP flow has a UDP similar counterpart. For the BitTorrent clients that use UDP as a transfer protocol, they can be identified by obtaining all the UDP flows that are associated with the suspected port. In both of these cases, all the associated flows for DHT and UDP peers can be traced back since they utilize the BitTorrent UDP port. The problem lies in extracting the TCP peers that were not contacted via DHT. These flows represent the peers

that were contacted via the .torrent trackers and use TCP. It can be assumed that bi-directional flows that have both source and destination ports greater than 1024 are BitTorrent related activity. However, this will result in a high probability of misclassifying other flows as BitTorrent traffic related. The main problem lies in deriving a general expression to extract all the TCP tracker flows. If the host is running other applications that utilize random port numbers on both ends, they should not be misclassified as BitTorrent participants.

To overcome this drawback, only the 'heavy hitters' from the tracker flows are selected by excluding the insignificant small flows. The phenomenon of heavy hitting flows can be observed whilst downloading a file, whether from a centralized server, or via different peers. High bandwidth consuming flows exist that have unique characteristics where the bytes per packet, total incoming bytes and duration are very high compared to other flows. The following query is used to extract heavy hitting tracker flows using TCP.

```
SELECT SUM(IN_BYTES) from flows where IPV4_DST_ADDR='Host Machine' AND
PROTOCOL=6 AND L4_SRC_PORT>1024 AND L4_DST_PORT>1024 AND
(IPV4_SRC_ADDR,L4_SRC_PORT) NOT IN

(SELECT IPV4_DST_ADDR,L4_DST_PORT from flows where PROTOCOL=17 AND
IPV4_SRC_ADDR=' Host Machine ' AND L4_SRC_PORT = 'SPCTD_PORTS')

 AND IN_BYTES>500*1024 AND DURATION>5 AND TCP_FLAGS>23 AND BPP>800
```

In these queries, the incoming flows to the suspected host are selected where the protocol is TCP; to remove peers which use UDP as a transport protocol, and both the source and destination ports are above 1024 and the destination IP/Port tuple was not contacted via UDP

to exclude the peers contacted via DHT. The condition to only extract the heavy hitting bandwidth consuming flows has been added in order to avoid misclassification and false positives by restricting only flows which have duration greater than 5 seconds, with the presence of the TCP push flag, and an average packet size greater than 800. These thresholds are based on the 'heavy hitters' heuristic mentioned earlier, where the thresholds were deduced according to the parameters set in [57] by A. Mahanti et al. The method of extracting tracker connected peers using TCP is summarized in Figure 18.

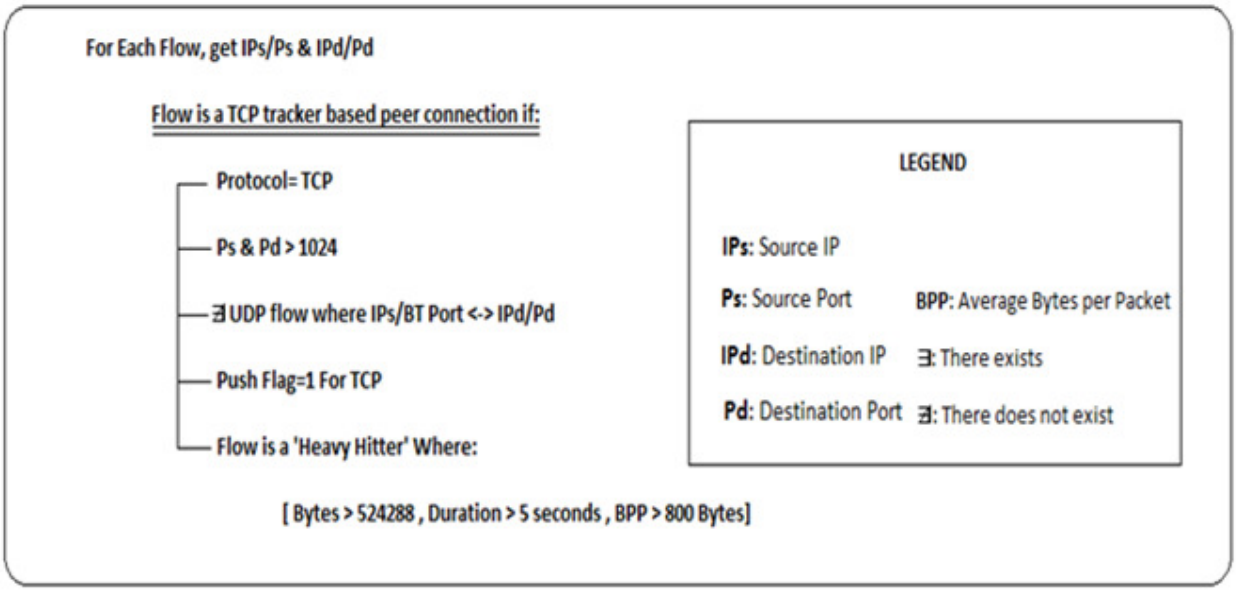

Figure 18 TCP Tracker Peers Extraction

3.3.2 Skype Traffic Heuristics and Extraction

The following section introduces the heuristics used in order to extract all the Skype related flows, after the host's IP and port which are suspected of Skype activity have been identified using the proposed classification scheme displayed in Figure 14 and Figure 15.

3.3.2.1 Extracting All Skype Flows

A Skype client does not initiate any TCP connection to any peer unless the Skype client and the corresponding peer have a UDP connection between them. If the initialization is successful, a TCP connection is established between them using the source IP and a random port on the client side and the destination IP and port of the peer, where the destination IP and port are the same as the ones used in the UDP probing process.

Skype flows can always be traced back to the specific Skype client generating them since Skype clients do not initiate any TCP connection to any peer unless the peer has been UDP probed by the Skype client using the predefined Skype port as illustrated in Figure 19 and discussed in Skype manual described in [36]. Once the Skype port is identified, all the associated Skype flows can be extracted as every TCP Skype flow has a UDP counterpart using the predefined Skype port.

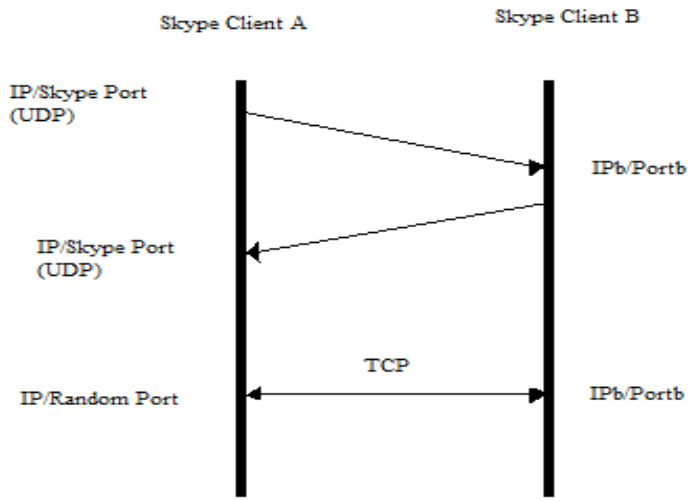

Figure 19 Skype Client Connecting to a Peer

After the host and port suspected of Skype activity have been identified using the scheme proposed in Figure 14 and Figure 15, it is essential to obtain all the flows related to the Skype client, in order to distinguish between different types of Skype activity, such as idle behavior and Skype calling. The following query can be used to obtain all the incoming flows that are associated with the Skype client.

```
SELECT * from flows where IPV4_DST_ADDR='Host Machine' AND (IPV4_SRC_ADDR,L4_SRC_PORT) in (SELECT IPV4_DST_ADDR,L4_DST_PORT from flows where IPV4_SRC_ADDR='Host Machine' AND L4_SRC_PORT=Skype_Port AND PROTOCOL=17)
```

In this query, all of the incoming flows to the identified host are selected, where the source IP/Port have been contacted by the host via a UDP connection and are using the identified Skype port to ensure capturing all of the flows that are associated with the Skype client.

A similar query is used to obtain the outgoing Skype flows by changing the destination to source and vice versa.

```
SELECT * from flows where IPV4_SRC_ADDR='Host Machine' AND (IPV4_DST_ADDR,L4_DST_PORT) in (SELECT IPV4_DST_ADDR,L4_DST_PORT from flows where IPV4_SRC_ADDR=''Host Machine' AND L4_SRC_PORT= Skype_Port AND PROTOCOL=17)
```

In this query, all the outgoing flows from the Skype client are selected by obtaining the flows where the source is the host running the Skype client and the destination IP/Port have been contacted by the host via a UDP connection and the identified Skype port.

By performing both of these queries, the flows that are associated with the Skype client are obtained. The method of extracting all the Skype flows related to a Skype host is summarized in Figure 20. The result of this query returns all the flows associated with the Skype client. These flows may fall into one of two categories: (I) Skype messaging and control flows, or (II) Skype call flows. The next section displays a method to differentiate between both types of Skype flows.

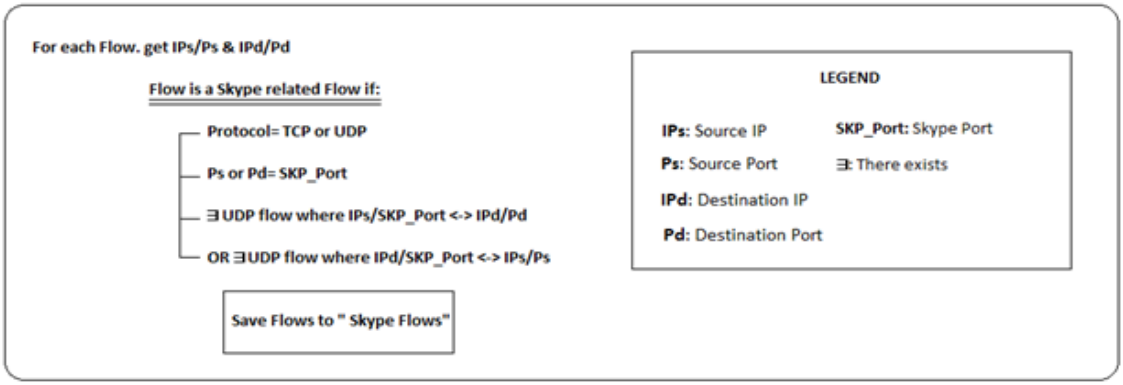

Figure 180 Skype Flows Extraction

3.3.2.2 Distinguishing Between Skype Call Flows and Skype Control Flows

Skype call flows and Skype control flows can be separated by observing different flow features such as byte count, packet count, bytes per packet and duration. In Skype call flows, these parameters are relatively higher when compared to Skype control flows due to the nature of the transferred audio and video packets in Skype call flows.

Since Skype clients strongly prefer UDP for voice and video connections, the focus is on identifying different UDP call flows.

Experimental analysis is conducted to determine thresholds and measurements to successfully distinguish between Skype probing flows and Skype voice and video flows. The first important feature was noticed in the total number of packets as it is essential for discriminating between regular DHT UDP control flows and UDP voice or video flows. A test was conducted on 10 voice and 10 video calls (10 minutes each) to determine the number of incoming packets. In the 10 test cases, the number of transferred packets were equal to or greater than 600 packets in a time frame of 30 seconds for all video and voice calls.

As mentioned on the Skype website [35], a normal voice or video call between two clients consumes around 3-16 Kbps of bandwidth throughout the full lifetime of the call. However, some call flows may have a less Kbps value as they Skype client may be connected to a slow internet connection, hence the slow data transfer resulting in a low Kbps value. The last distinguishing characteristic can be found in the bytes per packet value of Skype call flows. In the 10 mentioned audio test cases, the value of the bytes per packet varied from 90 to 140 bytes in all the Skype call flows, depending on the connection between the two Skype clients. On the other hand 10 test cases were conducted for video calls and the bytes per packet value for all the Skype call flows were in the range of 170 to 300 bytes. The values of the bytes per packet for both the voice and video test cases are displayed in Table 7 and the method of extracting the Skype voice or video calls is displayed in Figure 21.

Table 7 Average Byte per Packet for 10 Voice and Video Calls

	BPP Video Calls 10 Cases	
	1	181.4
	2	240.8
	3	231.5
	4	274.5
	5	300.2
	6	171.6
	7	299.4
	8	220.5
	9	190.2
	10	280.3
Average BPP	11	239.04

	BPP Voice Calls 10 Cases	
	1	90.6
	2	135.9
	3	115.4
	4	106.5
	5	131.2
	6	98.8
	7	120.4
	8	95.1
	9	110.3
	10	139.2
Average BPP	11	114.34

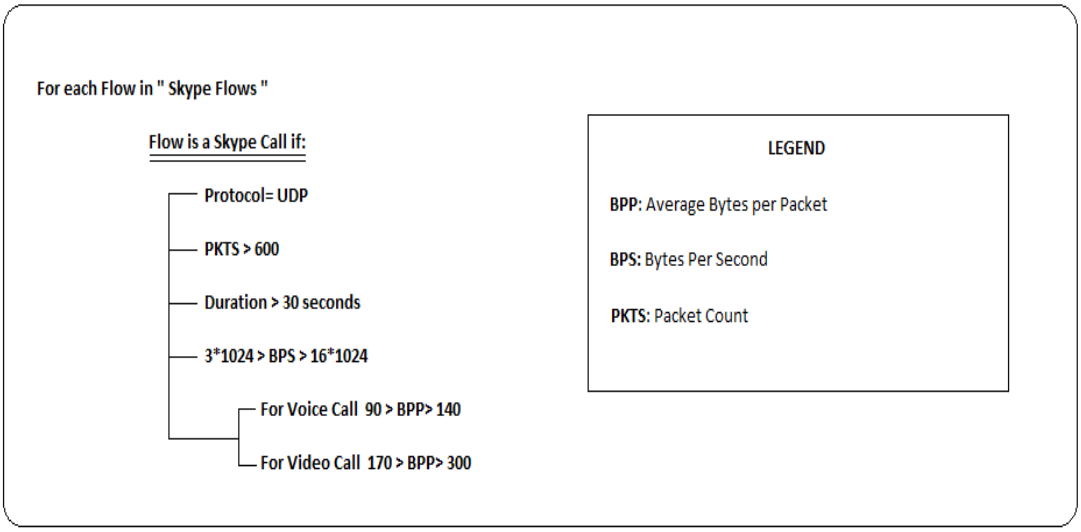

Figure 21 Skype Voice and Video Call Extraction

By applying the heuristic summarized in Figure 19, a list of all Skype call flows is obtained. The remaining flows that have not been labelled as Skype call flows are considered to be the Skype control flows that are used for messaging and signalling between peers. In Skype control flows, the value of the packet count, byte count, bytes per packet and duration are much less to those

corresponding values in Skype call flows which can be separated by using the predefined thresholds previously discussed.

