Peer-to-Peer Network Traffic Classification Based on Statistical and Behavioral Analysis

THESIS

Submitted in Partial Fulfilment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Submitted by:
JAGAN MOHAN REDDY DANDA
ID. NO. 2011PHXF011H

Under the Supervision of
Prof. Chittaranjan Hota

BITS Pilani
Pilani |Dubai |Goa |Hyderabad

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
2016

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **"Peer-to-Peer Network Traffic Classification Based on Statistical and Behavioral Analysis"** and submitted by **Jagan Mohan Reddy Danda** ID.NO. **2011PHXF011H** for award of Ph.D. degree of the institute embodies original work done by him under my supervision.

.....

DR. CHITTARANJAN HOTA
Professor,
Department of Computer Science,
BITS Pilani Hyderabad Campus,
Hyderabad, Telangana - 500 078

Date: .....

Dedication

To

My Mother, for her over-decades cultivating and enduring love.

My Father, who always takes me as his best pride.

My Brother, who always encouraged in his best.

Acknowledgements

I express my profound gratitude and sincere thanks to Prof. Chittaranjan Hota for his valuable suggestions, guidance, constant encouragement and intent supervision at every stage of thesis work. It has been a great learning process for me. It is in his association that gave me many opportunities to ameliorate my skills and knowledge.

I am thankful to Prof. N L Bhanumurthy, Head of the Department of Computer Science and Engineering, for all the necessary help I got during my thesis work. I am thankful to my DAC members Dr. Aruna Malapati and Prof. Y Yoganandam, who had been meticulous in reviewing my work time to time with their priceless suggestions.

I express my gratitude towards Prof. V S Rao (Director and Senior Professor of BITS Pilani Hyderabad campus and Acting Vice-chancellor of BITS Pilani) and Prof. M B Srinivas (Dean, Administration). I would also like to express my gratitude to Prof. P Yogeeswari (Associate Dean, Sponsored Research & Consultancy Division) and Prof. Vidya Rajesh (Associate Dean, Academic Research Division), for their constant support during my Ph.D work.

I am also indebted to Mr. Abhishek Thakur, for devoting his time and resources thus supplementing valuable inputs to my research work.

I am thankful to Mr. Ramesh Goud, who helped me with necessary networking setup during my Ph.D. course.

I owe my friends Rakesh Prasanna C, Muthukumaran K, Kiranmai G, Balaji V, Neha Singh T, Kasthuri I and Pratik Narang a token of appreciation for providing their support and encouragement that I received throughout my Ph.D. course.

I am obliged to Tata Consultancy Services (TCS), who funded my research work, Mr. Sitaram, TCS.

Last but not the least, I express my gratitude to my parents, brother and family for their constant support and unfailing guidance in whatever I did.

Abstract

The use of peer-to-peer overlay applications is growing dramatically, particularly for sharing video/audio, document files and software. The growth of such applications suggest that the use of P2P for illegal, malicious, and copyrighted data transfer traffic can have significant impact on the underlaying network. This resulting to reduced quality of service for other applications. It is therefore important to understand and characterize this traffic in terms of end-system behavior to provide network traffic planning. Detection and identification of P2P is one of the key tasks for Internet Service Providers.

In this research, a significant portion focuses on the problem of detecting peer-to-peer traffic from the web traffic, regardless of whether it is benign or malicious traffic. The classification of P2P traffic is challenging since traditional techniques, rely on mapping applications to well-know port numbers and payload data are ineffective against applications that use random ports or encryption. This research proposes three approaches to classify P2P traffic in real-time.

The first approach is based on statistical analysis of flow features, which are both port and payload agnostic. We extracted flow features from the network traces of P2P and NonP2P. We present a novel P2P traffic detection system using combined classifiers that integrate supervised ML algorithms to accurately differentiate between P2P and Nonp2P applications, and could also detect unknown P2P traffic. The results of proposed method can achieve high accuracy, outperforming comparable existing approaches to classify P2P network traffic. According to experimental results, we obtain the detection accuracy of more than 99.0% and a false positive rate of less than 0.1%.

The second approach, further enhances the host behavior mechanism by leveraging the host activity on the network to detect P2P flows and hosts in real-time. We proposed several heuristics to exploiting fundamental characteristics of P2P and NonP2P activity. These approaches rely on the connection patterns at transport layer, which does not need any machine learning algorithm. According to experimental results, we obtained an average detection accuracy of more than 96.55% and the false positive rate of 2.5%.

The third approach is based on Fuzzy pattern recognition system. In this approach, we classify P2P traffic based on the behavior. Features are extracted from the TCP and UDP headers. We propose several fuzzy membership functions to characterize P2P and NonP2P traffic. We obtain the detection accuracy of 98.89% for eMule application and the false positive rate of 1.11% and for the application μTorrent, detection accuracy of 97.51% and false positive rate of 2.49%. For NonP2P dataset, the detection accuracy of 99.96% and false positive rate of .04% is observed.

Contents

List of Tables **x**

List of Figures **xii**

1 Introduction **1**

1.1 Background 1

1.2 Overview of Peer-to-Peer Systems 3

1.2.1 What is Peer-to-Peer? 3

1.2.2 What is the Motivation for Peer-to-Peer? 4

1.2.3 Unstructured Networks 8

1.2.4 Structured Networks 9

1.2.5 Hybrid Networks 10

1.2.6 Gnutella 11

1.2.7 BitTorrent 14

1.2.8 Freenet 16

1.3 Issues with P2P overly 18

1.4 Background Study about Machine Learning 19

1.4.1 Supervised Learning 19

1.4.2 Unsupervised Learning 20

1.4.3 Performance Metrics for Classification 21
1.4.4 Feature Selection Techniques 21
1.5 Fuzzy Logic 26
1.5.1 Fuzzy Classification 27
1.6 Research Motivations 29
1.7 Research Contributions and Organization of Thesis 30
1.8 Thesis Organization 31

2 Literature survey **32**
2.1 P2P Traffic Classification 32
2.2 Classification Approaches 33
2.2.1 Port-based Approach 33
2.2.2 Protocol/Packet-level/Payload Based Approach 36
2.2.3 Connection Patterns at Transport-Layer 37
2.2.4 Flow based traffic Classification: Machine Learning 40

3 Data Collection and Preliminary Analysis **46**
3.1 Data Collection and Preliminary Analysis 46
3.2 Summary 59

4 Design of a privacy-preserving P2P traffic classifier **60**
4.1 Network Traffic Classification 60
4.2 System Overview 62
4.2.1 Background of Feature Selection 67
4.2.2 System Implementation 70
4.3 Result Analysis 74
4.4 Limitations 77
4.5 Conclusion 78

5 Host-Based P2P Traffic Identification **80**

5.1 Host-based P2P Traffic Identification using Heuristics 80

5.2 P2P and NonP2P Applications 81

5.2.1 NonP2P 83

5.2.2 P2P 84

5.2.3 Framework 84

5.2.4 Proposed Heuristics 85

5.3 Flow Based Vs Host Based approaches 91

5.4 Summary 92

6 P2P Traffic Identification: A Fuzzy Approach **93**

6.1 Proposed Fuzzy Recognition System for P2P Traffic Detection . . 93

6.1.1 Characterization of P2P Traffic 94

6.1.2 Behavior-based P2P Traffic Detection 96

6.1.3 Membership Functions for UDP and TCP Features 97

6.2 Performance Evaluation 100

6.2.1 Dataset Collection 100

6.2.2 Detection Accuracy 100

6.3 Conclusion 101

7 Design and implementation of a P2P-aware firewall **102**

7.1 P2P-aware firewall module 102

7.2 System Integration 104

7.3 Summary 109

8 Conclusion and Future scope of work **110**

8.1 Conclusion 110

8.2 Future Scope of the work 113

Index **114**

Bibliography **114**

List of Publications **122**

Biography **124**

List of Tables

1.1 Basic Message Types for Gnutella 12

2.1 Well-known port numbers used by several applications 35

2.2 Comparison of network traffic classification approaches 44

3.1 P2P Dataset 50

3.2 Application Signatures 51

4.1 Flow-based features 71

4.2 Attribute Selection with CSE and PCA 74

5.1 Application wise statistics 86

5.2 Flow vs Host based approaches 92

6.1 Feature Description 100

6.2 Detection Accuracy 101

List of Figures

1.1 Global Internet Phenomena of Asia-Pacific 2014 2

1.2 Global Internet Phenomena of Asia-Pacific 2015 3

1.3 An Example Overlay Network 4

1.4 Napster scenario 6

1.5 Peer to Peer Architecture 7

1.6 An unstructured P2P network 8

1.7 Structured P2P network 10

1.8 Hybrid P2P network 11

1.9 Gnutella Header 12

1.10 Searching and propagation of ping message in the Gnutella 14

1.11 BitTorrent file Swarm 15

1.12 Freenet distributed key-based routing 17

2.1 Inclinations of Applications and features [1] 33

2.2 Classification Approaches 34

2.3 Flow example, Host A opens HTTPS (443), HTTP (80) and default port for Kazaa (1214). Host A and Host B packets are grouped into either single TCP connection or two bi-flows or four flows. On the other hand, Host A again opens new port on both TCP and UDP, packets can be grouped into single TCP and UDP connection. . . 40

3.1 Network architecture of BITS-Pilani Hyderabad Campus 47
3.2 Network architecture of BITS-Pilani Hyderabad Campus 48
3.3 Logs for Internet traffic generated on one day at the university of the authors 49
3.4 Testbed for impact analysis of P2P on IPS/IDS 50
3.5 Payload-based detection of P2P traffic using Snort IDS 52
3.6 Virtualized Environment running P2P & NonP2P traffic at BITS Pilani, Hyderabad campus 53

4.1 Stacked Learning 66
4.2 Stacked Learning 67
4.3 Proposed Detection Framework using ensemble learning 69
4.4 Accuracy and Recall of Stacking and Voting ensemble learning with NB, BN and C4.5 76
4.5 Accuracy and Recall of Stacking and Voting ensemble learning with NB, BN, C4.5 and RF 77
4.6 Build Time of Staking and Voting with Full and Two subset Features 78

5.1 Framework for P2P Traffic Identification 85
5.2 Heuristic A 87
5.3 Heuristic B 88
5.4 Heuristic C 88

5.5 Heuristic D 89
5.6 Heuristic E 90
5.7 Detection rate of P2P traffic 91

6.1 UDP packets sent by NonP2P applications. 94
6.2 Peer Discovery/control messages sent by P2P Application over UDP. 96
6.3 TCP behavior observed in P2P applications. 97
6.4 TCP behavior observed in NonP2P applications. 98
6.5 Procedure to extract Host feature 99

7.1 Overview of System Integration 104
7.2 A snapshot of the P2P classification module invoked in parallel . . 105

Chapter 1

Introduction

1.1 Background

In recent times, peer-to-peer (P2P) overlay networks have become popular for file sharing applications. In the context of an overlay, each host is known as a peer. The behavior of the P2P networks, which connect the peers on top of a physical network, like IP, is growing dramatically. Client-server models are categorized by an asymmetric relation between client and server where client sends request and server responds back. But P2P networks are distributed in nature and for file sharing, each peer acts as a client as well as a server. These overlay networks build a logical network at application layer providing connectivity, routing and messaging among addressable end points. These types of networks have been used for voice-over IP like Skype and streaming media over P2P like IPTV in recent times.