3.4 Heuristics Organization, Order Impact & Contributions

As displayed in Figure 14, in order for a host and port to be classified as a Skype or BitTorrent participant, the host must satisfy as set of different conditions and parameters in a certain sequence. Individually, each heuristic is not sufficient to classify the required P2P traffic since some of these features are common between P2P applications and other services. For example, 'heavy hitters' exist in any kind of online communication that requires a lot of bandwidth such as video streaming, downloading, and online gaming. However, when the heavy hitter's heuristic is applied after confirming that a host may be a suspected BitTorrent participant, we can validate whether the host is running a BitTorrent client or not, as BitTorrent clients always have heavy hitting flows associated with them.

Our classification model is constructed by integrating the general P2P features and the application specific heuristics and organizing them in a sequential manner. If a host satisfies the classification heuristics in a specific order, it is classified as a BitTorrent or a Skype participant. The order in which the heuristics are applied is based on the sequence of the connection patterns of both BitTorrent and Skype clients.

Some of the mentioned heuristics have a major impact on the classification output, as BitTorrent clients have been shown to be one of the only applications that contact port 80 via UDP and not a regular TCP HTTP request. On the other hand, Skype clients exhibit unique

connection patterns when contacting a super node with a port number equal to 33033, and then shortly after a UDP request is sent to the Skype server to check for updates via port 80/443.

It is observed that the behavior of a BitTorrent client communicating with a tracker, and the connection pattern of a Skype client contacting the Skype server are similar. In both cases the client will send out a UDP request to a destination with port 80 or 443. Both behaviors can be separated by, (I) checking if the suspected port has an outgoing UDP flow to port 33033; which is unique for Skype super nodes connections or, (II) checking if the suspected port has 'heavy hitting' associated flows; which occurs in BitTorrent clients and not in Skype clients.

Some of these heuristics can be considered as general features that most P2P applications have in common. This includes excessive UDP probing where different P2P applications utilize this technique and are not sufficient to be applied on their own for classification. These features must be combined with stronger heuristics based on the connection patterns of both applications to obtain solid results with higher accuracy. By combining general P2P features with application specific heuristics, the accuracy is increased and the reliability of the classification model is enhanced.

Chapter 4 Testing and Experimentation

In this chapter, the results of applying the proposed classification scheme to a collection of network traces are presented. The traces are obtained from different sources, where a large number of hosts and applications were concurrently active to test the reliability of the proposed approach.

First we display the accuracy, false positive values, false negative values and the results of applying the proposed scheme to five different data sets where the targeted traffic to identify is BitTorrent traffic. Afterwards, the results and the limitations are briefly discussed.

Next we present the accuracy, false positive values, false negative values and the results of applying the proposed scheme to the same five data sets when the targeted traffic to identify is Skype traffic. Afterwards, we demonstrate the results and the limitations of applying the proposed scheme.

Finally we compare our approach with other existing approaches by applying them to the same data sets. The results will show that our approach outperforms all other selected approaches.

4.1 Data Sets Description

In order to validate and measure the efficiency of the proposed approach, it must be applied to real life examples to see how the classification apparatus will perform and to obtain concrete results regarding the accuracy and the performance.

Five data sets are used to test the accuracy of the obtained heuristics. First is a database containing Netflow records (175,943 flows, 1.37 GBs) that were acquired for duration of approximately one hour from 3 pm to 4 pm from an industrial office where multiple hosts were using the internet performing their daily routines and a known host was performing various P2P activities throughout the duration of the capturing; mostly BitTorrent downloading using 5 different BitTorrent file sharing clients which are: Bitcomet [63], Azureus[62], uTorrent [34], uTorrent Portable [34], and BitTorrent [65]. In addition, a Skype call was conducted where 6 minutes was a voice call and 5 minutes was a video call.

The second data set was acquired from a computer in a lab in Carleton University (1900345 flows, 5.6 GBs), where a known host was connected to the campus network. The duration of the session is 3 hours, captured from 11 am to 1 pm. Throughout the duration of the captured session, a BitTorrent file sharing client, a p2p radio streaming application, and a Skype client were active. In addition, a well known Trojan called 'SUS/UnkPacker' [73] was active but within a sandbox application in order to limit its threat within a quarantined zone. The SUS/UnkPacker's characteristics are similar to P2P applications where it sends out UDP requests to the attacker using random port numbers. The information was captured using Wireshark and flow information was created by exporting the packet information to flows. Wireshark has the capability of aggregating the captured packets to produce flow information. The flow information obtained from Wireshark is identical to Netflow in terms of the information it can provide such as IPs, ports, byte count, packet count and protocol.

The third data set was captured using Wireshark [61] from a host machine connected to the Carleton University residential network (2398789 flows, 6.8 GBs). The duration of the session is 4 hours captured from 12pm to 4pm. Throughout the duration of the captured session a BitTorrent file sharing client, and a Skype client were active. Additionally, 2 P2P T.V streaming applications were active, which are Sopcast [70] and PPStream [72]. Flow information was produced by converted the captured packets into flow data using Wireshark.

The fourth data set is an 8 hour long (16.3 GBs, 6587323 flows) session captured from from 10 am to 6 pm in the Systems & Computer Engineering Lab in Carleton University where multiple users were connected to the network, generating different types of internet traffic and one known machine was downloading files using two BitTorrent file sharing clients; uTorrent [34] & Vuze [62]. Also a 45 minute Skype video call was conducted during the session. The session was captured using NTOP [60]; an open source network probe. NTOP [60] is network traffic probe that is capable of monitoring network usage. NTOP [60] works by putting the NIC of a host in promiscuous mode in order to monitor all types of traffic on the interface and saves it as a TCP dump file. If the hosts on the network are connected to a common router, NTOP [60] will be capable of sniffing the incoming and outgoing traffic of all the hosts connected to the router. Flow information was generated for this session by converting the captured TCP dump file obtained from NTOP [60] to flow information using Wireshark [61].

The fifth data set is a 6 hour long (14.9 GBs, 5234974 flows) session captured with NTOP [60] from the Carleton University Residence network where multiple users were engaging in their daily internet activities and a known host was downloading files with two popular BitTorrent

file sharing clients which are Bitcomet [63] and BitTorrent [65]. Also a 20 minute Skype voice call was conducted during the captured session. Flow information was generated for this session by converting the captured TCP dump file obtained from NTOP to flow information using Wireshark.

The 5 data sets were collected from several locations under varying conditions during different times of the day to obtain a variety of testing sets. Due to hardware limitations, 2 of the testing sets are flow information of 1 host. However, in those cases, other applications were active in order to challenge the accuracy of the proposed scheme, as in data set 2 a P2P radio streaming application and a Trojan were active, while in data set 3, two P2P T.V. streaming applications were active. The other data sets contain flow information of multiple hosts within the network, where in each data set different internet applications were active and the traffic was captured from all the hosts connected to the network. Each data set was collected from a different location to ensure capturing different data sets under different conditions.

Detailed information regarding the 5 data sets is displayed in Table 8 showing the number of hosts, number of flows, traffic nature, and active applications for each data set.

Table 8 Data Sets Description

	Data Set 1 (8 Host, 1 Hour)		Data Set 2 (1 Host,3 hours)		Data Set 3 (1 Host, 4 Hours)		Data Set 4 (16 Hosts, 8 Hours)		Data Set 5 (22 Hosts, 6 Hours)	
	Bytes	**Flows**	**Bytes**	**Flows**	**Bytes**	**Flows**	**Bytes**	**Flows**	**Bytes**	**Flows**
Total	1.8GBs	17943	5.6GBs	1900345	6.8GBs	2398789	16.3GBs	6587323	14.9GBs	5234974
TCP	1.2GBs	12564	4.7GBs	1634295	3.94GBs	1391294	11.5GBs	4696761	10.2GBs	3612132
UDP	0.59GBs	5379	0.76GBs	266048	2.84GBs	1007491	4.7GBs	1910323	4.61GBs	1622841
Traffic on Known Ports	0.41GBs	4126	4.3GBs	1459193	5.1GBs	1799091	5.8GBs	2371436	4.17GBs	1465792
Traffic on Unknown Ports	1.37GBs	13619	1.2GBs	407216	1.6GBs	564420	10.4GBs	4215886	10.728	3769181
Applications Running	BitTorrent, Skype, HTTP, HTTPS FTP,SNMP, NetBIOS, SMTP		BitTorrent, Skype, Virus, P2P radio, HTTP, HTTPS, SNMP, NetBIOS		BitTorrent, Skype, Sopcast, PPStream, HTTP Video Streaming, HTTPS, SMTP, NetBIOS		BitTorrent, Skype, HTTP, HTTPS, FTP, SNMP, SMTP, NetBIOS, DNS, Bootps		BitTorrent, Skype, HTTP, HTTP Video Streaming, FTP, SMTP, Bootps, NetBIOS, IMAP, HTTPS	
BitTorrent Traffic	1.3GBs 1 Host		1.85GBs 1 Host		0.94GBs 1 Host		9.8 GBs & 0.48 GBs 2 Hosts		9.18 GBs 1 Host	
Skype Calls	6 mins voice call, 5 mins video call		12 mins, 8 mins, 9 mins voice calls		20 mins video call		45 mins video call		20 mins voice call	

4.2 BitTorrent Traffic Detection

In this section, the false positives, false negatives, and accuracy values for BitTorrent identification are defined. Afterwards, we demonstrate the results of applying the proposed classification scheme on each of the 5 data sets, when the targeted traffic to identify is BitTorrent Traffic. In the end of this section, the results of the classification process and the limitations regarding using this approach are briefly discussed.

The classification model's performance is evaluated by using the following byte-wise metrics: (I) Bytes false positives, (II) Bytes false negatives, (III) Bytes accuracy. Since the flows are not labelled and due to the absence payload information in all the data sets, there is no way to calculate the packet and flow accuracy for the testing sets. Regardless of the flow and packet accuracy, the byte-wise metrics provide a more useful measure for BitTorrent identification since it can provide information regarding the falsely classified bytes. The importance of the byte-wise metrics in BitTorrent classification is displayed by Gossett et al in [44] where they are mainly concerned with evaluating their BitTorrent identification apparatus in terms of bytes. In BitTorrent classification the most important goals are, (I) Identifying the hosts suspected of BitTorrent activity which is carried out using the proposed scheme and (II) Calculating the amount of downloaded bytes via BitTorrent clients for bandwidth management and resource monitoring. In addition, byte-wise metrics can be utilized in identifying and controlling congestion by observing the links that have a high volume of packets with a large byte count which have a high probability of causing congestion.

Byte wise false positives and negatives can be utilized in determining the model's probability in classifying BitTorrent related bytes as non BitTorrent and vice versa which in turn can be used for evaluating different models to determine the probabilities of misclassification. In addition, the byte wise accuracy can be utilized in determining the percentage of accurately classified BitTorrent bytes against the misclassified BitTorrent bytes and calculating its success ratio.

4.2.1 BitTorrent Traffic False Positives and False Negatives

The false positives probability is the model's probability of wrongly classifying non BitTorrent related bytes as BitTorrent related bytes. The falsely classified bytes are represented by the bytes generated by flows utilizing port numbers greater than 1024 and are generated by any application other than BitTorrent.

For the traffic generated by tracker UDP connected peers and DHT connected peers, the false positive values are equal to 0 In both cases, since the bytes belong to flows that can be traced back to their origins, whether they have a UDP counterpart utilizing the DHT UDP port for DHT transfers, or if the file transfer occurs via UDP using the BitTorrent predefined port. The bytes from the tracker TCP connected peers cannot be traced back as they do not utilize the BitTorrent UDP port. Thus, there is a probability of falsely classifying the generated traffic. Figure 22 represents the taxonomy of traffic on ports greater than 1024.