With the rapid development of P2P technology, the P2P traffic has accounted for 40-60% of Internet traffic [2]. Schulze et al. [3] observed that P2P traffic is

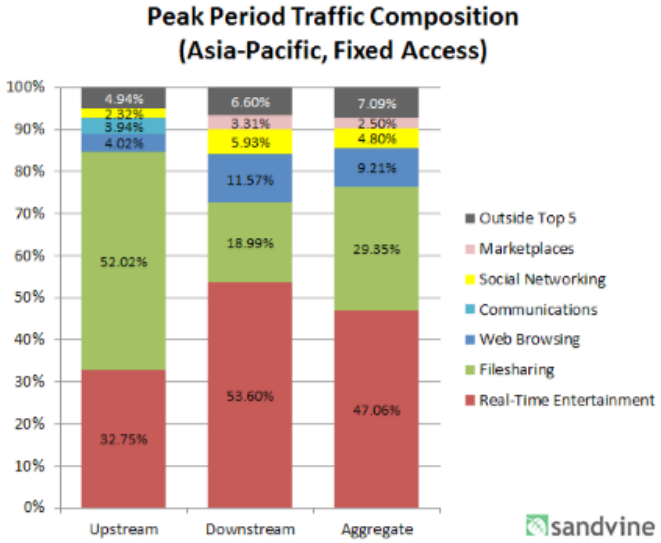


Figure 1.1: Global Internet Phenomena of Asia-Pacific 2014

responsible for 69.95% of the global Internet traffic and BitTorrent [4] is the most used protocol. In September 2015, global Internet phenomena report by Sandvine [5] points to the increased percentage of aggregated P2P traffic of Asia-Pacific fixed access networks from 29.35% in 2014 to 29.76% in 2015 and BitTorrent accounts 58.57% of total upstream and 22.65% of total downstream traffic during peek period is shown in Fig. 1.1 and Fig. 1.2.

There are large number of applications such as Napster [6], Gnutella [7], Kazaa [8], eDonkey [9] , Freenet [10], Skype [11] etc. developed for deploying P2P technologies in the Internet. There is an exponential increase of users taking interest in these applications. Researchers have contributed to designing large number of models based on P2P overlay networks, suiting to different requirements such as file sharing, anonymity, media streaming, and voice over P2P etc.

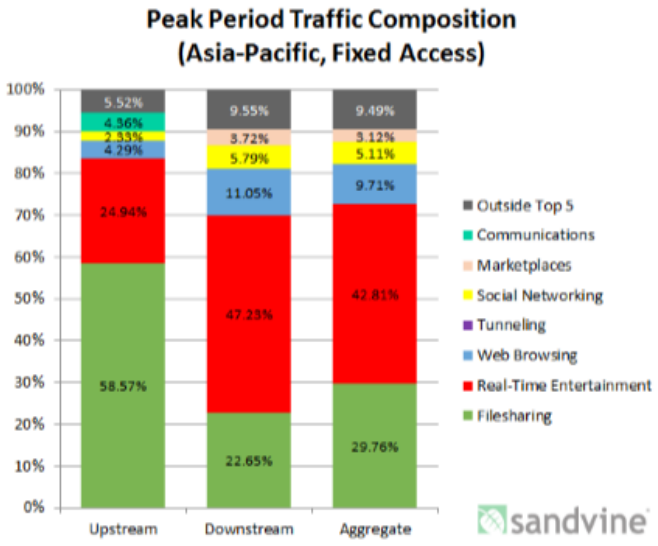


Figure 1.2: Global Internet Phenomena of Asia-Pacific 2015

1.2 Overview of Peer-to-Peer Systems

1.2.1 What is Peer-to-Peer?

Peer-to-peer is defined as a network of interconnecting nodes (called peers) that partitions the tasks or work load and access resources (CPU cycles, Storage, content) in the network. Peer systems are scalable and fault tolerant in their approach because they offer no single points of failure, and the network can grow and shrink without sacrificing the functionality of the system [12].

Client-server based architectures are characterized by asymmetric relationship between client and server where client queries and server responds. In contrast, in distributed P2P networks every node acts as both a server and a client. Peer-to-Peer network is a form of virtual network on top of a physical network, where the nodes form a subnet of the nodes in the physical network. But the application layer peers directly communicate at the logical network via directly over the underlying network. Overlay network provides connectivity, routing, and messaging amongst

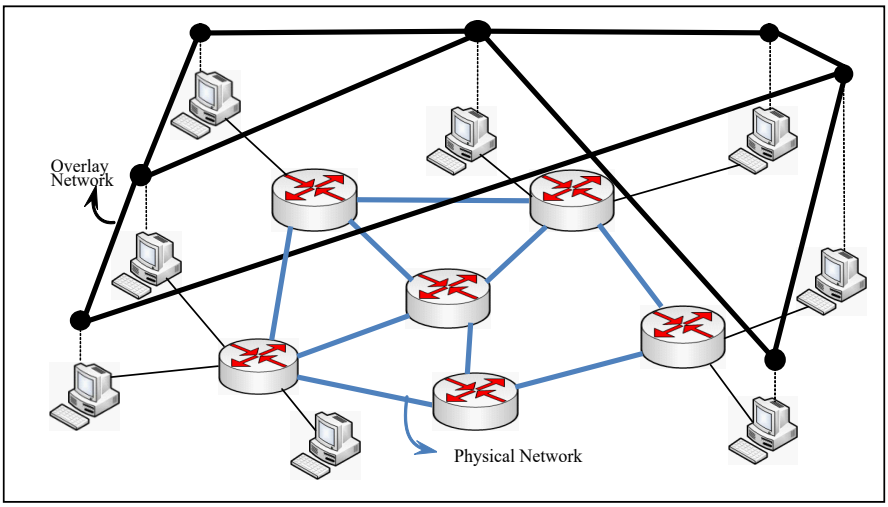


Figure 1.3: An Example Overlay Network

addressable end-points of the communication as shown in Fig. 1.3.

1.2.2 What is the Motivation for Peer-to-Peer?

The motive behind P2P applications have transformed the typical user's experience of getting content and communication services from the Internet and at the same time other Internet applications have not been built with P2P. P2P offers a uniquely self-scaling architecture and provide services at a low cost level where client/server architectures are not achievable.

Peer-to-Peer (P2P) overlay networks which connect peers on top of a physical network, be it over Internet Protocol (IP), Asynchronous Transfer Mode (ATM), Frame Relay (FR), Systems Network Architecture (SNA), Internetwork Packet Exchange (IPX) etc., are growing dramatically in their usage. P2P application usage has grown steadily since its inception. Client-server based architectures are characterized by asymmetric relationship between client and server where client queries and server responds. Contrast to that in distributed P2P systems every

node acts as both a server and a client. Peer-to-Peer overlay structure is brought to popularity by the file sharing applications like Napster [6], Gnutella [7] etc.

These overlay networks have been used for file sharing, Voice over P2P, and streaming media over P2P in recent times. Motivated by the extent of their usage, researchers are eager to study and improve the scalability and performance of these networks. Unlike the client-server model, a P2P network connects several peers directly. The architecture of a P2P network is determined by the characteristics of its overlay network, placement and scope of data, and the protocols used for communication. The choice of the architecture influences how the network can be used for various tasks like searching and downloading.

These P2P networks are distributed in nature and are named as `centralized` and `decentralized` networks. Centralized P2P networks are like a client-server model, but the server keeps index media files of all peers in the network. The drawback of this approach is single-point of failure. Decentralized P2P networks are scalable and fault tolerant because they offer no single-point of failure, and the network can grow and shrink without sacrificing the functionality of the system. Napster [6] was the first centralized distributed overlay network designed for music file sharing, the file indexes are stored on a central server and when a new peer searches a file .mp3/.mp4 it has to get the information from the central index server. The index server finds which peer has the corresponding file and informs the requester. Then file is directly downloaded from that peer. This process is shown in Fig. 1.4.

There are two classes of P2P overlay networks: structured, and unstructured. An unstructured P2P system consists of peers joining the P2P network with some loose rules, without any prior knowledge of the topology. Unstructured P2P

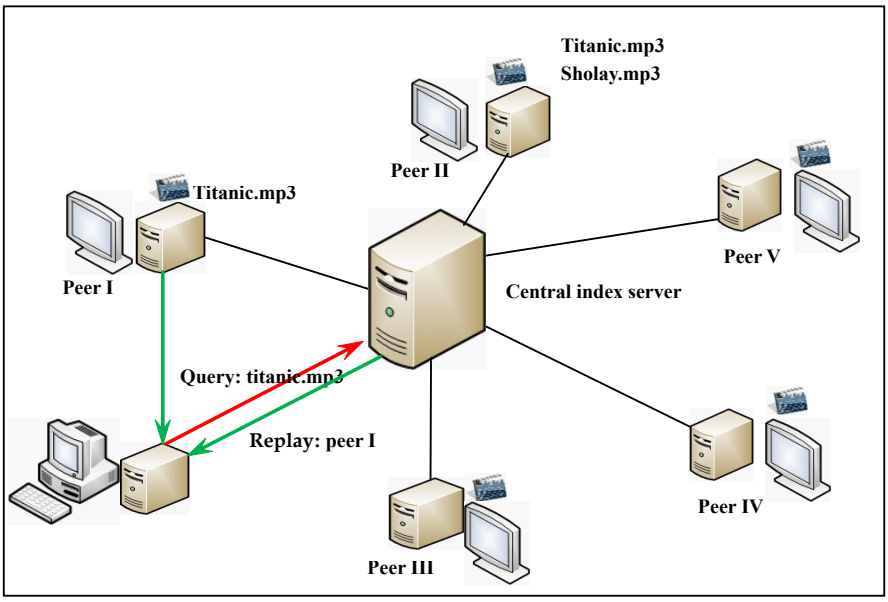


Figure 1.4: Napster scenario

networks such as Freenet [10], Gnutella [7], KaZaA [8], Direct Connect++ [13] offer decentralization and simplicity, but may require $O(N)$ hops to search a file when the network is made up of N nodes. In contrast, structured P2P overlay networks tightly control both the network topology and the placement of the content. Specifically, the content is stored at specified locations based on Distributed Hash Tables (DHTs), so as to improve the efficiency of the queries. Structured P2P overlays using DHTs like Content Addressable Network (CAN) [14], Chord [15], and Pastry [16] are well suited for large scale distributed applications because of their search efficiency which is $O(logN)$ for a network of N nodes.

In practice the functions of P2P networks is divided into three main components: (i) Routing and Messaging, (ii) Search and Content storage, (iii) Configuration and Peer role selection. Routing API, each peer maintain the state of some connection to other (neighboring) peers within the overlay, which include peer discovery and other peer state maintenance. The overlay messaging API, provides service at application layer that can exchange messages with other peers in the overlay and

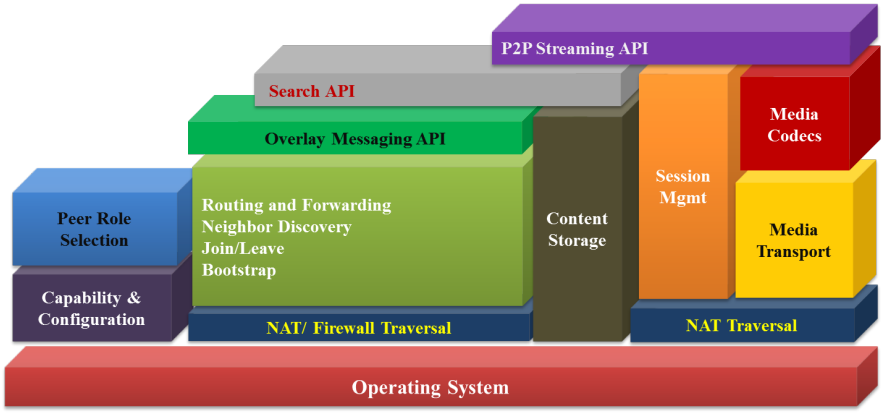


Figure 1.5: Peer to Peer Architecture

messages received from other peers can be forwarded to destination peer using message-forwarding function. Bootstrap provides the initial configuration information of a peer to join the overlay network. Join and leave functions allow peers to join and leave the overlay network whenever they wish to do so.

The content storage functionality at a peer should facilitate access to the stored object locally as well as by other peers using improved search indices and a query interfaces. Peers should also self-configure and assess their capabilities based on resource availabilities and stability, and further select their own role to either act as a super-peer or an ordinary peer in the overlay. The self configurable nature of a peer and the dependence of peers with each other allows malicious peers to abuse the trust. As internals are exposed to fellow peers in the name of sharing or distributing the workload, attackers can leverage this in compromising the P2P networks and hence creating havoc to other users. These networks also provide session and media management into overlay for P2P streaming applications like Session Initial Protocol (SIP), Real Time Protocol (RTP) and Voice Over P2P (VoP2P). The P2P architecture is shown in Fig. 1.5 with several such functions being performed at different layers.

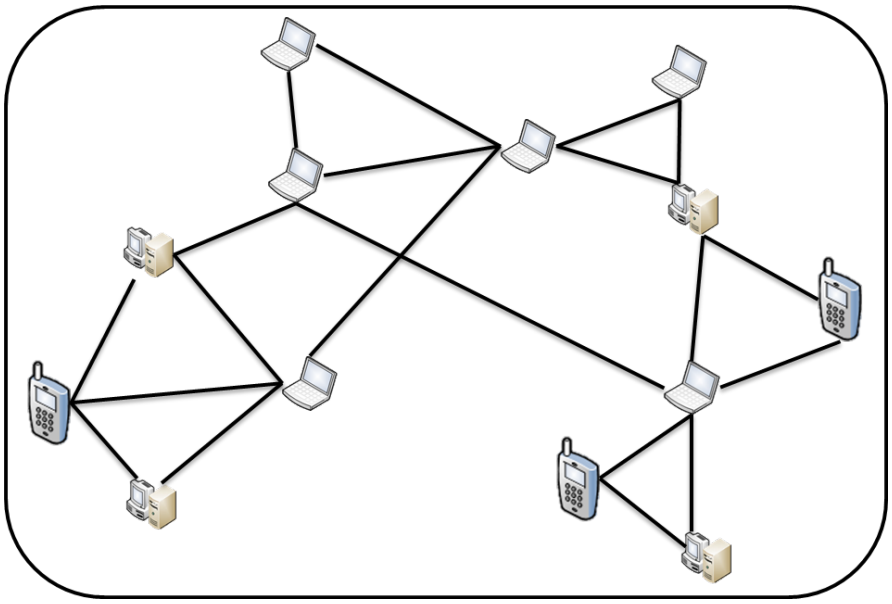


Figure 1.6: An unstructured P2P network

Based on the *topology* of the peers in the network, P2P networks are classified as being `Unstructured`, `Structured` or `Hybrid` [17].

1.2.3 Unstructured Networks

Peers in an unstructured P2P network are organized in a random graph. The links between nodes are established arbitrarily and hence there is no correlation between a peer and the content being managed or shared by it. An example is given in Fig. 1.6. Unstructured P2P networks use flooding, random walks or expanding Time-to-Live (TTL) search on the graph to query content stored by the participating peers [17]. If a peer wants to find some piece of information in the network, the query has to be flooded through the network in order to find as many peers as possible sharing that information. In such a system, the network is easy to construct.

1.2.4 Structured Networks

In contrast to the unstructured P2P networks, structured P2P overlay networks have a tightly coupled topology. The content in such systems is not placed at a random peers but rather at specified location. Every data item in the overlay network is assigned a (key, value) pair, and peers are organized into a graph that maps each data-key to a peer. This enables efficient discovery of data items using the key of a data element [18]. Structured P2P systems are based on Distributed Hash Tables (DHTs), which are decentralized and distributed systems provide a lookup service similar to a hash table.

Fig. 1.7 is an example of *P-Grid overlay*. Each pair of peers is responsible for keys with the indicated prefix as shown in the labelled tree. In addition, each peer has a routing table or cache that associates key prefixes with other peers' identifiers. *P-Grid* algorithm uses the local routing table in the peer to match the nearest prefix and it will performs separate checking to see whether the peer is still online. To update routing table in *P-Grid*, it uses public key mechanism for authenticating peer-to-peer interactions when needed. With the DHT data structure and algorithm, peers can easily map data-keys to nodes. This facilitates faster look up for any data object in a small number of overlay hops. These networks provide a cooperative, stable, and robust mechanism for storing and retrieving content. Some of the prominent structured P2P systems, which are self-organized include the Chord, Kademlia, PASTRY etc.

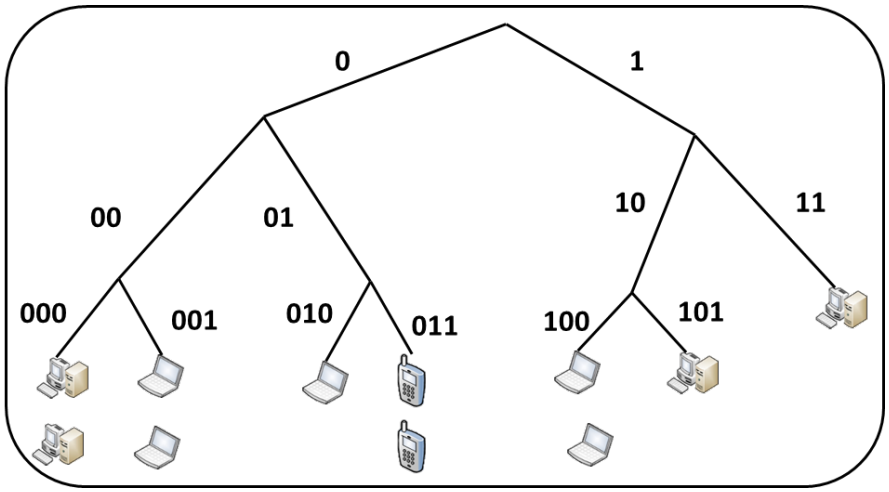


Figure 1.7: Structured P2P network

1.2.5 Hybrid Networks

Hybrid P2P systems are not pure P2P networks. They combine the traditional client-server model with the P2P model. Hybrid P2P systems require the presence of one (or multiple) server-like central entity, without whose presence the network will not sustain. An example of Hybrid P2P systems is given in Fig. 1.8. *Hierarchical* approaches introduce the notion of "super-peers" or "ultra-peers". The majority of overlay responsibilities (related to routing, indexing, etc.) is assigned to a small subset of these more powerful nodes. A super-peer may be chosen based on the participation/contribution of the peer in the network, its uptime, bandwidth, publicly visible IP address, etc.

The hybrid P2P networks are having two categories of peers called `Super Node` (SN) or `Ultra Peer` (UP) and `Ordinary Nodes` (ON) or `Normal Peers` (NP). The super peers are very powerful in terms of the capability of the peer and are controlled by several ordinary peers and these ONs become peers in the overlay network to maintain the P2P structure and to share the files. Once the super node goes down then among one of ON as become the super node. These networks use

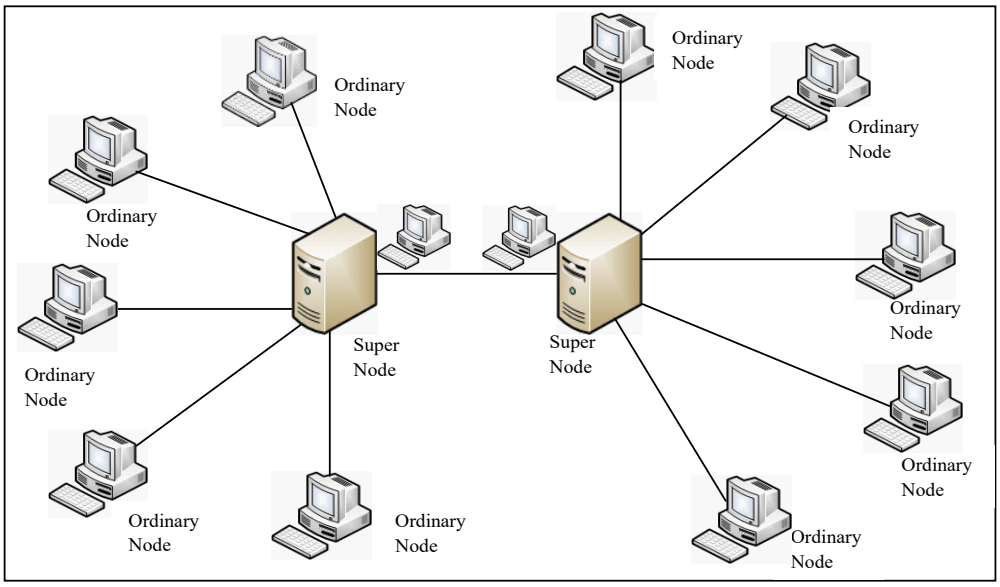


Figure 1.8: Hybrid P2P network

central control that is generally called as centralized networks. Example networks are KazaA [8] and eDonkey [9] or eD2K, Direct Connect++ (DC++) [13].