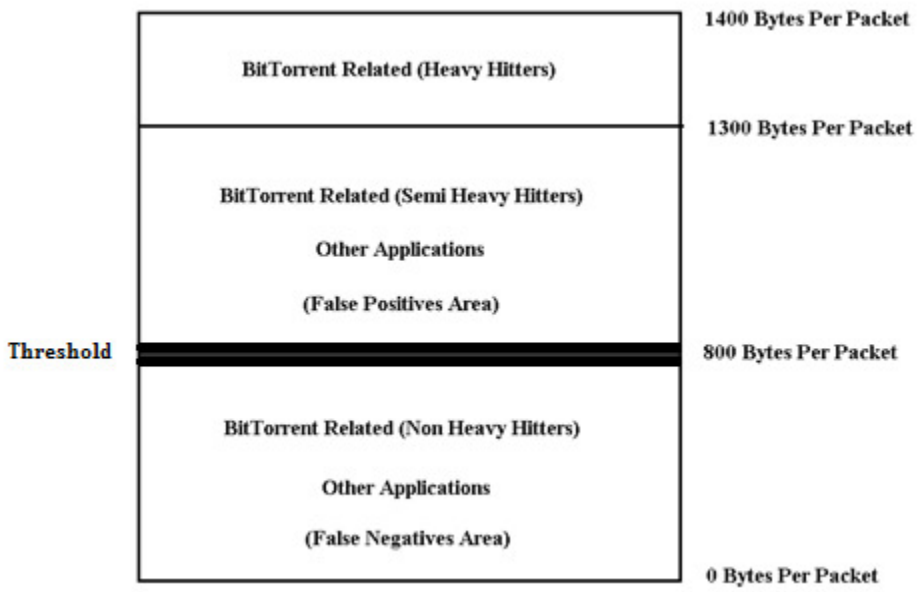

Figure 19 TCP Traffic on Ports greater than 1024

From Figure 22, we can observe that the bytes generated from flows in the range of 1300 to 1400 bytes per packet are the BitTorrent heavy hitters utilizing the maximum available MTU. BitTorrent applications are one of the only types of applications that have heavy hitters utilizing ports greater than 1024 on both the source and destination ends. Other applications that have heavy hitting flows utilizing ports greater than 1024 such as online gaming and Xbox Live connections are listed as official ports on the IANA port list. Additionally, some P2P video streaming applications such as Sopcast [70] and T.V.ants [71] use ports greater than 1024 but utilize smaller bytes per packet values ranging from 200-1200 bytes, according to an analysis conducted in [66] by Silverston et al. The flows within the 1300-1400 bytes per packet range are guaranteed to be BitTorrent related and do not contain any false positives or false negatives.

As displayed in Figure 22, the bytes generated from flows falling within the range of 800-1300 bytes per packet may belong to other applications using ports greater than 1024 or BitTorrent

traffic which are considered as ‘semi heavy hitters’ as they consume a moderate amount of bandwidth but not as much as the ‘heavy hitters’ within the upper limit of 1300-1400 bytes per packet. The false positive bytes fall in the range of 800-1300 bytes where a non BitTorrent related bytes may be wrongly classified as BitTorrent related since other traffic may fall in the same range. Initially, the threshold of the bytes per packet for the heavy hitters was set to BPP > 800 bytes to accommodate different path MTUs and uTP packet shaping as discussed in detail in the previous chapter. The false positives values can be extracted by adjusting the heavy hitter’s byte per packet threshold from BPP > 800 to 800 < BPP < 1300 to extract the bytes generated by other applications that may be misclassified as BitTorrent related bytes. Although the range of 800 to 1300 bytes may contain BitTorrent related bytes, we shall assume that all the bytes in that range are non BitTorrent related, in order to extract the false positives. Since payload data is absent and the flows are not labelled, this is the only feasible approach to obtain an estimate of the false positives values even though some of these bytes may be BitTorrent related.

The false negatives probability is the model’s probability of classifying BitTorrent related bytes as non BitTorrent related bytes. These falsely classified bytes are represented by the bytes from flows utilizing ports greater than 1024 and are BitTorrent related bytes that have not been classified as such.

The bytes in the lower limit in Figure 22 that fall within the 0-800 bytes per packet range are generated by other applications using ports greater than 1024 and the BitTorrent non heavy hitters. The false negative bytes fall in this range where BitTorrent related bytes may be

misclassified as a non BitTorrent related bytes. These bytes are generated from early aborted or non responsive peer connections. The contribution of these bytes to the total downloaded bytes via BitTorrent clients is very low as they are not considered heavy hitters. In all of the 5 testing cases the value of the false negatives within the 0-800 bytes per packet range represented 0.06 % of the total traffic in bytes, which can be neglected due to its insignificant value. For example, if a value of 1000 MBs has been classified as BitTorrent related bytes, then 940 MBs are actual BitTorrent related bytes and 60 MBs are misclassified non BitTorrent related bytes.

Nonetheless, a significant portion of the false negative bytes is represented by the difference between the bytes classified as BitTorrent related and ground truth total of BitTorrent bytes, and should be taken into consideration during testing.

Equations 1 and 2 summarize the values of the false positive probability and the false negative probabilities.

$$\alpha = \frac{x}{T} \quad (1)$$

$$\beta = \frac{y}{T} \quad (2)$$

Where T is equal to the total number of download bytes via BitTorrent, x is equal to the non BitTorrent bytes classified as BitTorrent bytes and y is equal to the BitTorrent bytes classified as non BitTorrent bytes. While α and β are equal to the false positive probability and the false negative probability respectively.

4.2.2 BitTorrent Traffic Accuracy Definition

Throughout the testing duration for all the data sets, the total amount of downloaded bytes via BitTorrent clients was documented from each of the BitTorrent clients' statistics page; which contains the exact number of downloaded bytes throughout the BitTorrent client's session. A sample of a BitTorrent client's statistics page is displayed in Figure 23 where the number of downloaded bytes is displayed. This number serves as the ground truth when calculating the accuracy of this approach. After each testing case, the total number of downloaded bytes is recorded by documenting the total number of downloaded bytes from each of the BitTorrent clients' statistics page.

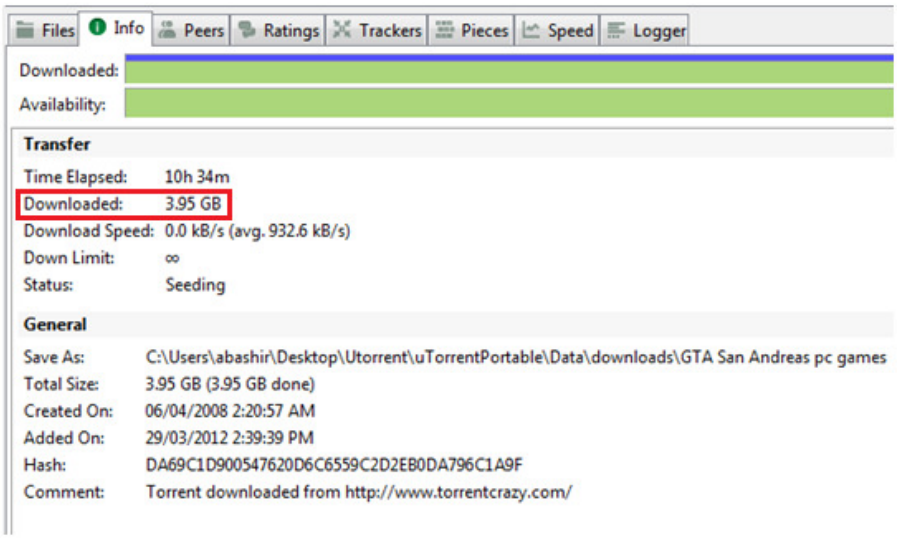

Figure 23 BitTorrent Client's Statistics Page

The accuracy is defined as the ground truth value obtained from the BitTorrent client's statistics page minus the values of the false classified bytes. The accuracy equation is displayed in Equation 3.

$$\pi = \frac{T-(x+y)}{T} \quad (3)$$

Where T is equal to the total number of download bytes via BitTorrent, x is equal to the non BitTorrent bytes classified as BitTorrent bytes, y is equal to the BitTorrent bytes classified as non BitTorrent bytes, and π denotes the byte wise accuracy.

4.2.3 BitTorrent Byte-Wise Classification and Results

After applying the classification scheme proposed in the previous chapter to extract the host's IP and port suspected of BitTorrent activity, it is necessary to calculate the total amount of consumed bytes for each BitTorrent host in each testing set. This is carried out by applying the BitTorrent traffic queries summarized in Figure 24 and calculating the accuracy of this approach.

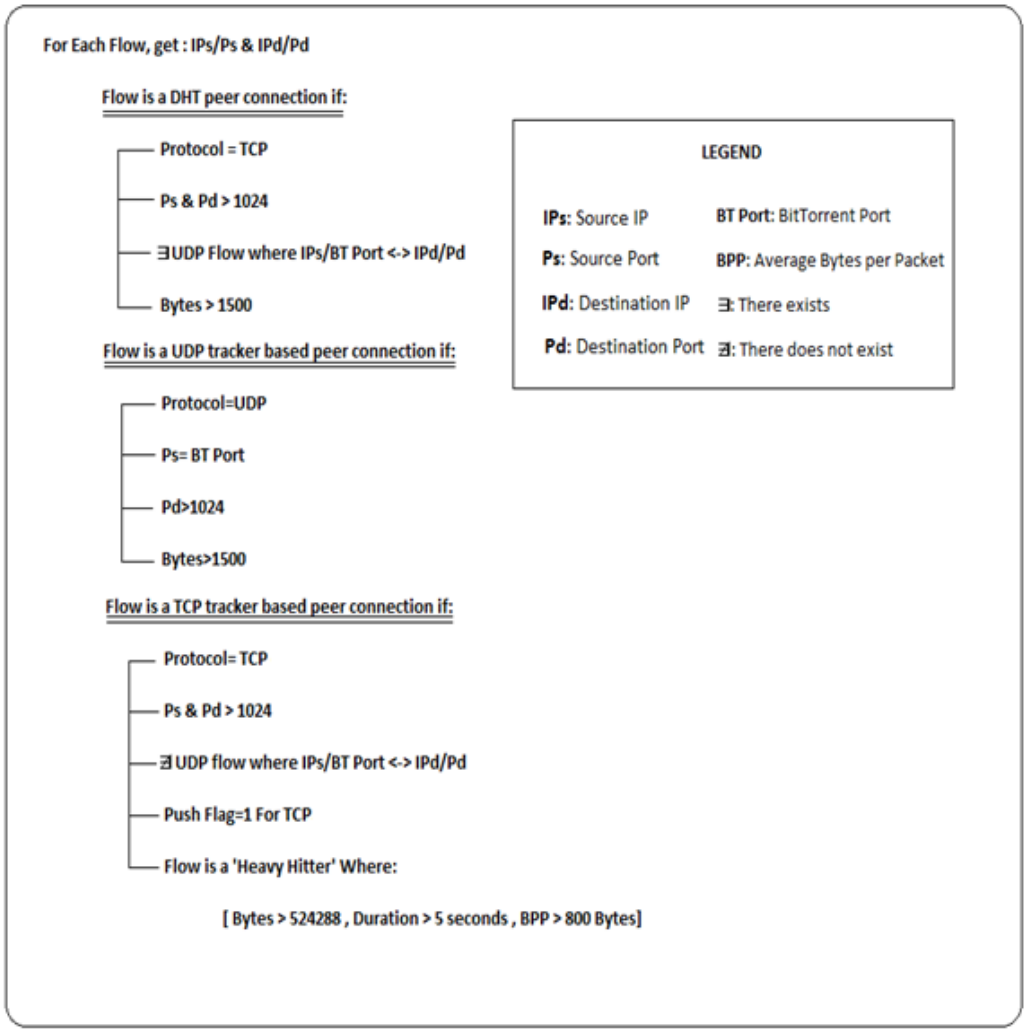

Figure 24 BitTorrent Traffic Extraction Queries

4.2.3.1 Data Set 1

The host classification approach proposed in the previous chapter was successful in extracting the suspected hosts and ports in data set 1 and the results are displayed in Table 9. In this data set, 5 BitTorrent clients were active, and the proposed scheme was successful in extracting the IPs and ports of each of the 5 BitTorrent clients, which were all running on the same machine.

Table 9 Suspected BitTorrent Hosts and Ports in Data Set 1

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing flow to Port 80 (torrent contact)	Associated heavy hitting flows	Classification
192.168.1.138	52799	✓	✗	✓	✓	BitTorrent
192.168.1.138	60891	✗	✓	✗	✗	Other
192.168.1.138	42176	✓	✗	✓	✓	BitTorrent
192.168.1.138	12474	✓	✗	✓	✓	BitTorrent
192.168.1.138	16677	✓	✗	✓	✓	BitTorrent
192.168.1.138	35854	✓	✗	✓	✓	BitTorrent

The results of the BitTorrent traffic queries (Figure 24) for data set 1 are displayed in Table 10. By applying the BitTorrent traffic queries, a value of 96.7 % of the total BitTorrent traffic is obtained. By using the false positive and false negative equations summarized in Equation 1 and Equation 2, a value of 1.6% and 3.3% is obtained respectively. By calculating the accuracy using the equation in Equation 3, a value of 95.1% is obtained.

Table 10 BitTorrent Classification Results and Accuracy for Dataset 1

BitTorrent Traffic	Value in Bytes	Number of Packets	Number of Flows
Traffic from Peers via Trackers (TCP)	667152351	844742	5486
Traffic from Peers via Trackers (UDP)	454646947	654298	2531
Traffic from Peers via DHT	233169458	324895	954
Total Incoming Traffic	1354968756	1823935	8971
Total Incoming Traffic (Ground Truth)	1400897536		
Captured Percentage	***96.7%***		

4.2.3.2 Data Set 2

In the second data set the ports generating the P2P traffic was unknown, and the proposed scheme was successful in identifying the BitTorrent port and the other suspected ports. In this data set, one BitTorrent client was active and its port number was extracted. Additionally, 3 other ports were suspected of P2P activity, which represent the Skype client, the P2P radio application, and the Trojan activity which were all active during the duration of this data set. The hosts and ports suspected of P2P activities are displayed in Table 11. By monitoring the Trojan's activity, it can be observed that the Trojan will utilize a set of random UDP ports to communicate with the attacker. The byte count and the packet count of the UDP flows generated by the Trojan were very small and its contribution to the total amount of downloaded bytes can be neglected.

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP Flow to Port 80 (Tracker connection)	Associated heavy hitting flows	Classification
134.117.61.54	19654	✗	✓	✗	✗	Other
134.117.61.54	58965	✓	✗	✓	✓	BitTorrent
134.117.61.54	3654	✓	✗	✗	✗	Other
134.117.61.54	12994	✓	✗	✗	✓	Other

Table 11 Suspected BitTorrent Hosts and Ports in Data Set 2

The results of the BitTorrent traffic queries for data set 2 are displayed in Table 12. By applying the BitTorrent traffic queries in Figure 24, a value of 94.7% of the total BitTorrent traffic is

obtained. By using the false positive and false negative equations summarized in Equation 1 and Equation 2, a value of 2.9% and 5.3% is obtained respectively. By calculating the accuracy using the equation in Equation 3, a value of 91.8% is obtained.

The value of the accuracy is relatively less than the accuracy obtained from the other data sets due to the presence of the P2P radio application which generates heavy hitting flows utilizing ports greater than 1024. The value of the false positives in this data set are represented by the heavy hitting flows generated by the P2P radio application that were misclassified as BitTorrent related flows. The average bytes per packet of the heavy hitting flows generated by the P2P radio application was in the range of 600-1000 bytes which do not contribute to the total downloaded bytes as much as the heavy hitters generated by the BitTorrent client which fall within the 1300-1400 bytes range. In this data set the accuracy was affected due to the presence of a P2P radio streaming application which shares similar features with BitTorrent clients. Nevertheless, the proposed scheme was able to identify the BitTorrent traffic with 91.8% accuracy.

Table 12 BitTorrent Classification Results and Accuracy for Dataset 2

BitTorrent Traffic	**Value in Bytes**	**Number of Packets**	**Number of Flows**
Traffic from Peers via Trackers (TCP)	1119879123	767032	149658
Traffic from Peers via Trackers (UDP)	339738608	249768	89542
Traffic from Peers via DHT	427819248	289032	98537
Total Incoming Traffic	1887436979	1305832	337737
Total Incoming Traffic (Ground Truth)	1993069671		
Captured Percentage	***94.7 %***		

4.2.3.3 Data Set 3

In data set 3, the host's IP generating the P2P traffic was known, but the associated P2P ports were not known. The proposed scheme was successful in identifying the ports suspected of P2P activity and are displayed in Table 13. The port associated with the BitTorrent client was identified as well as 3 other ports suspected of P2P activity. These 3 ports represent the operating ports of Skype client and the 2 P2P T.V. streaming applications.

Table 13 Suspected BitTorrent Hosts and Ports in Data Set 3

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP Flow to Port 80 (Tracker connection)	Associated heavy hitting flows	Classification
192.168.8.138	4456	✓	✗	✗	✓	Other
192.168.8.138	36589	✓	✗	✓	✓	BitTorrent
192.168.8.138	25541	✗	✓	✗	✗	Other
192.168.8.138	8668	✓	✗	✗	✓	Other

The results of the BitTorrent traffic queries (Figure 24) for data set 3 are displayed in Table 14. By applying the BitTorrent traffic queries, 96.7% of the captured traffic was accounted for. By using the false positives and false negatives equations summarized in Equation 1 and Equation 2, a value of 5.6% and 3.1% is obtained respectively. By calculating the accuracy using the equation summarized in Equation 3, a value of 91.3% is obtained.

The value of the accuracy in this data set is the lowest value when compared to the other data sets. This is due to the presence of 2 P2P T.V. streaming applications, which resulted in relatively high false positives values, leading to a reduced accuracy value. The value of the false positives in this data set are represented by the heavy hitting flows generated by the P2P T.V. streaming applications that were misclassified as BitTorrent related flows. The average bytes per packet for the heavy hitting flows generated by the P2P T.V. streaming applications are in the 800-1200 bytes range which falls in the range stated in [66] where the authors discuss the parameters of flows generated by P2P T.V. streaming applications. The heavy hitting flows generated from the P2P T.V. streaming applications represented 5.6% of the total traffic, while the heavy hitting flows generated by the BitTorrent client represented 42.5% of the total traffic. From this behavior, it can be concluded that the heavy hitters generated from BitTorrent clients have a much larger contribution to the total downloaded bytes when compared to the heavy hitters generated from the P2P T.V. streaming applications.

Table 14 BitTorrent Classification Results and Accuracy for Dataset 3

BitTorrent Traffic	**Value in Bytes**	**Number of Packets**	**Number of Flows**
Traffic from Peers via Trackers (TCP)	492405843	364732	10254
Traffic from Peers via Trackers (UDP)	265262361	176825	4965
Traffic from Peers via DHT	228467836	160884	3659
Total Incoming Traffic	986136040	702441	18878
Total Incoming Traffic (Ground Truth)	1019215872		
Captured Percentage	***96.7 %***		

4.2.3.4 Data Set 4

For the fourth data set collected in the Systems & Computer Engineering lab in Carleton University, a table of all hosts suspected of BitTorrent activity within the network is built, by

applying the scheme proposed in the previous chapter. Aside from the known machine generating P2P traffic, several other hosts were identified as potential P2P suspects in the network. In this data set a known host was running 2 BitTorrent clients where the IPs and ports were identified and highlighted in Table 15. Another machine running a BitTorrent client was also identified and was recorded in Table 15. The last record in Table 15 represents a host which was suspected of P2P activities but could not be verified as a BitTorrent client as the host and port did not match the criteria set in the proposed scheme.

Table 15 Suspected BitTorrent Hosts and Ports in Data Set 4

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing flow to Port 80 (torrent contact)	Associated heavy hitting flows	Classification
134.117.61.46	61256	✓	✗	✓	✓	BitTorrent
134.117.61.46	16700	✗	✓	✗	✗	Skype
134.117.61.46	19880	✓	✗	✓	✓	BitTorrent
135.178.111.101	57362	✓	✗	✓	✓	BitTorrent
125.149.116.103	57362	✗	✓	✗	✗	Other

Table 16 represents the results of the BitTorrent traffic queries when performed on the known host in data set 4. By applying the BitTorrent traffic queries in Figure 24, a value of 96 % of the total BitTorrent traffic was accounted for. By using the false positive and false negative equations summarized in Equation 1 and Equation 2, a value of 1.2% and 4% is obtained

respectively. By calculating the accuracy using the equation in Figure 24, a value of 94.8% is obtained.

During the session a total of 9.8 GBs was downloaded via BitTorrent clients, and by using the proposed technique, 9.45 GBs were successfully accounted for and labelled as BitTorrent related traffic. In addition, approximately 116 MBs (1.2%) were falsely classified as BitTorrent related activity, which can be overlooked as it represents a very small value when comparing the sizes of both the accuracy and the false positive value.

Table 16 BitTorrent Classification Results and Accuracy for Known Host in Dataset 4

BitTorrent Traffic	Value in Bytes	Number of Packets	Number of Flows
Traffic from Peers via Trackers (TCP)	5770313728	7212892	758965
Traffic from Peers via Trackers (UDP)	3860856832	4289840	312458
Traffic from Peers via DHT	519045120	5863639	395162
Total Incoming Traffic	10150215680	17366371	1466585
Total Incoming Traffic (Ground Truth)	10565619548		
Captured Percentage	***96%***		

Table 17 represents the results of the BitTorrent classification queries (Figure 24) and the accuracy when applied on the unknown suspected host in data set 4. In this case the ground truth of total downloaded bytes could not be obtained as the machine running the BitTorrent client was unknown. The number used for comparing the accuracy is the total number of bytes downloaded using unregistered ports greater than 1024. This approach is used by Gossett et al in [44] where the authors assume that any flow utilizing ports greater than 1024 on both ends is BitTorrent related, and they use the total amount of downloaded bytes on ports greater than 1024 as the ground truth for calculating the accuracy of their BitTorrent identification apparatus. Although this number is not 100 % accurate, it is somewhat close to the ground truth. If a host is downloading files via a BitTorrent client, the traffic on the unregistered ports

greater than 1024 will be approximately equal to the true amount of downloaded bytes via the BitTorrent client, since BitTorrent clients are one of the only applications that have heavy hitters utilizing ports greater than 1024 on both ends. The method presented in this thesis of obtaining the total downloaded bytes via the BitTorrent clients' statistics page serves as an exact value for testing the accuracy. However, in this case the host generating the BitTorrent traffic was unknown so this method could not be applied.

Table 17 BitTorrent Classification Results and Accuracy for Unknown Host in Dataset 4

BitTorrent Traffic	Value in Bytes	Number of Packets	Number of Flows
Traffic from Peers via Trackers (TCP)	343303681	361371	9658
Traffic from Peers via Trackers (UDP)	1429400	16812	684
Traffic from Peers via DHT	158966874	187019	5293
Total Incoming Traffic	503699955	565202	15635
Total Incoming Traffic (Ground Truth)	518547898		
Captured Percentage	***97.1%***		

By applying the BitTorrent traffic queries in Figure 24, a value of 97.1 % of the total BitTorrent traffic is obtained. By using the false positive and false negative equations summarized in Equation 1 and Equation 2, values of 1.4% and 3.9% respectively are obtained. By calculating the accuracy using the equation in Equation 3, a value of 94.7% is obtained.

4.2.3.5 Data Set 5

In the fifth data set collected from the Carleton University Residence network the IPs and ports suspected of BitTorrent activity were extracted by applying the classification scheme proposed in the previous chapter, and are displayed in Table 18. In this data set a host was running 2 BitTorrent clients and the proposed classification scheme was successful in extracting both the IPs and ports of the host running the BitTorrent clients.

Table 18 Suspected BitTorrent Hosts and Ports in Data Set 5

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing flow to Port 80 (torrent contact)	Associated heavy hitting flows	Classification
172.19.18.175	7892	✓	✗	✓	✓	BitTorrent
172.19.18.175	1935	✗	✗	✗	✓	Other
172.19.18.175	44485	✓	✗	✓	✓	BitTorrent
172.19.18.175	9936	✗	✓	✗	✗	Other

Although other hosts were suspected as P2P hosts, they could not be verified as they did not meet the classification criteria. For most cases the hosts have more than 100 outgoing UDP flows to different destinations, but could not be further validated as they do not exhibit the basic BitTorrent and Skype connection patterns mentioned earlier. This can be interpreted as one of two scenarios: (I) both the host and the port are P2P participants, but the application was idle (no calling for Skype and no downloading for BitTorrent), (II) or the host and port are other P2P applications aside from BitTorrent and Skype which also use DHT UDP probing.

The results of the BitTorrent traffic queries (Figure 24) for data set 5 are displayed in Table 19. Initially in this set, the result of the captured traffic was 108 % of the BitTorrent traffic, which indicates the existence of a large number of false positives. As mentioned, in this data set a video streaming web page was open. Unlike traditional HTTP video, this application was using port 1935 which is a port used by Adobe flash for video streaming and is highlighted in Table 18. After excluding this port from the BitTorrent traffic queries, a value of 96.3 % of the total

BitTorrent traffic was accounted for. By using the false positive and false negative equations summarized in Equation 1 and Equation 2, values of 0.9% and 3.7% respectively are obtained. By calculating the accuracy using the equation in Equation 3, a value of 95.4% is obtained. Port numbers which are greater than 1024 and are registered to other applications should be taken into consideration to avoid misclassification and false labelling.

In this session a total of 9.53 GBs was downloaded via BitTorrent clients, where 9.18 GBs could be accounted for via the BitTorrent traffic extraction queries, and around 84.1 MBs (0.9 %) were falsely classified as BitTorrent related bytes. The latter number is very small when compared to the total downloaded bytes.

Table 1 BitTorrent Classification Results and Accuracy for Known Host in Dataset 5

BitTorrent Traffic	Value in Bytes	Number of Packets	Number of Flows
Traffic from Peers via Trackers (TCP)	5330402671	6663003	521489
Traffic from Peers via Trackers (UDP)	2244892520	2363043	325458
Traffic from Peers via DHT	2284458287	3263511	304781
Total Incoming Traffic	9859753478	12289557	1151728
Total Incoming Traffic (Ground Truth)	10241771557		
Captured Percentage	***96.3 %***		

4.3 BitTorrent Traffic Detection Summary and Limitations

As shown, the accuracy of this approach can be calculated by comparing the total amount of downloaded bytes obtained from the BitTorrent clients' statistics page for each data set, and the total number of bytes from the BitTorrent traffic queries (Figure 22). For the 5 different test cases, the proposed technique yielded accurate results ranging from 91.3%-95.4%. The only limitation of using this approach is that the BitTorrent client downloading the files must support DHT networking, where the client periodically sends out UDP requests to neighboring nodes. If

the DHT option is disabled from the client's options, the client will not send out UDP requests. However, this does not prevent the proposed scheme from classifying, since the BitTorrent client still receives DHT UDP probing requests but will not be able to reply to them when DHT probing is disabled. The problem lies when this approach is applied to the previous generation of BitTorrent clients that do not support DHT networking, as the proposed technique will not be able to identify them since they function in a different fashion and solely rely on trackers to contact peers. Due to the rapid progression of BitTorrent technology, almost every available BitTorrent client today supports DHT networking, which strengthens the feasibility of this approach. Furthermore, the capturing of the session must begin before the user commences any download via the BitTorrent clients to ensure the capturing of the connection patterns that BitTorrent clients exhibit before the downloading phase.