P2P overlays may also be classified based on the *dependency* of peers [19]. *Flat* or pure P2P networks treat all participating peers equally. That is, all peers enjoy the same share of resources in the network and have similar responsibilities. In the next section, we examine the most dominant P2P protocols in reality today that are Gnutella, BitTorrent and Freenet for content delivery networks in terms of their protocol design for providing an insight to the reader.

1.2.6 Gnutella

Gnutella was the first pure P2P file-sharing application and one of the popular protocol. In this network, the peers join in an unstructured overlay using flooding for routing. Most recent Gnutella protocol adopted a *superpeer/ultrapeer* model which is of high-capacity and stable. All the queries are routed using a flooding

Table 1.1: Basic Message Types for Gnutella

Message Type	**Description**
Ping	Used to discover other peers in the Gnutella network
Pong	The response to a Ping. Provides the IP address and part number of the host and extensions supported by the peer.
Query	Search for a file. Specifies the minimum transfer speed of the peer and the search criteria. The search criteria is text, such as a string of keywords.
QueryHit	Response to a query. A peer returns query hit responses to previously forwarded queries back along the connection from which the query was received.
Push	Download a request for firewalled peers.
Bye	Tell the remote host that the connection is being closed.

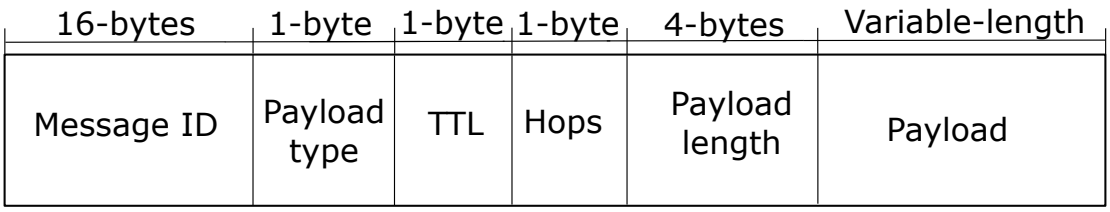


Figure 1.9: Gnutella Header

mechanism between superpeers. Each ultrapeer maintains connection to a set of other ultrapeers. To join any peer in the Gnutella network, one has to contact a known peer of the network and request to establish a Gnutella protocol connection to the remote peer. Once a peer is connected to the network, it receives information about other peers through the protocol messages as shown in Table 1.1. Fig. 1.9 shows the header format of Gnutella and each message has the following fields:

A. ***Message ID***: A 16-byte field contains a globally unique message ID.

B. ***Payload type***: A 1-byte field containing the message type.

C. *Time to Live (TTL)*: A 1-byte field is decremented by each peer receiving the message until the TTL is 0. This value is no larger than 3.

D. ***Hops***: A 1-byte field incremented by each peer receiving the message and that indicates the number of hops the message has travelled so far.

E. ***Payload Length***: A 4-byte field containing the number of bytes in the remainder of the message.

F. ***Payload***: A variable-length field, the contents of which are message dependent.

An example Gnutella network is shown in Fig. 1.10. Messages received by a peer are forward to other peers that are connected to it. In this way, messages are flooded in the network to ensure that it reaches it's intended destination. To control the scope of flooding of Gnutella messages, the TTL value is set in the descriptor header specifying the number of times the message should be forwarded before it can be discarded. When a peer receives a 'ping' message from a new peer in the network, it replies back with a 'pong' message and decrements the TTL value of the message. The ping message is forwarded to other neighboring nodes which reply with a 'pong' and decrement the TTL value. Once the TTL value of the message is zero, the message is dropped from the network [20]. Distributed discovery of resources in the Gnutella P2P network works in much the same way as sending ping/pong messages in the network; the initial query is generated by a peer and sent to neighbor peers directly connected to it. These neighbor peers evaluate and then forward the query to other neighbor peers. Peers that receive the query message evaluate it against its own data store and if they have resources that match the query respond back with a 'QueryHit' message to the

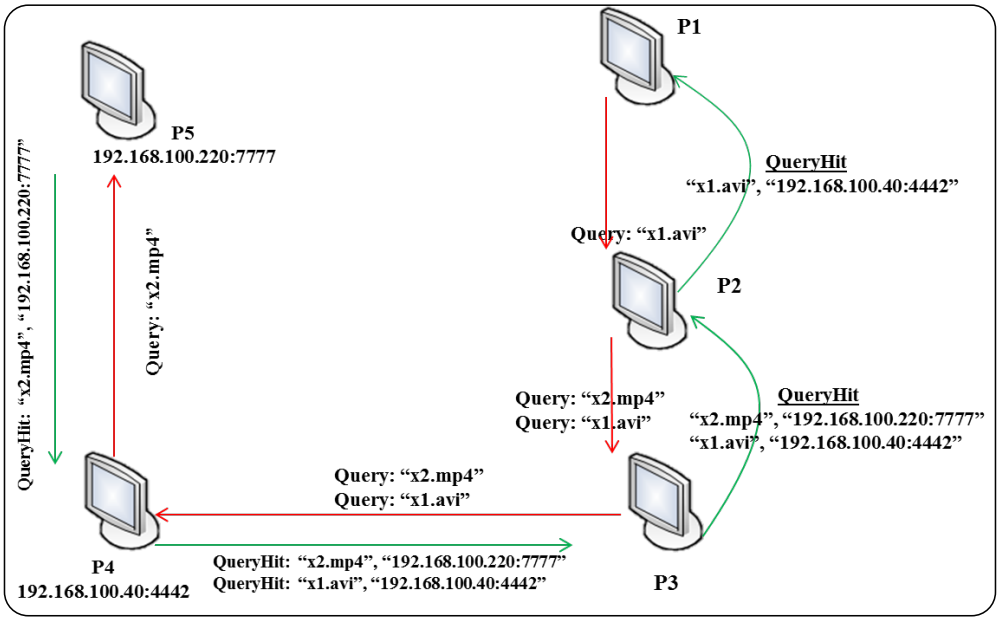


Figure 1.10: Searching and propagation of ping message in the Gnutella

original peer. Once the query initiator has aggregated the responses to its original query, it initiates download of the resource by contacting the source peer in the network. The Gnutella protocol does not support file download over its application level protocol rather a target peer must contact the source peer directly using IP routing to establish a HTTP connection to download the file .

1.2.7 BitTorrent

BitTorrent protocol is designed for distributing large files using mutual distribution of the pieces between a set of peers called as a `swarm` [21]. Peers join a swarm to download a file and leave the swarm after the file download is complete. The downloaded file is available at peer called as `seed`, which provides a *.torrent* file corresponding to the content of the file. The .torrent file provides the information about individual fixed-size pieces of the file called *tracker*. When a request comes from a P2P client to a tracker, the tracker determines other peers that are already

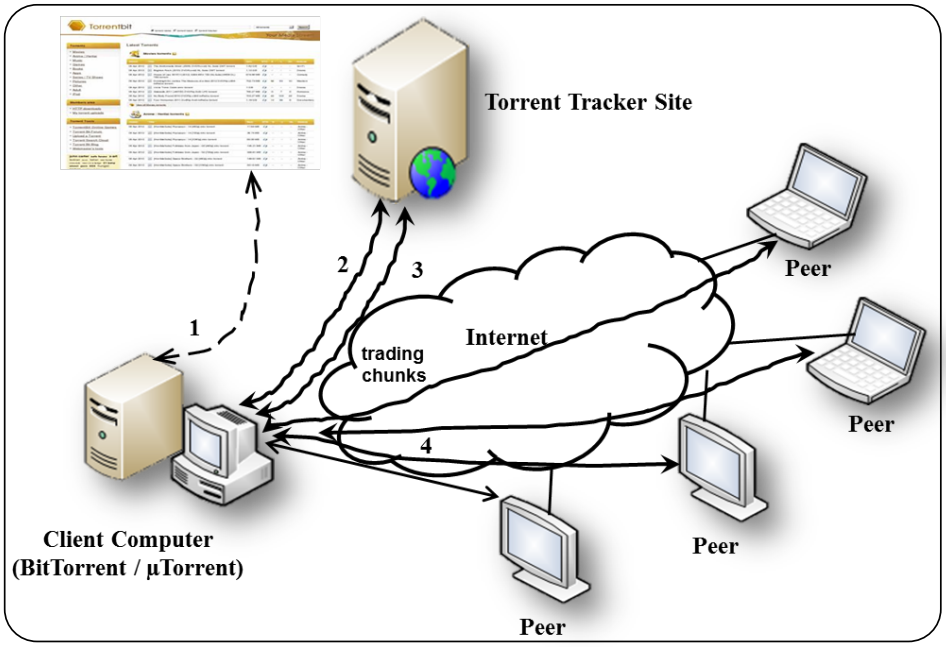


Figure 1.11: BitTorrent file Swarm

in the swarm to access the piece of the content as shown in Fig. 1.11.

BitTorrent protocol has the following message types:

A. ***Choke***: The initial state of a peer, which signifies that it is not sending data to the neighbor peer.

B. ***Unchoke***: The neighbor peer is to receive pieces from this peer.

C. ***Interested***: The peer is willing to receive pieces from this neighbor.

D. ***Not interested***: The peer is not willing to receive pieces from this neighbor.

E. ***Have***: The piece that this peer, is most recently downloaded and verified.

F. ***Bitfield***: A bitfield that has a 1 bit in each position, indicating pieces already sent by this peer.

G. ***Request***: Identifies a piece being requested for download.

H. ***Piece***: Identifies a piece being sent by a neighbor peer.

I. ***Cancel***: A flow control mechanism to cancel previously requested pieces, typically used near the completion of a file download.