4.4 Skype Traffic Detection

In this section, the false positives, false negatives, and accuracy values for Skype identification are defined. Afterwards, the results of applying the proposed classification scheme on each of the 5 data sets are demonstrated when the targeted traffic to identify is Skype Traffic. At the end of this section, the results of the classification process and the limitations regarding using this approach are briefly discussed.

4.4.1 Skype Flow Evaluation Metrics

The byte wise false positive probability is the model's probability of misclassifying non Skype related bytes as Skype related bytes. While the byte wise false negative probability is the models probability of misclassifying Skype related bytes as non Skype related bytes.

On the other hand, the flow wise false positive probability is the model's probability of misclassifying non Skype related flows as Skype related flows. While the flow wise false negative probability is the models probability of misclassifying Skype related flows as non Skype related flows.

After analyzing the behavior and nature of Skype clients, It can be observed that a Skype client does not initiate any kind of TCP connection with different peers unless there is a UDP bidirectional counterpart that contacts the same end point as stated in the Skype connection pattern heuristic earlier. This characteristic has been verified and tested for all Skype flows, concluding that in order for Skype to establish a connection to any peer, there must be a TCP and UDP connection to the same IP but using different. Based on the connection patterns that Skype clients exhibit while connecting to different end points, all flows and bytes can be traced back to the suspected host and Skype port, resulting in zero false positives and false negatives in both flows and bytes, as we can deduce whether a flow and it's generated bytes is Skype related or not by checking if the flow follows the Skype connection pattern where each TCP flow has a UDP counterpart using the predefined Skype port. Equation 4 and Equation 5 displays the Skype false positives, false negatives and accuracy for both bytes and flows.

$$\pi_1 = \frac{T_1 - (x_1 + y_1)}{T_1} \quad (4)$$

$$\pi_2 = \frac{T_2 - (x_2 + y_2)}{T_2} \quad (5)$$

Where T_1 is equal to the total number of Skype bytes, x_1 is equal to the non Skype bytes classified as Skype bytes, y_1 is equal to the Skype bytes classified as non Skype bytes, and π_1 denotes the byte wise accuracy. On the other hand, T_2 is equal to the total number of Skype flows, x_2 is equal to the non Skype flows classified as Skype flows, y_2 is equal to the Skype flows classified as non Skype flows, and π_2 denotes the flows wise accuracy. Both π_1 and π_2 are equal to 100 % if the Skype host's IP/Port have been successfully identified.

As displayed in equation 4 and Equation 5, we can easily extract all Skype related flows and bytes once the IP/Port of the Skype host have been identified. Consequently, byte wise metrics and flow wise metrics will not provide valuable information regarding the performance of the model since the byte wise and flow wise accuracy are 100%, provided that the IP/Port are successfully identified.

After each data set, the host's IP and Skype port were documented so they can be utilized in determining whether the classification scheme was successful in identifying the Skype host's IP/Port. If the identified IP/Port matches the documented Skype IP/Port, then the classification model was successful, and the Skype host is labelled as 'Host Detected'.

Unlike BitTorrent clients, the consumption of bandwidth of in Skype clients is relatively low, resulting in the insignificance of performing byte-wise evaluations. Other related Skype

identification models [50-55] are not concerned with byte wise and flow wise metrics as they do not give accurate evaluation of their models since they are 100% if the IP/Port is identified. The main focus of the recent Skype detection models is on identifying Skype hosts IP/Port and distinguishing between Skype call flows and Skype control flows. These metrics can provide information about whether the model is reliable in indentifying different types of Skype hosts in different networks under various conditions such as network topology and the usage of proxies. In addition, these metrics can provide information about whether the model is consistently capable of identifying different kinds of Skype call flows (audio and video) and if it is able to accurately extract them even if they are conducted under different conditions such as connections with limited bandwidth and congested links.

The model is evaluated by 2 factors: (I) the success of the model in identifying the Skype host's IP/Port, (II) and the success in differentiating between Skype call flows and Skype control flows.

The identification of Skype host's IP/Port and the separation of Skype call flows and Skype control flows provides network analysts with valuable information regarding any conducted Skype call, such as the duration, the nature of the call (audio or video), the number of super nodes used to relay the connection, and the geographic location of both ends of the calls by using their IPs. By Identifying Skype hosts and Skype call flows we can prioritize the host's connection in order to improve the call quality and avoid lag and dropped frame rates in Skype calls. In addition, network administrators can identify different Skype users within their networks by referencing the identified IP/Port.

For each data set, a table is presented displaying whether the IP/Port of the Skype client have been identified. In addition, a table is presented displaying the flows labeled as Skype call flows and their flow features such as byte count, packet count and duration.

4.4.2 Skype Call Duration Detection

The objectives of the Skype call identification scheme proposed are, (I) to determine the existence of Skype voice call flows or Skype video call flows, and (II) to obtain a rough estimate of the duration of any placed Skype call.

Following each testing case, the duration of the conducted Skype calls was documented by using the Skype application's timer (Figure 25) to serve as the ground truth for calculating the accuracy of determining the duration of the conducted Skype calls.

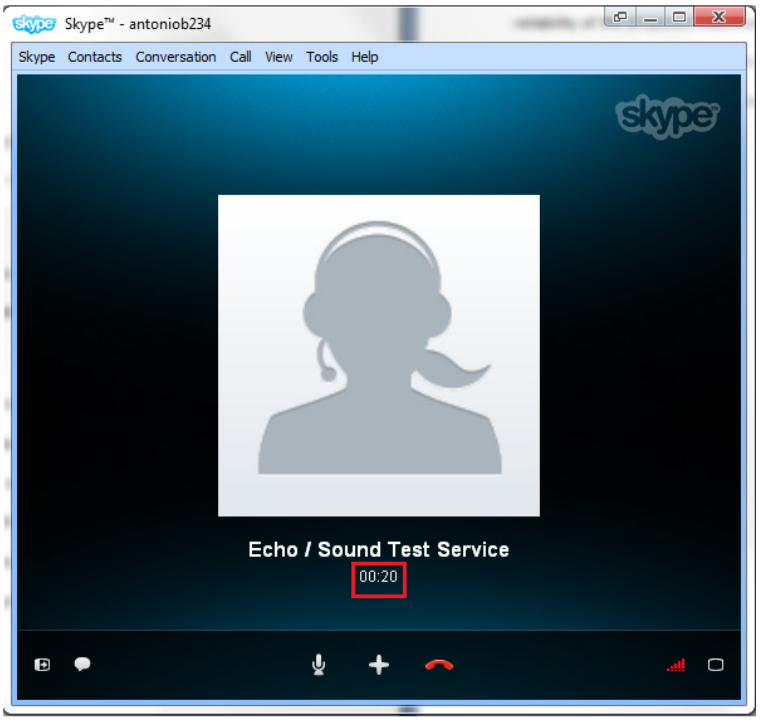

Figure 25 Skype Call Duration

The error margin of this approach can be calculated by checking the duration of the flows classified as Skype call flows, and the total duration of the call that was conducted obtained from the Skype application timer. An error margin will exist since there is always a slight difference between the duration of the flows labelled as Skype calls and the original duration of the Skype call conducted. This is due to several factors namely (I) the Skype client does not disconnect from the super nodes immediately after the Skype call is finished, (II) the client communicates with the super node, transmitting packets before the call timer commences, which results in the slight difference between both the original call duration and the duration of the flows labelled as Skype calls, and (III) if the call is dropped and re-established due to a bad or slow connection, the duration of the flows labelled as Skype calls will not represent the true value of the duration of the conducted Skype call.

4.4.3 Voice & Video Traffic Identification

The same data sets used for the BitTorrent phase are used to test the accuracy and reliability of the proposed classification scheme when the targeted traffic to identify is Skype traffic. Detailed descriptions of each of the 5 data sets are displayed in Table 9 where in each data set several Skype calls are conducted.

For each data set, the proposed classification scheme was applied in order to identify the host's IP and port suspected of Skype activity. Afterwards, it is necessary to extract all the flows that are associated with the identified Skype client, and to determine the existence of any placed Skype calls. The method of grouping all the Skype flows and identifying Skype call flows is displayed in Figure 26 and is applied to all the data sets.

Once all of the Skype flows have been acquired, it is essential to differentiate between actual voice or video traffic and other probing, messaging or signalling traffic. In this analysis we will focus on classifying the cases where UDP is used for the data transfer of voice and video calls, given that Skype clients strongly prefer UDP for voice and video connections.

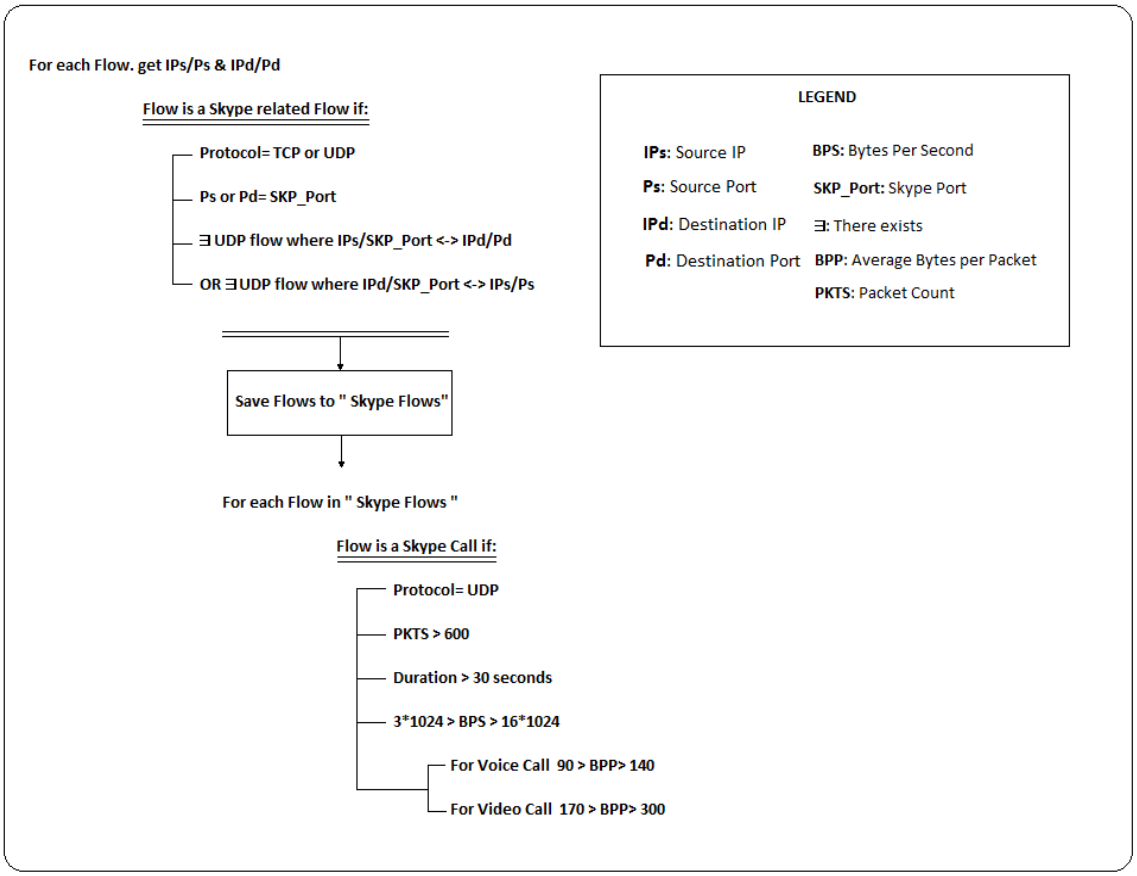

Figure 26 Skype Flow and Skype Call Extraction Method

4.4.3.1 Data Set 1

In data set 1, the scheme proposed in the previous chapter was successful in identifying the host's IP and port suspected of Skype activity and the results are displayed in Table 20.

Table 20 Suspected Skype Hosts and Ports in Data Set 1

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP flow to port 33033 & 80/443	Associated heavy hitting flows	Classification
192.168.1.138	52799	✓	✗	✗	✓	Other
192.168.1.138	60891	✗	✓	✓	✗	Skype
192.168.1.138	42176	✓	✗	✗	✓	Other
192.168.1.138	16677	✓	✗	✗	✓	Other
192.168.1.138	35854	✗	✓	✗	✗	Other

Table 21 represents the results of applying the Skype call extraction queries (Figure 26) on data set 1 where an 11 minute Skype call was conducted (6 minutes voice, 5 minutes video).