For file downloads to be successful in Bittorrent, atleast one peer (seeder) with the complete file must participate in the swarm until the complete set of file pieces have been fully disseminated in the network, as a result of which Bittorrent to efficiently handles hot spots or requests for large files [22]. In the case of a large file, the seeder must stay on-line long enough until atleast one peer has successfully downloaded the file or subsets of the full set of file pieces in available on peers in the network. Eventually as more peers in the system complete their download and remain as seeders in the network, request for file start to complete quickly since each new peer participating in a file swarm adds additional upload bandwidth.

1.2.8 Freenet

Freenet is a pure P2P decentralized network for security, anonymity and deniability [21]. Freenet operates as an unstructured network of nodes that poll their unused disk space to store data files in the network and cooperate to route request to the most likely physical location [23]. Nodes in Freenet are arranged in a loosely structured overlay network similar to Gnutella. The nodes are connected to a set of neighbor nodes which they route message to/from. Peers join the Freenet network by discovering address of one or more existing members of the network. Again like the case of Gnutella, discovery of known peers in the network is not part of the protocol definition although out-of-band means downloading a list of peers from a server or manually supply the Freenet node with an address of peers to contact, are popular ways to setting up initial connection to the net-

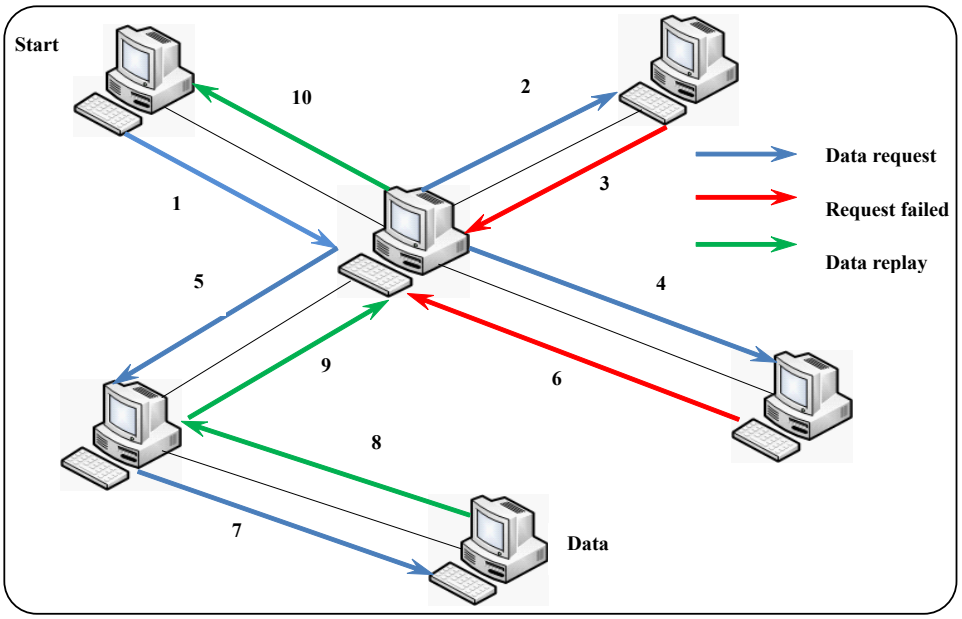


Figure 1.12: Freenet distributed key-based routing

work. After a remote Freenet node is contacted, a secure channel is built over the TCP/IP transport layer and the new node sends a 'Request.HandShake' message to the remote peer to initiate a Freenet connection. If the remote peer is active, it can reply back with a 'Reply.HandShake' message indicating that it accepted the connection request.

Searching in Freenet networks only support exact match queries using the file keys. It is not possible to perform a range/blind search in the network. To search in the network, a user posts a query to its local node containing the descriptive text of the file to be retrieved from the network. The node hashes the descriptive text to obtain a file key and the routing algorithm is used to determine the likely location of the file in the system. If no match is found, a 'Request.Data' message is routed to a next hop node till a 'Reply.NotFound/Send.Data message' is received or when TTL value reaches 0. In the case that a 'Reply.NotFound' message is returned, the originating node tries to reach the second next hop for the file key.

Once a match for the file key is found, a Send.Data message, containing the file requested, is routed from the source node to the querying node by back-tracing along the path through which the query was routed which is as shown in Fig. 1.12.

1.3 Issues with P2P overly

In order to access and share files on ones computer within a P2P network, one must open a specific TCP port through the firewall for the P2P software to communicate. In effect, once firewall has opened the port, one can no longer be protected from malicious traffic coming through it. Another security concern is that when one downloads files from other peers, one doesn't know for sure that the file is legitimate. P2P systems lack the tools available to a centralized administrator, making it much more difficult to implement security protections. Because the P2P file sharing content represents a relevant traffic share, network operators and service providers would like to detect, classify, regulate and may charge for carrying this content. Moreover, P2P traffic has many characteristics that overlap with malicious traffic; e.g., multiple persistent high-throughput flows (similar to spyware), communication with centralized p2p trackers (similar to botnet communications), large number of simultaneous peer connection requests, many of which are unsuccessful due to peer-churn (similar to self-propagating malware infections and port-scan attacks), communication on uncharacteristic ports, receiving requests from a peer and forwarding those requests to neighbors immediately, trying to connect using both TCP and UDP ports etc. Consequently, in addition to its relevant network resource consumption, the growing of P2P traffic has serious implications for network security devices such as intrusion-detection systems (IDS),

firewalls, policy/SLA enforcers etc.

1.4 Background Study about Machine Learning

In this thesis, we deliberate different Machine Learning and Fuzzy pattern classification approaches. ML is a powerful technique in the area of data mining which finds useful patterns from the given data. ML techniques are broadly categorized into two learning groups: Supervised and Unsupervised learning. **Supervised Learning** uses a training data to classify unseen examples from the model file. **Unsupervised Learning** doesn't rely on the training dataset rather it groups the instances into clusters that have similar characteristics.

1.4.1 Supervised Learning

Supervised learning is the machine learning task of inferring a function from labelled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances [24]. This requires the learning algorithm to generalize from the training data to unseen examples in a reasonable way. Examples of supervised learning algorithms are, Decision trees, Ensembles (bagging, Boosting, Random Forest), k-NN, Linear Regression, Neural Networks and Logistic Regression etc.

1.4.2 Unsupervised Learning

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. By contrast with Supervised Learning or Reinforcement Learning, there are no explicit target outputs or environmental evaluations associated with each input; rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output [25].

The only things that unsupervised learning methods have to work with are the observed input patterns x_i, which are often assumed to be independent samples from an underlying unknown probability distribution $P_I[x]$, and some explicit or implicit *a priori* information as to what is important.

Density estimation techniques explicitly build statistical models (such as Bayesian Network) of how underlying causes could create the input. Feature extraction techniques try to extract statistical regularities (or sometimes irregularities) directly from the inputs. However unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Many methods employed in unsupervised learning are based on data mining methods used to pre-process data. Examples of unsupervised learning algorithms are, Clustering(k-means, mixture models, hierarchical clustering), ExpectationMaximization algorithm (EM), Principal Component Analysis (PCA), Independent Component Analysis (ICA), Singular Value Decomposition (SVD).

1.4.3 Performance Metrics for Classification

To measure the performance of a classification algorithm, different metrics can be used. The metrics employed in this work are briefly discussed below:

- **Accuracy:** Accuracy of the classifier is the ratio of sum of true positives (TP) and true negatives (TN) and to the sum of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) for all classes.

$$Accuracy\ in\ \% = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \tag{1.1}$$

- **Detection rate**, a.k.a True Positive rate, is the ratio of TP over the sum of TP and FN, or the ratio of relevant records retrieved to the total number of relevant records.

$$Detection\ rate\ in\ \% = \frac{TP}{TP + FN} \times 100 \tag{1.2}$$

1.4.4 Feature Selection Techniques

Feature selection is the process of reducing the number of features, with the aim of removing those features from the learning algorithm which have low impact on the classification problem.

Primary motivation behind feature selection is that the training data contains many features which are either redundant to the classification problem, i.e., they provide no further information than the currently selected features, or are totally irrelevant to the classification problem itself. Consider the simple example of features that may be extracted from network flows for the purpose of classification

of network traffic. Many features that can be extracted for this purpose- such as average packet size, duration of the flow, number of unique ports used, etc. But certain features are not at all relevant to the task of classifying network traffic. For example, every TCP packet receives an acknowledgement in response. A count for the number of ACK packets is not a good feature for classifying network traffic because it cannot help in distinguishing between different applications.

Reducing the number of features provides direct benefit in terms of lesser training time for the learning model. Such 'feature selection' is also known to reduce the problem of 'over-fitting' or the variance error [26], and is also critical to overcome the class imbalance problem [27]. It should be noted that although the final accuracy of the learning algorithm will depend on the learning technique used, suitability of the original features obtained (i.e., those obtained prior to use of feature selection) etc., feature selection techniques are effective in optimizing the performance of the classifier since the number of features used for classification are reduced.

Correlation-based Feature Selection

Correlation-based Feature Selection (CFS) algorithm is a simple filter method. Given a full set, it finds an optimal subset that contains features that are highly correlated with class label and uncorrelated with each other. The 'class label' field in the training set is the target value of that particular instance of the training set [28].

CFS evaluates correlation of the feature subset on the basis of this hypothesis- "A good feature subset contains features highly correlated with (predictive of) the classification, yet uncorrelated with (not predictive of) each other" [26]. This

hypothesis relies on two metrics- one is the correlation of the feature with the class, and other is inter-correlation amongst features. The 'feature-class' correlation indicates how much the feature is correlated with its class, while 'inter-correlation' amongst features tells about correlation between any two features.

$$M_s \quad = \quad \frac{k\bar{r_{cf}}}{\sqrt{k \quad + \quad k(k-1)\bar{r_{ii}}}} \tag{1.3}$$

The above equation calculates the merit values (M_s) for each subset of k features. For each value of k, all possible subsets are chosen and merit values are computed. The subset giving the highest merit is the output of CFS. In Equation 1.3, $\overline{r}_{cf}$ is the average correlation between feature and class, and $\overline{r}_{ii}$ is the average inter-correlation between two features. The heuristic metrics $\overline{r}_{cf}$ and $\overline{r}_{ii}$ are calculated as the Symmetrical Uncertainty (SU) [26] given in Equation 1.4.

$$SU \quad = \quad 2.0 \times \left[\frac{H(X) + H(Y) - H(X,Y)}{H(X) \quad + \quad H(Y)}\right] \tag{1.4}$$

where H(X) is defined as entropy.

Entropy: In information theory, entropy (Shannon entropy) is a measure of uncertainty or disorder. Shannon defined the entropy H of a discrete random variable X with possible values $x_1, x_2, ...x_n$ and probability mass function $P(X)$, as given below:

$$H(X) \quad = \quad E[I(X)] \quad = \quad E[-ln(P(X))] \tag{1.5}$$

Here, E is the expected value operator, and I is the information content of$X$ [29].

$I(X)$ is itself a random variable or it can also be written as:

$$H(X) = -\sum_{x \epsilon X} p(x) log_2(p(x)) \tag{1.6}$$

Consistency-based Subset Evaluation

The consistency-based Subset Evaluation (CSE) search algorithm [30] evaluates the feature subsets and finds an optimal subset of relevant features that are consistent to each other. To determine the consistency of a subset, the combination of feature values representing a class are given a pattern label. All instances of a given pattern should thus represent the same class. A pattern is *inconsistent* if there exist at least two instances such that their patterns are same but they differ in their class labels. The overall inconsistency of a pattern p is calculated by Inconsistency Count (IC):

$$IC(p) = n_p - c_p \tag{1.7}$$

where n_p is the number of instances of the pattern p, and c_p is the number of instances of the majority class of the n_p instances.

The overall consistency of a subset S is calculated using Inconsistency Ratio (IR). IR is the sum of all inconsistency counts over all the patterns of the feature subsets divided by the total number of instances in the dataset:

$$IR(S) = \frac{\sum_p IC(p)}{\sum_p n_p} \tag{1.8}$$

Principal Component Analysis

Strictly speaking, Principal Component Analysis (PCA) is not a feature *selection* but a feature *reduction* technique. PCA is a dimensionality reduction technique which reduces the initial number of features to a smaller number of uncorrelated features, which are calculated as the linear combination of the original features. Since PCA does involve calculation of a smaller set of features from the original, larger set, it also qualifies as a feature 'selection' technique.

PCA aims at reducing the dimensionality of the dataset. It transforms higher-dimensional data into lower-dimensional space by identifying Principle Components (PCs) which are uncorrelated and better captures the variability of the data [31]. The first dimension is chosen so as to capture as much variability as possible. The second dimension is orthogonal to the first, and it aims to capture as much remaining variability as possible subject to the constraint imposed by the first, and so on [32]. The mathematics behind PCA is briefly described here.

Given a $N \times M$ matrix X. Where N is the number of training samples and M is the number of features. The mean μ of the set of N training samples (corresponding to a column vector) is given by

$$\mu = \quad \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1.9}$$

Matrix X is preprocessed such that each column has zero mean. This is done by subtracting μ from each x_i.

$$w_i = \quad x_i - \mu \tag{1.10}$$

Further, the covariance matrix is defined by

$$C = \frac{1}{N}\sum_{i=1}^{N} w_i.{w_i}^T \tag{1.11}$$

If $\lambda_j, j = 1, 2 \ldots n$, are the eigenvalues of C, then the eigenvalues can be ordered such that $\lambda_1 \leq \lambda_2 \ldots \lambda_n$, where n is the number of eigenvalues. Further, let matrix $Y = v_j, j = 1, 2 \ldots n$ be the eigenvectors of C. These are ordered in such a way that i^{th} eigenvector corresponds to the i^{th} largest eigenvalue. Finally, the columns of matrix XY are a linear combination of the original attributes. These new attributes are called principal components [32]. The top k principal components are chosen which capture the variability of the data up to a predefined threshold (taken as 95% in the Weka machine learning tool [33]).

1.5 Fuzzy Logic

The term *fuzzy logic* was first introduced by Lotfi Zadeh (1956). Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. By contrast, in Boolean logic, the truth values of variables may only be 0 or 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between *absolutely true* and *absolutely false* or some intermediate truth degree [34]. In the analogy to various definitions of operations on fuzzy sets (intersection, union, complement, etc.) one may ask how propositions can be combined by connectives (conjunction, disjunction, negation, etc.) and if the truth degree of a composed proposition is determined by the truth degrees of its components, i.e. if the connectives have their corresponding truth functions [35].

Fuzzy logic and probability, both address different forms of uncertainty. While both fuzzy logic and probability theory can represent degrees of certain kinds of subjective belief, fuzzy set theory uses the concept of fuzzy set membership, i.e., how much a variable is in a set (there is not necessarily any uncertainty about this degree), and probability theory uses the concept of subjective probability, i.e., how probable is it that a variable is in a set (it either entirely is or entirely is not in the set in reality, but there is uncertainty around whether it is or not). The technical consequence of this distinction is that fuzzy set theory relaxes the axioms of classical probability, which are themselves derived from adding uncertainty, but not degree, to the crisp true/false distinctions of classical Aristotelian logic.

1.5.1 Fuzzy Classification

Fuzzy classification is the process of grouping elements into a fuzzy set [34] whose membership function is defined by the truth value of a fuzzy propositional function.

A fuzzy class $\sim C = \{i| \sim \prod(i)\}$ is defined as a fuzzy set $\sim C$ of individuals i satisfying a fuzzy classification predicate $\prod$ which is a fuzzy propositional function. The domain of the fuzzy class operator $\sim \{.|.\}$ is the set of variables V and the set of fuzzy propositional functions $\sim PF$, and the range is the fuzzy power-set (the set of fuzzy subsets) of this universe, $\sim P(U)$:

$$\sim \{.|.\} : Vx \sim PF \rightarrow \sim P(U) \tag{1.12}$$

Accordingly, fuzzy classification is the process of grouping individuals having the same characteristics into a fuzzy set. A fuzzy classification corresponds to a

membership function μ that indicates whether an individual is a member of a class, given its fuzzy classification predicate $\sim \prod$.

$$\mu :\sim PFxU \rightarrow \sim T \tag{1.13}$$

Here, $\sim T$ is the set of fuzzy truth values (the interval between zero and one). The fuzzy classification predicate $\sim \prod$ corresponds to a fuzzy restriction "i is R" of U. where R is a fuzzy set defined by a truth function. The degree of membership of an individual i in the fuzzy class $\sim C$ is defined by the truth value of the corresponding fuzzy predicate.

$$\mu \sim C(i) : = T(\sim \prod(i)) \tag{1.14}$$

Intuitively, class is a set that is defined by a certain property, and all objects having that property are elements of that class. The process of classification evaluates for a given set of objects whether they fulfil the classification property, and consequentially are a member of the corresponding class. However, this intuitive concept has some logical subtleties that need clarification. In classical logic the truth values are certain. Therefore a classification is crisp, since the truth values are either exactly true or exactly false.

In the next section we bring out the research motivations and contributions made in this thesis.

1.6 Research Motivations

The motivation of this thesis is aimed at building an effective *privacy preserving P2P* classifier. To preserve privacy, P2P classification should be payload oblivious. The classifier distinguishes the network traffic into two groups: P2P and Non-P2P traffic. Research questions and challenges that are addresses by this thesis are listed below:

Research Questions:

1. What is the best approach to classify real-time P2P traffic?

2. What are the optimal features that are needed to classify P2P traffic?

3 Which Machine Learning algorithm is suitable for P2P traffic classification?

The list of challenges to classify P2P traffic are listed as below:

A: P2P traffic has many characteristics that overlap with Internet traffic.

B: P2P Networks are heterogeneous in nature.

C: Protocol encapsulation (traffic is tunnelled) over HTTP.

D: Applications allow users to encrypt traffic on HTTPS.

E: Applications support multiple service channels like IRC, Proxy et.

F: Applications run behind NAT/Firewall.

1.7 Research Contributions and Organization of Thesis

The main contributions of this thesis lie in designing and implementing P2P traffic classification based on statistical and host based analysis.

- **Statistical Analysis:** This approach relies on traffic's statistical characteristics to identify network applications. An assumption is made that traffic at the network layer has statistical characteristics such as flow duration, packet length, number of bytes per packet, packet arrival periodicity for a number of Internet applications. The results thus, obtained stimulated new classification techniques based on traffic flow statistical properties. To deal with large datasets and multi-dimensional space of flow and packet features is one of the reasons for the introduction of ML techniques in this thesis.

  This thesis proposes multiple classifiers approach to improve the accuracy of P2P traffic identification than a single classifier in (near) real-time. In this approach different learning algorithms over the same dataset have been combined and the detailed work of statistical properties to identify network traffic using ML approach is discussed in Chapter 4 which also describes the advantages and limitations of this approach.

- **Host Based Analysis:** Host-based approach monitors the behavior of an individual host. In this approach the behavior of P2P applications were studied. The network data is classified based on the *connection patterns*, instead of looking at individual flows. The sequence of flows or packets to and from a specific endpoint are matched with a set of predefined heuristics. These heuristics do not require packet payloads. The detailed proposed

heuristics approach and its limitations are discussed in Chapter 5.

1.8 Thesis Organization

The reminder of this thesis is organized as follows. Chapter 2 introduces the P2P traffic detection techniques and methodologies. This chapter also provides the taxonomy of P2P traffic detection, and the advantage and disadvantages of each type of detection technique are also discussed. Data collection and preliminary analysis of P2P traffic identification using signature based approach with the help of Snort is discussed in Chapter 3. Chapter 4 presents the proposed P2P traffic identification framework with statistical fingerprints using ML techniques, with details regarding its design, architecture, and implementation. This chapter also discusses a novel combined classifier that is proposed to identify P2P traffic flows. The detection of P2P traffic classification using *connection patterns* without packet payloads with a set of predefined heuristics on real-time datasets is discussed in Chapter 5. A novel approach to detect P2P traffic based on behavioral metrics using Fuzzy based approach is discussed in Chapter 6. Chapter 7 presents the P2P-aware firewall to block any suspicious behavior of IP address based on the output of classifier modules. In this chapter the Firewall is configured as *in-line* to generate automate rules and dynamically firewall rules and Chapter 8 concludes the thesis.

Chapter 2

Literature survey

2.1 P2P Traffic Classification

This section describes the present classification approaches and methods. Network traffic classification is, *a method of classifying traffic data sets based on features passively observed in the Internet traffic according to classification goals.* In recent times, several approaches have been proposed for Peer-to-Peer (P2P) traffic classification. Different techniques used for P2P traffic classification are port-based, protocol-based, statistical flow or flow features and per host or social behavior of hosts as shown in Fig. 2.1.

The goal of this thesis is to classify traffic into *coarse classification*, i.e., whether traffic is transaction-oriented, bulk-transfer or P2P file sharing. Port based is one of the traditional approach to classify P2P traffic using transport-layer (TCP or UDP) port numbers, but as P2P applications began port randomization, this approach became less accurate to classify P2P traffic.