Table 21 Results of Skype Call Detection Queries on Dataset 1

IP Address	Bytes Per Packet	Packet Count	Duration is Seconds	Bytes Per Second	Classification
213.146.189.234	128.9	11870	241	6.2 Kbps	Audio Call
149.13.32.249	116.3	13707	278	5.6 Kbps	Audio Call
74.126.125.180	223.7	8435	222	8.3 Kbps	Video Call

Out of 17943 flows in data set 1, only 3 flows are classified as active Skype calls, where their statistics matched the Skype call classification heuristic and are displayed in Table 21. The existence of two separate audio call flows does not mean that there were two audio calls conducted, as the Skype client may connect to several super nodes to relay the traffic to the other end of the Skype call. The timestamps of the flows should be checked to verify the actual number of Skype calls. If the timestamps of the flows overlap, that means that they belong to the same call. If not then they are separate Skype calls. The last flow in Table 21 represents the video portion of the call where the average bytes per packet values were within the video range previously stated.

The total number of seconds of the three flows is equal to 741 seconds or approximately 12.3 minutes. Since the first two flows overlap in a timeframe of 8 seconds, we can subtract this value from the total value giving a total time of 733 seconds or 12.2 minutes which is roughly equal to the real time of the Skype call which is 11 minutes, with a 1 minute difference between both durations.

4.4.3.2 Data Set 2

In data set 2, the classification scheme proposed in the previous chapter was successful in identifying the host's IP and port suspected of Skype activity and is displayed in Table 22.

Table 22 Suspected Skype Hosts and Ports in Data Set 2

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP flow to port 33033 & 80/443	Associated heavy hitting flows	Classification
134.117..61.54	19654	✓	✗	✗	✓	Other
134.117..61.54	58695	✗	✓	✓	✗	Skype
134.117..61.54	3654	✓	✗	✗	✓	Other
134.117..61.54	12994	✓	✗	✗	✓	Other

In this data set, 4 ports were suspected as potential P2P participants where the Skype client's port number was identified and the other 3 ports represents the ports of the BitTorrent client, the P2P radio streaming application and the Trojan activity, which were all active throughout the duration of this session.

Table 23 represents the results of the Skype call extraction queries when performed on data set 2 where 3 Skype calls were conducted over the span of 29 minutes (12, 8, 9 minute calls).

Table 23 Results of Skype Call Detection Queries on Dataset 2

IP Address	Bytes Per Packet	Packet Count	Duration in Seconds	Bytes Per Second	Classification
111.221.74.20	105.7	37700	1385	2.5 Kbps	Audio Call
96.54.44.121	109.2	10508	486	2.3 Kbps	Audio Call
24.54.10.11	112.05	16219	493	3.6 Kbps	Audio Call

In the first two flows, it can be observed that the bytes per second value does not fall within the set threshold of 3-16 Kbps as the connection speed was very slow when the Skype calls were conducted. As mentioned, Skype is designed to operate even if there is limited bandwidth available. Thus, the bytes per second range set in the classification scheme is not a decisive condition for classifying Skype calls, as slow connections will fall outside of the range. By comparing the timestamps of the flows, an overlap was noticed between the first two flows where they started and finished at the same time. However, the first flow commenced again shortly after it ended by 40 seconds. From this behavior, it can be concluded that the first two IPs are super nodes that were used to relay the connection for the first call, and for the second call only one super node was used. The timestamp for the third flow does not overlap with the others, representing a third separate call. The sum of the duration of all three flows is 2364 or 39.4 minutes. Since the first two flows are overlapping and were used in the first call, the duration of the second flow can be subtracted from the total to obtain 1878 seconds or 31.2 minutes. This value is approximately equal to the ground truth of 29 minutes with a slight difference of approximately 2 minutes between both the original call duration and the duration of flows labelled as Skype calls.

4.4.3.3 Data Set 3

In data set 3, the classification scheme proposed in the previous chapter was successful in identifying the host's IP and port suspected of Skype activity and is displayed in Table 24.

Table 24 Suspected Skype Hosts and Ports in Data Set 3

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP flow to port 33033 & 80/443	Associated heavy hitting flows	Classification
192.168.8.138	4456	✓	✗	✗	✓	Other
192.168.8.138	36589	✓	✗	✗	✓	Other
192.168.8.138	25541	✗	✓	✓	✗	Skype
192.168.8.138	8668	✓	✗	✗	✓	Other

Table 25 represents the results of the Skype call extraction queries (Figure 26) when performed on dataset 3 where a 20 minute video call was conducted via a host machine in a Carleton residence dorm and the receiving end was connected to a coffee house Wi-Fi connection.

Table 25 Results of Skype Call Detection Queries on Dataset 3

IP Address	Bytes Per Packet	Packet Count	Duration in Seconds	Bytes Per Second	Classification
67.193.154.255	290.02	26969	711	11 Kbps	Video Call
24.213.82.183	266	5405	193	7 Kbps	Video Call
130.132.106.40	137	1064	234	0.6 Kbps	Audio Call

During the call, the connection was dropped several times due to the bad connection of the end user connected to a public Wi-Fi. The first two flows are labelled video flows as their statistics match the parameters mentioned in the Skype call heuristic. The last flow is mislabelled as a voice call due to the slow connection between both hosts as the average byte per packet and

the bytes per second values were too small to be considered video calls. Nonetheless, the flow was labelled as a call, regardless whether it was a voice or video. The total time of the three flows is 18.9 minutes which is less than the original 20 minute ground truth due to the several drops in the connection between the hosts, with a 1.1 minute difference between both durations.

4.4.3.4 Data Set 4

In data set 4, the classification scheme proposed in the previous chapter was successful in identifying the host's IP and port suspected of Skype activity and is displayed in Table 26.

Table 26 Suspected Skype Hosts and Ports in Data Set 4

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP flow to port 33033 && 80/443	Associated heavy hitting flows	Classification
134.117.61.46	61256	✓	✗	✗	✓	Other
134.117.61.46	16700	✗	✓	✓	✗	Skype
134.117.61.46	19880	✓	✗	✗	✓	Other
135.178.111.101	57362	✓	✗	✗	✓	Other
125.149.116.103	57362	✗	✓	✗	✗	Other

Table 27 represents the results of the Skype call extraction queries (Figure 25) when performed on data set 4.The two flows in Table 27 are labelled as Skype calls in data set 4 where a 45

minute video call was conducted. The flow features of both the flows in Table 27 matched the Skype video call criteria where both the flows were using the Skype predefined port which was identified earlier. By adding the duration of both flows, we obtain a value of 2744 seconds or approximately 46 minutes. This value is off by 1 minute, when compared to the original duration of the call (45 minutes) yielding an error value of approximately 2% when both durations are compared.

Table 27 Results of Skype Call Detection Queries on Dataset 4

IP Address	Bytes Per Packet	Packet Count	Duration in Seconds	Bytes Per Second	Classification
173.32.150.45	285	98139	1951	14 Kbps	Video Call
82.31.149.37	240	27068	793	8 Kbps	Video Call

4.4.3.5 Data Set 5

In data set 5, the proposed classification scheme was successful in identifying the host's IP and port suspected of Skype activity and is displayed in Table 28.

Table 28 Suspected Skype Hosts and Ports in Data Set 5

IP	Port	Within Torrent Probing Range	Within Skype Probing Range	Outgoing UDP flow to port 33033 & 80/443	Associated heavy hitting flows	Classification
172.19.18.175	7892	✓	✗	✗	✓	Other
172.19.18.175	1935	✓	✗	✗	✓	Other
172.19.18.175	44485	✗	✓	✓	✗	Skype
172.19.18.175	9936	✓	✗	✗	✓	Other

In data set 5, a 20 minute voice call was conducted. Table 29 represents the results of the Skype call extraction queries (Figure 26) when performed on data set 5. The 2 flows in Table 29 represent the flows labelled as Skype call flows where their flow features matched the Skype voice call criteria. The sum of the duration of both flows is equal to 1198 seconds or approximately 19.9 minutes, which is off by a very small value of 0.1 minute when compared to the ground truth duration of 20 minutes.

Table 29 Results of Skype Call Detection Queries on Dataset 5

IP Address	Bytes Per Packet	Packet Count	Duration in Seconds	Bytes Per Second	Classification
213.199.179.153	113	81395	988	9 Kbps	Audio Call
212.8.166.36	96	13440	210	6 Kbps	Audio Call

4.5 Skype Detection Summary and Limitations

The accuracy of this approach can be obtained by comparing the timestamps and the duration of flows classified as Skype call flows to the actual duration of the conducted call, as displayed in the results section where the difference in durations for each testing case was roughly ±1-2 minutes. For each of the test cases the difference between the acquired and the true duration was very small. This indicates the high accuracy of determining the duration of any placed Skype calls using Netflow traces. After obtaining the Skype port through the previously mentioned heuristics, it is easy to extract all of the Skype flows, and identify whether they are call flows or peer pinging flows. This is due to that the Skype clients use the predefined UDP port for contacting peers, which ensures the capturing of all the flows associated with the Skype client, avoiding false positive and false negative values as displayed in the Skype flow accuracy in Equation 3. The only drawback of this approach is that the Skype client must be version 5 or higher since older versions do not operate in the same manner. In addition, the capturing of the session must begin before the user opens the Skype client to ensure the capturing of the start-up connection pattern of Skype clients. The results are summarized in Table 30.

Table 30 Skype Test Results Summary

	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5
Skype Host Detected	✓	✓	✓	✓	✓
Original Skype Call Duration	6 mins voice, 5 mins video	12, 8, 9 mins voice	20 mins video	45 mins video	20 mins voice
Detected Skype Call Duration	6 mins voice, 6.2 mins video	13 mins 9 mins 10.2 mins	18.9 mins	46 mins	19.9 mins
Error Margin	+1.2 mins	+2.2 mins	-1.1 mins	+1 mins	-0.1 mins

4.6 Results Compared to other Researches

This section presents a comparison between the results of the classification scheme presented in this thesis and the results of other proposed schemes for BitTorrent and Skype identification.

4.6. 1 BitTorrent Test Results Comparison

In this section, we display the accuracy of the proposed scheme for identifying BitTorrent traffic when compared to other approaches suggested in different papers. In order to validate the results, two approaches are selected for comparison.

The first method is a host behavior approach proposed by Mahanti et al in [57]. The suggested approach utilizes the following five features for identifying BitTorrent traffic:

1- Flow Size
2- Flow Duration
3- Flow Concurrency
4- Transfer Volume
5- IP Geographical Location

Graphical models are generated for each application using the mentioned features. If a flow satisfies the thresholds within the model, the flow is considered a BitTorrent related flow. The authors assume that all BitTorrent related file transfers occur via TCP. Detailed description regarding the mentioned approach can be found in [57].

The second method selected for comparison is a machine learning approach proposed by Yuan et al in [67]. The suggested approach utilizes SVM with a RBF kernel function for extracting BitTorrent flows. The proposed scheme uses 19 unique flow features for classification such as protocol, average packet size and IPs. The different parameters and specifications regarding this approach are discussed in detail in [67].

In order to validate the accuracy of the scheme proposed in this thesis, a comparison is performed between the results of the proposed scheme and the two mentioned approaches when applied to the five data sets used for the testing phase.

Table 31 and Figure 27 represent the accuracy of the proposed scheme in this thesis, the host behavior approach suggested by Mahanti et al [57], and the SVM approach suggested by Yuan et al [67] for each of the 5 data sets.

Table 31 BitTorrent Byte-Wise Accuracy Results of the 3 Approaches

Data Sets	Proposed Scheme	Host Behavior	SVM
Data Set 1	95.1	67.5	93.8
Data Set 2	91.8	92.1	88.5
Data Set 3	91.3	71.2	74.3
Data Set 4	94.8	81.4	81.3
Data Set 5	95.4	73.8	72.1

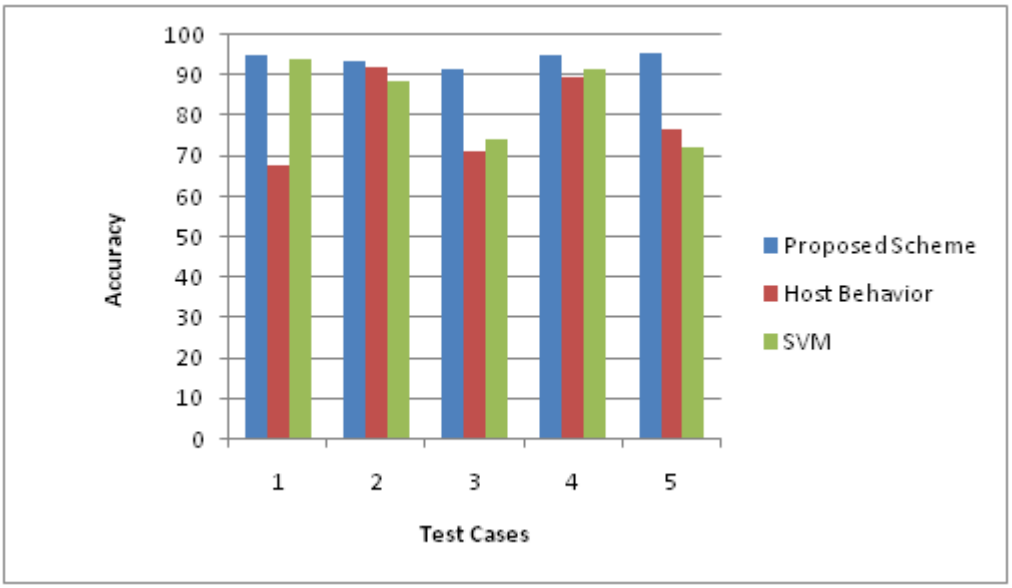

Figure 27 Accuracy Histogram of the 3 Approaches

The approaches suggested in [57] and [67] produce strong results when applied to data sets that do not contain challenging background traffic. The accuracy is significantly decreased when these methods are applied to data sets that contain BitTorrent traffic overlapped with applications that generate similar flows and behave in the same fashion.