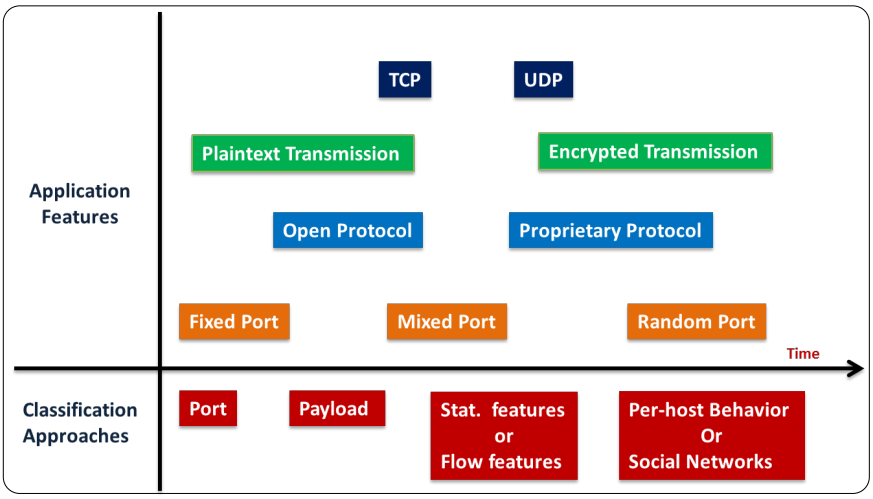


Figure 2.1: Inclinations of Applications and features [1]

2.2 Classification Approaches

This section describes a classical approach used to profile P2P applications. There are four different techniques in network traffic classification. A detailed summary of these approaches of pros and cons are discussed in the subsequent sections. Fig. 2.2 illustrates network traffic classification approaches.

- Port-based Approach
- Payload-based Approach
- Connection Patterns at Transport Layer
- Flow feature based Approach

2.2.1 Port-based Approach

One of the early techniques to identify traffic at application layer was based on port numbers. Internet Assigned Numbers Authority (IANA) has published list of registered port numbers which is available in the web [36]. This method is

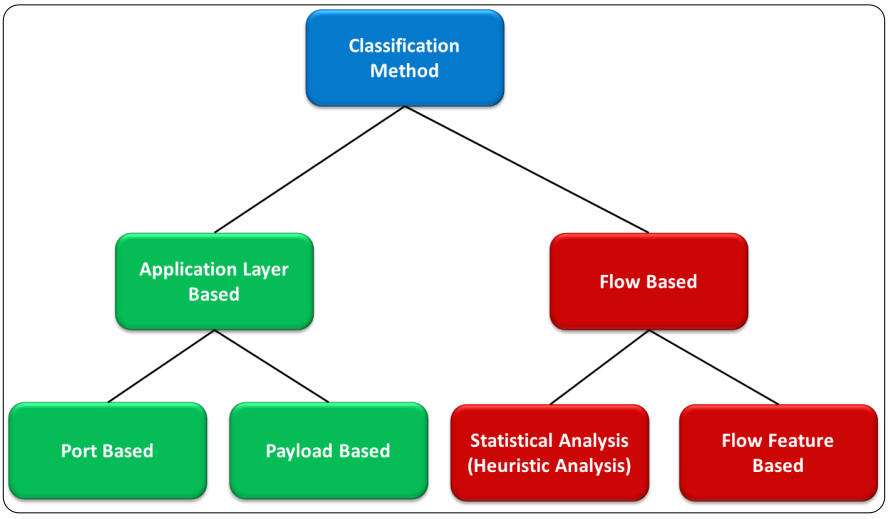


Figure 2.2: Classification Approaches

the basic and straight forward and accurate to detect P2P traffic within network traffic. Most P2P applications have default port numbers on which they function. Some commonly used P2P applications and their well-known port numbers are given in Table 2.1.

With port based analysis it is easy to identify weather traffic is P2P or not. But it has several limitations. Most of the P2P applications change their default port numbers by allowing users manually to configure whatever they like. Madhukar *et al.* [37] observed The P2P traffic was 30%-70% in University of Calgary. Authors examined only TCP port numbers to classify P2P traffic. Their analysis revealed that the port based approach is ineffective for classifying P2P traffic. Additionally, many newer P2P applications begin to masquerade their port numbers over well-known ports like 80 and 443, to bypass the Firewalls or Intrusion Detection Prevention System (IDPS). Recently, some applications like μTorrent, PPStream, PPLive etc., have changed from using TCP to UDP, which is a challenge to traditional approaches. Hence, port-based analysis is less effective in traffic identification.

Table 2.1: Well-known port numbers used by several applications

Protocol	**TCP Port**	**UDP Port**
Direct Connect	411, 412, 1025-32000	1025-32000
eDonkey	2323, 3306, 4242, 4500, 4501, 4661-4674, 4677, 4678, 4711, 4712, 7778	4665, 4672
Gnutella	6346, 6347	6346, 6347
Kazaa	1214	1214
Limewire	6346	6347
BearShare	6346	6346
eMule	4662	4672
BitTorrent	6881-6889	6881-6889
winMx	6699	6257
AIM-messages	5190	5190
AIM-video	1024-5000	1024-5000
ARES Galaxy	32285	32285
Blubster	41170-41350	41170-41350
FastTrack	1214, 1215, 1331, 1337, 1683, 4329	
GoBoogy	5335	5335
HotLine	5500-5503	
iMesh	80, 443, 1863, 4329	
IRC	6665-6669	
MP2P	10240-20480, 22321, 41170	41170
MSN-voice	6901	6901
Napster	5555, 6666, 6688, 6699-6701, 6257	
Qnext	5235-5237	5235-5237
SoulSeek	2234, 5534	2234, 5534
WinMX	6699	6257
XMPP/Jabber	5222, 5269	5222, 5269
Yahoo-voice	5000-5001	5000-5010

2.2.2 Protocol/Packet-level/Payload Based Approach

To overcome the limitations of the previous approach, an accurate classification requires *payload* examination. It monitors traffic passing through the network and identifies byte strings associated with an application or perform more complicated syntactical matching. The most payload based approaches compare each packet content to a set of known signatures (pattern matching) from the database. It is an accurate approach for P2P traffic classification, but examining payload is an arguable methodology due to privacy concerns to an individual. Examining payload technology is referred as Deep Packet Inspection (DPI), which requires full packet payloads. There are several commercial and open-source tools available in the research community to identify the P2P traffic according to signatures and these tools map few Bytes from every packet payload written in some regular expression format.

Many researchers have developed P2P traffic identification techniques using signature based approach. Gummadi *et al.* [38] proposed extraction of application signatures to characterize the workload of KaAaA. They do not provide any accuracy and scalability of their signature generation. Sen *et al.* [39] have generated for five most popular P2P protocols by analyzing application-level signatures of Gnutella, eDonkey, DirectConnet, BiTorrent and KaZaA protocols with high accuracy. Their approach improved the accuracy of classification, but the signature discovery is poor and not suitable for real-time analysis and the signatures are generated only for TCP but not for UDP. The signatures are manually generated using the off-line tools. Madhukar *et al.* [37] extracted signatures from most popular P2P applications: Gnutella, BiTorrent and KaZaA. Their approach uses few packets and extracted simple signature to accurately identify P2P applications.

Recently Park *et al.* [40] proposed automated generation of applications-level signatures, which do not need any prior information about the protocol.

Few of these include, L7-filter [41], Cisco's NBAR [42] , Juniper's netscreen-IDP [43], QOSMOS' [44], nTop [45] , Open-DPI [46] and Snort [47]. These tools monitor every packet payload and raise alerts when the predefined signature is matched. With this type of classification technique we can overcome the demerits of port based classification techniques. But, when P2P applications are evolving continuously and their signatures can also change, we need to keep monitoring and update new signatures observed. However, this type of classification has limitations like, if the P2P applications use tunnels or encrypted traffic, it is difficult to detect it. Another limitation of this technique is at a high bandwidth it needs more resources in terms of hardware requirements like CPU speed, memory size etc.

2.2.3 Connection Patterns at Transport-Layer

Karagiannis *et al.* [48] proposed a novel-approach for classifying P2P traffic, based on the transport-layer connection patterns. It doesn't require payload analysis. Rather, the analysis based on *IP-Port* pairs. Irrespective of application level data, the connection level patterns remain the same. It relies on two major heuristics to identify P2P traffic.

- Look for source-destination IP pairs that concurrently use both TCP and UDP. If such a pair exists then they are marked as a P2P flow.
- Examine all source IP, source Port and destination IP, destination Port pairs. Look for all pairs for which the number of distinct IPs communicated

is equal to the number of distinct ports communicated.

Limitation to their approach does not include UDP packets, the effectiveness of TCP/UDP heuristics is minimal and also the traces contain TCP - SYN, FIN and RST packets, so data packet size cannot be utilized. Madhukar *et al.* [37] proposed a sliding-window to observe the flows. In their approach, they eliminated all flows use standard port numbers of NonP2P applications to create IP, port pairs to identify P2P applications. Unfortunately, this approach fails to detect port masquerading. Another limitation is that the IP-port heuristic is ineffective when an IP host communicates with another IP host on a single port, which is actually identified as a P2P behavior over UDP.

Sen *et al.* [49] proposed P2P traffic identification using flow-level records and analyzed these records with Zipf's distribution. They studied three popular P2P systems – FastTrack, Gnutella and DirectConnect. They separated signalling traffic from the data traffic to accurately identify P2P applications. The key finding in their approach is that the traffic volume generated by individual host is extremely variable, which is less than 10% of the IP addresses contribute around 99% of the total traffic volume. The traffic distribution is skewed for both upstream and downstream traffic at the prefix, and Autonomous Systems (AS) suggested coarse-grained traffic management.

Karagiannis *et al.* [50] proposed a traffic classification approach based on host behavior at transport layer. They looked at a multilevel approach to classify traffic according to the applications that generate them. Their traffic classification approach uses host behavior first and then social, functional, and application level behavior. At social level, their approach identifies hosts with similar behavior which is evident from the interactions a host makes with other hosts. At a func-

tional level, it identifies what function a host plays in the network, i.e. either a provider or a consumer of the service, or both. At an application layer, they look for transport layer interactions to identify traffic.