The proposed scheme outperformed the approaches suggested in [57] and [67] in all of the 5 data sets due to the following:

1- P2P radio applications use UDP probing to connect to peers that are tuned in to the same radio channel by using a random UDP listening port. Once the application is connected to the peers, the transfer of audio packets takes place via UDP. All the flows generated from P2P radio applications utilize the UDP port that is predefined in the application's settings. P2P radio streaming applications do not contact port 80 and only generate heavy hitting flows that have average bytes per packet value within the 600-

1000 bytes range according to [66]. P2P radio streaming flows are not misclassified or labelled as BitTorrent or Skype flows in the scheme proposed in this thesis, since they do not satisfy the mentioned heuristics or the separating thresholds. The host behavior approach in [57] does not account for UDP traffic so the flows generated from the P2P radio streaming application does not affect the accuracy of the method in [57]. The SVM approach suggested in [67] is heavily affected by the P2P radio streaming flows (data set 2) as it relies on flow features for classification and the flow features of P2P radio streaming flows are similar to those of BitTorrent file transfers.

2- P2P T.V. streaming applications utilize UDP probing in order to contact peers that are tuned in to the same T.V. channel by using a random UDP listening port predefined in the software's settings. Once the application is connected to the peers, the transfer of the data packets takes place via TCP. The average bytes per packet of the data transfer flows generated by P2P T.V. streaming application are within the 200-1200 according to [66]. If a host is concurrently running BitTorrent and P2P T.V. streaming applications, the TCP flows within the range of 200-1200 bytes per packet will contain a mixture of both P2P T.V. streaming flows and BitTorrent file transfer flows. The scheme proposed in this thesis as well as both the approaches suggested in [57] and [67] cannot distinguish between these flows as they share similar flow features, leading to a high value of false positives. The proposed scheme outperformed the approaches in [57] and [67] due to the average bytes per packet condition where flows are labelled as BitTorrent flows if the average bytes per packet value is greater than 800 bytes. This threshold does separate between BitTorrent and P2P T.V. streaming flows, nonetheless it ensures

capturing the majority of BitTorrent flows and only a small portion of the P2P T.V. streaming flows that fall in that range which minimizes the false positives value. The accuracy of the proposed approaches in [57] and [67] is severely hindered due to the similarity of BitTorrent flows and P2P T.V. streaming flows (data set 3).

3- The SUS/Unkpacker [73] Trojan and other similar viruses utilize a set of random UDP ports in order to communicate with the attacker and in some cases it may contact an IP with a destination port 80. This may be similar to other P2P applications which causes mislabelling the flows generated from the Trojan as BitTorrent related flows. In the proposed scheme, a host must satisfy a collection of heuristics in a sequential manner in order to be considered a BitTorrent or Skype participant. The flows generated by viruses and Trojans are not mislabelled since they do not satisfy all of the heuristics and conditions. The proposed approaches in [57] and [67] may mislabel virus and Trojan flows as P2P flows due to the similarity between their flow features (data set 2).

4- P2P clients are not the only applications that use ports greater than 1024. Other services such as Adobe flash video and Google Talk use port numbers that are greater than 1024 and are registered as official ports on the IANA port list. The authors in [57] state that in order for a flow to be classified as a P2P related flow, the source and destination port number must be greater than 1024. This assumption results in a high value of false positives when BitTorrent traffic overlaps with traffic generated from other services that use ports greater in 1024. In data set 5, a HTTP streaming page was open which uses the Adobe flash video port (1935). The accuracy of both the host behaviour approach suggested in [57] and the SVM approach suggested in [67] is heavily affected and

produced inaccurate results when compared to the approach suggested in this thesis as displayed in Table 30 for data set 5. This is due to the presence of Adobe flash video flows which can be misinterpreted as BitTorrent flows as their flow features are very similar where they both generate heavy hitters and use ports greater than 1024.

5- The host behavior approach proposed in [57] operates in the assumption that all BitTorrent file transfers take place only via TCP. This assumption leads to a very large value of false negatives when this approach is applied to traces generated by the new generation of BitTorrent clients using uTP where all data transfers use UDP. The approach presented in this thesis and the SVM approach in [67] are not be affected by the transport layer protocol of BitTorrent clients as displayed in Table 30 where 4 out of 5 data sets had BitTorrent clients using UDP for file transfers. The SVM approach is protocol independent and does not rely on it for classification. On the other hand, the method presented in this thesis takes into consideration both data transfers that use TCP and UDP.

4.6.2 Skype Test Results Comparison

In this section, we display the accuracy of the proposed scheme for identifying Skype hosts when compared to other approaches suggested in different papers. In order to validate the results, two approaches are selected for comparison.

The first approach, is the Skype host identification scheme suggested in [55] by L. Ptacek. In this approach, a reverse DNS lookup is preformed on the IP addresses within Netflow traces to determine whether or not a certain host has contacted an IP that resolves to the Skype server domain name. Once an IP address contacting a Skype domain name is detected, the IP is labelled as a potential Skype host. Further investigation is carried out to determine if the suspected host is truly a Skype participant or not. Detailed information regarding this approach can be found in [55].

The second approach used for comparison is the scheme suggested in [36].In this particular approach, Chun-Ming et al have been successful in reverse engineering the Skype protocol where 12 key steps have been identified. All Skype clients must go through these 12 steps which take place on the start up of any Skype client. These 12 steps capture the connection pattern and the behavior of Skype clients from the initialization and the start up phase until the authentication and call placement. The only drawback using this approach is that it is only valid for older Skype versions, since the newer Skype clients do not follow the same 12 steps demonstrated in [36].

In order to validate the accuracy of the scheme proposed in this thesis, a comparison is made between the results of the proposed scheme and the two mentioned approaches when applied to the 5 data sets used for the testing phase.

Both the selected approaches and the model suggested in this thesis can accurately extract all Skype flows once the Skype host has been identified. As a result, we will compare the

approaches in terms of whether they are successful in identifying the Skype's host IP/Port and add a 'Host Detected' label if they are successful.

Table 32 represents the results of each approach in identifying the suspected Skype hosts in each of the 5 data sets.

Table 32 Skype Host Detection for the 3 Approaches

	Skype Host Detected				
	Data Set 1	Data Set 2	Data Set 3	Data Set 4	Data Set 5
Suggested Approach	✓	✓	✓	✓	✓
Chun-Ming et al's Approach	✗	✗	✗	✗	✗
L. Ptacek Approach	✗	✓	✗	✓	✓

The approach suggested in this thesis was successful in identifying the suspected Skype hosts in each of the 5 data sets as displayed in the test results section.

The approach suggested by Chun-Ming et al [36] could not identify any of the Skype hosts in the data sets. In each of the 5 data sets the Skype clients that were active were all version 5.x.x or higher. The scheme proposed by Chun-Ming et al uses 12 key steps that take place in the older version of Skype, which renders their approach obsolete when applied to the current newer versions of Skype. The 12 steps mentioned in [36] did not undergo drastic changes from the older versions of Skype to the 5.x.x versions, but some key features used for identification have been changed. One of the important changes can be noticed in the Skype update check, where in [36] the authors state that it takes place via TCP and a random port number. On the other hand, in the newer version of Skype the update check takes place via UDP and the predefined

Skype port. This approach was selected to demonstrate that the older classification models must be continuously updated and modified in order to accommodate the changes of the newer versions and should be based on more sophisticated features and connection patterns that are unlikely to change with newer versions.

The approach suggested by L.Ptacek in [66] was successful in identifying the Skype hosts in 3 out of 5 data sets. The approach in [36] is highly dependent on the results of a reverse DNS lookup. Some IP addresses remain ambiguous after a DNS lookup and also some proxies may be used which result in the inaccuracy of this approach. Another drawback of this approach is that sometimes the IP addresses within Netflow traces are encrypted or scrambled for security concerns and privacy which makes this approach infeasible for application. Although this approach is sometimes successful in identifying Skype hosts, it is highly unreliable if the result of the reverse DNS lookup fails to resolve the IP addresses to Skype domains.

Chapter 5 Conclusions

5.1 Conclusions

Developing a generalized approach for accurately identifying all types of internet traffic is infeasible, since applications have overlapping features and common characteristics which increase the probability of misclassification. Despite these similarities, each application has a set of unique characteristics which can be obtained by analysing the applications behaviors and connection patterns. Even if general approaches yield fine results, it will not perform as well when applied to data sets with challenging back ground traffic, as displayed d in the test results section. The future of internet traffic classification lays in developing applications specific models that are capable of accurately identifying the targeted applications which are more reliable than general approaches.

In this thesis, a set of heuristics and classification schemes based on the discriminating attributes and behavioral nature of different P2P applications are presented and integrated together in order to successfully identify the targeted P2P applications, which are BitTorrent clients and Skype. The proposed classification model relies solely on information that is captured by using any traditional collector that provides IP flow information.

These heuristics have been integrated and combined together to produce a classification model capable of indentifying BitTorrent and Skype flows with high accuracy. Although some of these heuristics and features have been previously utilized, they are organized in a different fashion to serve the purpose of classifying P2P applications, as the order of applying these heuristics is

essential to achieve high accuracy. Some of the presented heuristics are referenced in other researches but have been combined with other newly obtained ones such as the DHT UDP probing heuristic and the other application specific heuristics based on the newer versions of both BitTorrent and Skype that are displayed in Chapter 3.

The presented classification scheme has been tested on real life data sets where P2P and non P2P activities were conducted. The accuracy output for BitTorrent byte-wise accuracy was very high ranging from 91.3-95.4%. The information regarding the total amount of downloaded bytes is very useful for network analysts and administrators as it can be used for bandwidth management and allocation, and in identifying 'bandwidth hogs' that cause congestion and deteriorate network performance. On the other hand, all Skype flows were successfully extracted by exploiting certain connection patterns and start up features that are unique for the newer versions of Skype. All of the voice and video call flows were successfully extracted by using parameters and thresholds which have been deduced by monitoring a collection of live ongoing Skype calls. Additionally, an original technique for determining a rough estimate of any conducted Skype call using Netflow traces has been presented, giving fine results where the difference between the deduced duration and the original duration was ±1-2 minutes for all the tested cases.

The results of the proposed scheme have proven to be superior to other existing approaches in terms of accurately identifying BitTorrent and Skype flows within Netflow traces.

5.2 Limitations

As mentioned this approach only detects two types of P2P activities which are BitTorrent and Skype. Nonetheless, these two application are considered to be two of the most important and popular P2P services. BitTorrent clients are extremely popular for file sharing and downloading, and can consume a very large portion of the available bandwidth, while Skype is considered the number one application when it comes to VoIP services. Other classification methods should be devised for identifying different P2P traffic other than the mentioned applications such as P2P video streaming and online gaming as these heuristics are based on features unique for both BitTorrent clients and Skype clients, and are only applicable to these applications.

The presented model has been tested with the up to date versions of both BitTorrent and Skype applications. Its results have been proven to be superior to those of other approaches. Although newer versions may change the behavior of these applications, some core connection patterns will remain the same. For example, A BitTorrent client will always contact the trackers in the .torrent file in order to initiate the downloading process. In addition for Skype, a client will always connect to a super node on start-up. The way these patterns are carried out may change with later versions, but the fundamental connection patterns will remain which in turn solidifies the validity of the classification procedure.

5.3 Suggestions for Future Work

The proposed model performs the classification procedure offline, where the internet traces are stored before processing them. The model can be enhanced to achieve real time

classification instantaneously. This can be accomplished by setting the export interval of the Netflow collector to a relativity small time frame, and by feeding the exported traces to the classification model as soon as they arrive. The model can then identify different BitTorrent and Skype activities instantly and take action accordingly. The speed in which the real time classification takes place may vary depending on different factors such as the export interval and the complexity and size of the exported traces.

In addition, the classification model should be frequently updated and modified in order to accommodate the changes of the newer versions of both BitTorrent and Skype applications. The newer versions should be analysed to identify other unique features and distinguishing connection patterns in order to improve the accuracy and avoid misclassification.

References

[1] IPOQUE, "Internet Study 2008-2009", *Retrived: 12 May 2012,* http://portal.ipoque.com/downloads/index/get/?id=265.

[2] P.Schneider. "TCP/IP traffic Classification Based on port numbers", *Diploma Thesis, Division of Applied Science, Harvard University*, Cambridge, MA, USA, 1996.

[3]S. Yoon, J.W. Park, Y.S. Oh, J.S. Park, M.S. Kim. " Internet Application Traffic Classification Using Fixed IP-port", *In Proc. APNOMS (Asia-Pacific Network Operations and Management Symposium),* Jeju, Korea, September, 2009.

[4] CAIDA, "CoralReef", *Retrieved: 7 March 2012,* http://www.caida.org/tools/measurement/coralreef/

[5] M. Roesch. "Snort - Lightweight Intrusion Detection for Networks", *In Proc. 13th LISA Conference,* Seattle, WA, USA, November 1999.

[6] V. Paxon. "Bro: a system for detecting network intruders in real-time", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, v.31 n.23-24, p.2435-2463, Berkeley, CA, U.S.A, December 1999.

[7] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. "ACAS: Automated Construction of Application Signatures", *In Proc. SIGCOMM'05 MineNet Workshop*, Philadelphia, PA, USA, August 22-26, 2005.

[8] S. Sen, O. Spatscheck, and D. Wang. "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures", *In Proc. WWW2005*, New York, NY, USA, May 17-22, 2004.

[9] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, and H. Chung. "Content-aware internet application traffic measurement and analysis", *In Proc. IEEE/IFIP NOMS*, Seoul, South Korea, April 2004.

[10] A. Moore and K. Papagiannaki. " Toward the Accurate Identication of Network Applications", *In Proc. PAM'05*, Boston, MA, U.S.A, March 2005.

[11] L. Bernaille, R. Teixeira, and K. Salamatian. "Early application identification", *In Proc. ACM CoNEXT*, Lisbon, Portugal December 2006.

[12] M. Horng, C. Chen, C. Chuang, and C. Lin. "Identification and Analysis of P2P Traffic – An Example of BitTorrent" , *In Proc. ICICIC'06*, Beijing, China 2006.

[13] Matsuda, T. Nakamura, F. Wakahara, and Y. Tanaka. " Traffic Features Fit for P2P Discrimination", *In Proc. APSITT 2005* , Yangon, Burma, 2005.

[14] A. Finamore, M. Mellia, , M. Meo and M. Rossi. "KISS: Stochastic Packet Inspection" *in Proc. Traffic Measurement and Analysis (TMA) Workshop at IFIP Networking 2009*, Aachen, Germany,
May 2009.

[15] A. Madhukar and C. Williamson. "A Longitudinal Study of P2P Traffic Classification", *In Proc. 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006)*, pp. 179–188. IEEE Press, New York, NY, U.S.A 2006.

[16] Z. Guo and Z. Qiu. "Identification Peer-to-Peer Traffic for High Speed Networks Using Packet Sampling and Application Signatures", *In Proc. 9th International Conference on Signal Processing (ICSP 2008)*, pp. 2013–2019. IEEE Press, New York, NY, U.S.A, 2008.

[17] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. "Flow clustering using machine learning techniques", *In Proc. Fifth Passive and Active Measurement Workshop*, Antibes-Juan-Les-Pins, France, 2004.

[18] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. "Traffic Classification through Simple Statistical Fingerprinting", Computer *Communications Review*, 37(1):7–16, 2007.

[19] N. Williams, S. Zander, and G. Armitrage. " A Preliminary PerformanceComparison of Five Machine Learning Algorithms for Practical IP Traffic FlowClassification" ,*Computer Communication Review*, 30:5–16, October 2006.

[20] T. Auld, A. W. Moore, and S. F. Gull. "Bayesian neural networks for internet traffic classification", *In Proc. IEEE Transactions on Neural Networks*, Volume 18(1):223–239, January 2007.

[21] R. Wang, Y. Liu, Y. Yang, and X. Zhou. "Solving P2P Traffic Identification Problems Via Optimized Support Vector Machines", *In Proc. IEEE/ACS International Conference on Computer Systems and Applications,(AICCSA'07)*, pg. 165-171, Amman, Jordan, May 2007.

[22] S. Zander, T. Nguyen, and G. Armitage. "Automated traffic classification and application identification using machine learning", *In Proc. IEEE 30th Conference on Local Computer Networks (LCN 2005)*, Sydney, Australia, November 2005.

[23] A. Lakhina, M. Crovella, and C. Diot. "Mining anomalies using traffic feature distributions", *In Proc. ACM SIGCOMM*, Philadelphia, PA, USA, August 2005.

[24] A. Moore and D. Zuev. "Internet traffic classification using bayesian analysis", *In Proc. ACM SIGMETRICS*, New York, NY, USA, 2005.

[25] E. Ramirez, "BitTorrent Popularity Statistics", *Retrieved: 14 June 2012*, http://torrentfreak.com/bittorrent-more-popular-than-ever-071009/.

[26] E. Ramirez, "BitTorrent Growth and Progression", *Retrieved: 14 June 2012* http://torrentfreak.com/bittorrent-sites-show-explosive-growth-080322/.

[27] N. Unuth, "What is Skype? ", *Retrieved: 4 March 2012*, http://voip.about.com/od/voipsoftware/a/whatisskype.htm

[28] T. Karagiannis , A. Broido, and M. Faloutsos. “Transport layer identification of P2P traffic”, *In Proc. IMC'04*, Taormina, Sicily, Italy, 2004.

[29] T. Karagiannis , K. Konstantina, and A. Papagiannaki. “ BLINC: Multilevel traffic classification in the dark”, *In Proc. SIGCOMM'05*, Philadelphia, PA, USA, 2005.

[30] S. Zhang, C. Gu, and X. Xue. “Encrypted Internet Traffic Classification Method based on Host Behavior”, *JDCTA: International Journal of Digital Content Technology and its Applications*, Vol. 5, No. 3, pp. 167 ~ 174, 2011.

[31] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. “Network monitoring using traffic dispersion graphs”, *In Proc. ACM IMC*, New York, NY, U.S.A, October 2007.

[32] J. Chung, B. Park, J. Won , J. Strassner and W. Hong. ”Traffic Classification Based on Flow Similarity”, *In Proc. IPOM 2009*. LNCS, vol. 5843, pp. 65–77, Heidelberg , Germany, 2009.

[33] B. Cohen, “BitTorrent Protocol Specification”, *Retrieved: 21 October 2011*, http://bittorrent.org/beps/bep_0003.html

[34] BitTorrent Inc.,”uTorrent”, *Retrieved: 26 October 2011,* http://www.utorrent.com

[35] Microsoft Skype Division, “Skype”, *Retrieved: 12 November 2011,* http://www.skype.com

[36] L. Chun-Ming and Y. Chan. “Network Forensic on Encrypted Peer-to-Peer VoIP Traffics and the Detection, Blocking, and Prioritization of Skype Traffics”, *In Proc. 16th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2007)*, Pages 401-408, Paris, France, June 2007.

[37] Cisco, “Netflow”, *Retrieved 4 February 2012*, http://www.cisco.com/go/netflow

[38] S. Campbell. ”NetFlow Now the Industry Standard in Network Traffic Accounting”, *Retrieved: 15 December 2011,* http://netflow.tmcnet.com/articles/250278-netflow-now-industry-standard-network-traffic-accounting.htm

[39] S. Campbell. “NetFlow Sets the Standard in Traffic Monitoring”, *Retrieved: 15 December 2011 ,* http://netflow.tmcnet.com/newsall.aspx

[40] G. Barlett, J. Heidemann, and C. Papadopoulos. ”Inherent Behaviors of On-line Detection of Peer-to-Peer File Sharing”, *In Proc. IEEE Global Internet*, Anchorage, Alaska, USA, 2007.

[41] A. Madhukar and C. Williamson. ”A Longitudinal Study of P2P Traffic Classification”, *In Proc. 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006)*, pp. 179–188. IEEE Press, New York, NY, U.S.A, 2006.

[42] Z. Guo, Z. Qiu. “Identification Peer-to-Peer Traffic for High Speed Networks Using Packet Sampling and Application Signatures”, *In Proc. 9th International Conference on Signal Processing (ICSP 2008),* pp. 2013–2019. IEEE Press, New York, NY, USA, 2008.

[43] J. Gomes, P. Ignacio, M. Freire, M. Pereira and P. Monteiro. “Analysis of Peer-to-Peer Traffic Using a Behavioural Method Based on Entropy”, *In Proc. IEEE International Performance, Computing and Communications Conference (IPCCC 2008)*, pp. 201–208. IEEE Press, New York, NY, USA 2008.

[44] A. Gossett , I. Papapanagiotou, and M. Devetsikiotis. “An apparatus for P2P classification in Netflow traces”, *In Proc GLOBECOM Workshops (GC Wkshps)*, 2010 IEEE , vol., no., pp.1361-1366, 6-10, Miami, Florida, U.S.A, December 2010.

[45] L. Liu, H. Feng, W. Huang, and Y. Li. ”A Peer-To-Peer Traffic Identification Method Using Machine Learning”, *In Proc. International Conference on Networking, Architecture, and Storage (NAS 2007)*, pp. 155–160. IEEE Press, New York, NY, USA, 2007.

[46] M. Soysal, E. Schmidt. ”An accurate evaluation of machine learning algorithms for flow based P2P traffic detection”, *In Proc. 22nd International Symposium on Computer and Information Sciences (ISCIS 2007)*, pp. 1–6, IEEE Press, New York, NY, USA, 2007.

[47] M. Collins, and M. Reiter. “Finding Peer-to-Peer File Sharing Using Coarse Network Behaviors”, *In Proc. 11th European Symposium on Research in Computer Security*, pages 1–17, Hamburg, Germany, September 2006.

[48] L. Zhou, Z. Li, and B. Liu, “P2P Traffic Identification by TCP Flow Analysis”, In *Proc. of International Workshop on Networking, Architecture and Storages*, Shenyang, China, December 2006.

[49] A. Loewenstern, “BitTorrent DHT Protocol Specification”, *Retrieved: 21 October 2011,* http://bittorrent.org/beps/bep_0005.html

[50] S. Ehlert, S. Petgang, T. Magedanz, and D. Sisalem. “Analysis and signature of Skype VoIP session traffic”, *In Proc. CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology* , pp. 83–89, St. Thomas, USVI, USA, Nov/Dec 2006.

[51] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. “Revealing Skype traffic: When randomness plays with you”, *In Proc. SIGCOMM’07*, New York, August 2007.

[52] M. Perényi, and S. Molnár. ”Enhanced Skype traffic identification” , *In Proc. 2nd International Conference on Performance Evaluation Methodologies and Tools, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*, ICST, Brussels, Belgium , 2007.

[53] S. Guha, N. Daswani, and R. Jain. ”An experimental study of the Skype peer-to-peer VoIP system”, *In Proc. 5th International Workshop on Peer-to-Peer Systems*, pp.1-6 Santa Barbra, CA, USA, February 2006.

[54] S. Baset, and H. Schulzrinne. ”An analysis of the Skype peer-to-peer internet telephony protocol”, *In Proc. IEEE INFOCOM'06*, Barcelona, Spain, April 2006.

[55] L.Ptacek. “Analysis and Detection of Skype Network Traffic”, *Masaryk University Faculty of Informatics, Master Thesis*, Brno, Czech Republic, Spring 2011.

[56] D. Adami,C. Callegari, S. Giordano, M. Pagano, and T. Pepe. ”A Real-Time Algorithm for Skype Traffic Detection and Classification”, *In Proc. NEW2AN*, St. Petersburg, Russia, 2009.

[57] N. Basher, A. Mahanti, A. Mahanti, C. Willamson, and M. Arlitt. “A Comparative Analysis of Web and Peer-to-Peer Traffic”, *In Proc. WWW’08*, pages 287–296,Beijing, China, April 2008.

[58] L. Parfeni, “uTorrent Used by Half of BitTorrent File-Sharers”, *Retrieved: 23 January 2012,* http://news.softpedia.com/news/uTorrent-Used-by-Half-of-BitTorrent-File-Sharers-222185.shtml

[59] A. Norberg, “uTP Protocol Specifications”, *Retrieved: 23 January 2012,* http://bittorrent.org/beps/bep_0029.html

[60] L. Deri ,”NTOP”, *Retrieved: 15 April 2012*, http://www.ntop.org/products/ntop/

[61] Wireshark Foundation, “Wireshark”, *Retrieved: 13April 2012*, http://www.wireshark.org

[62] Vuze Inc. “Vuze” (Formerly Azureus), *Retrieved: 26April 2012*, http://www.vuze.com/

[63] Bitcomet Development Group. ‘BitComet”, *Retrieved: 26 April 2012*, http://www.bitcomet.com/

[64] J.Sieka. “DC++”, *Retrieved: 26 April 2012,* http://dcplusplus.sourceforge.net/

[65] B. Cohen. “BitTorrent Client”, *Retrieved: 28 April 2012,* http://www.bittorrent.com/

[66] T. Silverston. O. Fourmaux, A. Botta, A. Dainotti, A. Pescap`e, G. Ventre, and K. Salamatian. “Traffic analysis of Peer-to-Peer IP T.V. communities”, *Computer Networks, vol. 53, no. 4*, pp. 470–484, New York, NY, U.S.A, 2009.

[67] R. Yuan, Z. Li, X. Guan, L. Xu. "An SVM-based machine learning method for accurate Internet traffic classification", *In Proc. Information System Frontier 12(2):149–156,Hingham*, MA, U.S.A, 2010.

[68] M. Varvello and M. Steiner. "Traffic Localization for DHT-based BitTorrent Networks", *In Proc. IFIP/TC6 Networking*, Valencia, Spain, May 2011.

[69] Sopcast Inc."Sopcast", *Retrieved: 6 May 2012*, http://www.sopcast.org

[70] Zhejiang University. "T.V.ants", *Retrieved: 6 May 2012*, http://T.V.ants.en.softonic.com

[71] PPStream Inc."PPStream", *Retrieved: 6 May 2012,* www.pps.T.V./en/

[72] "Sus/UnkPacker Trojan", *Retrieved: 6 May 2012,* http://www.sophos.com/en-us/threat-center/threat-analyses/suspicious-behavior-and-files/Sus~UnkPacker.aspx

[73] Juniper Inc. "JFlow", *Retrieved: 12 March 2012,* http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config2.html

[74] Inmon Corporation. "sFlow", *Retrieved: 6 March 2012,* http://www.sflow.org/

[75] Internet Engineering Task Force (IETF). "IPFIX Specification", *Retrieved: 28 April 2012,* http://tools.ietf.org/html/rfc5101