Hurley *et al.* [51] proposed a set of heuristics for P2P traffic and web traffic identification by using host behavior. They could identify 90% of the flows accurately that go out and come into a host. The heuristics were developed by using information like source host, destination host, connections between the hosts, and the flow activity like how many packets per flow etc. Yan *et al.* [52] proposed P2P traffic identification scheme based on both host behavior and flow behavior. First they identified whether a host is running a P2P application by matching it's behavior with a set of predefined rules like the number of ports open, number of IPs connecting, number of failed connections etc. Next they refined this identification by comparing the statistical features of each flow in the host with several flow features' profiles like flow duration, flow volume etc. The identification accuracy was above 90%.

John *et al.* [53] proposed heuristics to classify Internet traffic based on network applications that include P2P applications along with other types of applications. The heuristics used were based on connection patterns between two hosts. Heuristics like usage of both TCP and UDP concurrently, a particular port usage, ratio of IP/port pairs etc. were used to achieve identification accuracy of upto 99.8%. Perenyi *et al.* [54] derived a set of heuristics from the robust properties of P2P traffic collected or revealed from a traffic aggregation. They also presented a traffic analysis based on behavior of active users, ratio between P2P users and normal users etc. and observed that the daily profile of P2P traffic intensity is less variable.

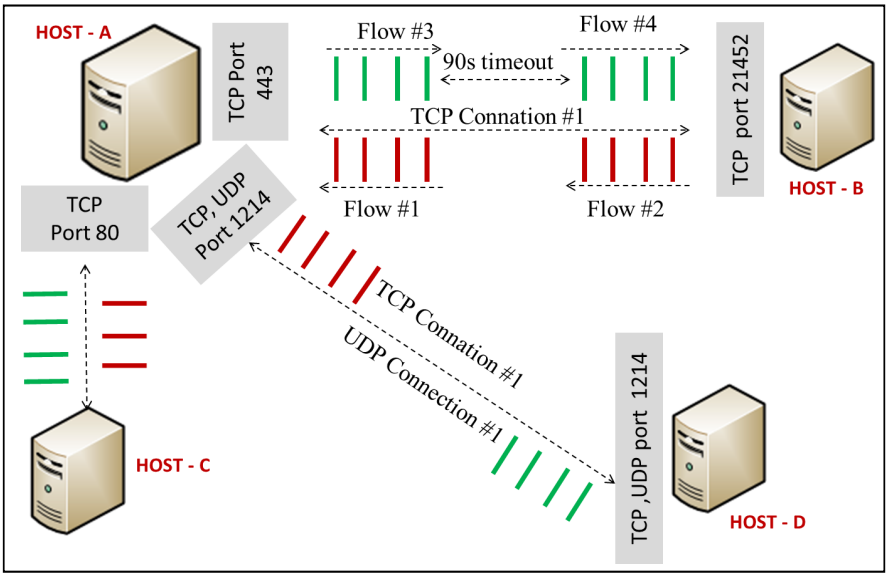


Figure 2.3: Flow example, Host A opens HTTPS (443), HTTP (80) and default port for Kazaa (1214). Host A and Host B packets are grouped into either single TCP connection or two bi-flows or four flows. On the other hand, Host A again opens new port on both TCP and UDP, packets can be grouped into single TCP and UDP connection.

2.2.4 Flow based traffic Classification: Machine Learning

Researchers have also studied approaches that are independent of packet content, such as statistical techniques based on network flows. Another approach is to identify P2P traffic by examining the flows that can vary from one direction of an individual application to bidirectional between hosts. A typical flow has 5-tuples i.e., source IP, destination IP, source port, destination port, and protocol. Fig. 2.3 illustrates different types of flows between hosts. In section 1.4 we discussed the background study about machine learning (ML). This section focus on the past research work to identify P2P traffic using ML.

Nguyen et al. [55] presented application of several ML algorithms for IP traffic identification with several single classifiers, which justify the usage of ML algorithms for any type of traffic classification work. A wide variety of techniques

have been practised to classify traffic either in backbone or in the age-networks.

J. Li *et al.* [56] proposed C4.5 and REPTree algorithms to classify the traffic as P2P or Non-P2P and then compared with K-Means algorithm which is an unsupervised technique. The authors have derived features in the following way: first, they aggregate the packets into flows with five-tuples, and then the features like source port, destination port, flow duration, total packets, total bytes, packet inter-arrival time, packet payload, packet length are extracted from first to eight packets. The traffic traces considered in their work were from applications like DNS, HTTP, POP3, FTP, Streaming, BitTorrent, eMule, Game, PPlive, MSN, KaZaa, Gnutella, Skype etc. They have built up a dataset by randomly sampling traffic from these traces to eliminate bias. Their results showed higher accuracy with decision tree (C4.5) than REPTree and K-Mean clustering algorithms.

Y. Zhang *et al.* [57] proposed a supervised Machine learning algorithm to classify different P2P applications. The authors have computed features like transport layer protocol, payload length, and difference of control packets proportion between two directions. Authors have used categories of applications: File-sharing applications (Bit torrent, eMule, Thunder), P2P streaming media (PPLive, PPStream) and P2P voice application (Skype). From the above features they built a decision tree with C4.5 algorithm and measured the classifier accuracy which came out to be 96.7%. H. Xu *et al.* [58] proposed P2P traffic identification using nave Bayes and decision tables. Features like PUSH packets, ACK packets, Data packets length etc. are extracted from the flow using NETMATE [59] tool. Using WEKA [33] tools feature selection algorithm, they pruned the selection to eight features out of forty four features extracted by NETMATE. Their traffic identification accuracy rate was 97%.

An alternative flow based approach is to accurately identify P2P traffic using ML techniques, which is promises to classify P2P network traffic. Each flow is characterized by the same set of features but with different feature values. ZHAo *et al.* [60] proposed real-time feature selection in traffic classification which is the most important step in ML. Feature selection not only improves the accuracy, but also improves the computational performance. In their approach, they calculated first quartile of bytes in packet, control bytes, total bytes of IP packet and inter-arrival time features from every flow. Their experimented result shown that, C4.5 achieves greater than 96% accuracy and RandomForest achieves greater than 99%. The tree based algorithms always either over-fit or under-fit the data if the dataset is imbalanced i.e., the variance error is more which is the limitation of their approach.

H. Chen *et al.* [61] proposed traffic identification using supervised ML algorithms: support vector machine (SVM). The authors used a proprietary tool called as OmniPeek [62] to collect the traffic and extract the features. The features extracted include traffic duration, average traffic speed, average packet length, ratio of average TCP packet length to that of UDP, ratio of TCP traffic to UDP traffic, etc. Their experiments used several SVM parameters to measure the accuracy of different attributes sets. The RBF kernel with SVM method achieved an accuracy rate of 99.54%. J. Erman *et al.* [63] identified traffic using unsupervised techniques, such as K-Means and DBSCAN. With K-Means, authors obtained 85% accuracy and with DBSCAN it is 75%. To accurately identify traffic, authors extracted flow statistics form their flows with features like total number of packets, mean packet size, mean payload size, bytes transmitted for bi-flow etc.

C. Gu *et al.* [64] proposed a novel P2P traffic classification scheme using back

propagation neural network learning algorithm. They have proposed Genetic Algorithm (GA) for best feature selection to reduce the time complexity during the training phase. To identify the P2P flows, authors have derived features like total_fpackets, total_bpackets, etc. They achieved an accuracy of 86.75%. Ang *et al.* [65] proposed an adaptive classification ensemble framework for P2P networks, using Support Vector Machines (SVM) to adjust the Voting scheme dynamically by combining a subset of classifiers according to the test data. Dong *et al.* [66] proposed multi-classifier model for traffic classification using C4.5, Naïve Bayes and SVM as base classifiers with Voting as the fusion scheme. They showed that multi-classifier models with fusion improved accuracy over single classifier models.

Wang *et al.* [67] proposed voting scheme as ensemble learning to identify traffic from flows to dynamically update the on-line classification module when changes occurred in the off-line module for every few hours and days. Wang *et al.* [68] proposed traffic identification from flows using randomly selected five packets set. During the feature extraction, authors did not use inter-arrival time while achieving better accuracy with few packet sets. Dan *et al.* [69] proposed the voting scheme as ensemble learning, with base classifiers as DTNB, ONER, and BPNs and their experimental results showed that traffic identification with 97.27% of average precision rate.

Traditional ML techniques, i.e using a single ML classifier (ex: C4.5, SVM, etc.), build a model on training dataset and validate over test dataset. These classifiers may over-fit or under-fit the training data that results in more false positives and false negatives when compared to ensemble learning techniques. More details of single classification techniques can be found in [56, 57, 58, 61, 63, 64, 55, 65]. Authors in [66, 67, 69] proposed multi-classifier technique, namely *Voting*

Table 2.2: Comparison of network traffic classification approaches

Approach	**Characteristics**	**Pros**	**Cons**
Port-based	Relies on *port numbers*	Easy to implement and less computational complexity	Inaccurate due to random port numbers
Payload-based	Relies on packet *payload*	Highly accurate	may not work for encrypted traffic, requires high computing resources, and accurate when signatures are known
Connection Patterns	Uses packet headers	Applicable for encrypted traffic	Less accurate when compared to payload based
Flow based	accumulate packets into flows and extracts flow metrics	Applicable for encrypted and identify unknown applications from target classes	Needs machine Learning algorithms which increases the computational cost in terms of model build time.

scheme, which combined multiple traditional models to obtain better classification accuracy to discriminate between P2P and NonP2P traffic. The drawback of $Voting$ scheme is that by increasing the number of base classifiers, the performance degrades. Table 2.2 summaries the comparison of these approaches.

Next Chapter 3 of this thesis covers the data collection and preliminary analysis to understand the underlying communication of peer-to-peer applications. In the first phase, data collection, ran several P2P applications individually in a controlled environment on our testbed. Second phase, we analyzed the captured

P2P traffic based on the port and signature based approaches. We created our own set of Snort rules, to identify whether the host/IP address is P2P or Nonp2p.

Chapter 3

Data Collection and Preliminary Analysis

3.1 Data Collection and Preliminary Analysis

BITS-Pilani Hyderabad Campus network furnishes services to three thousand to four thousand on-campus students along with faculty and guest users every day, providing wired as well as wireless connectivity in the various parts of the BITS-Pilani Hyderabad campus. Internet connectivity is facilitated in the BITS-Pilani Hyderabad Campus via leased lines drawn from two Internet Service Providers (ISPs). The BITS-Pilani Hyderabad Campus network consists of a three-tier architecture, consisting of two core switches one for fall-back to another, multiple distribution switches and additional access switches with a 10 Gb backbone fibre-optic. This architecture is shown in Fig. 3.1 and Fig. 3.2. Wireless connectivity is also available in the academic blocks and other designated areas within the campus by deploying Wi-Fi access points and a wireless controller. Access to

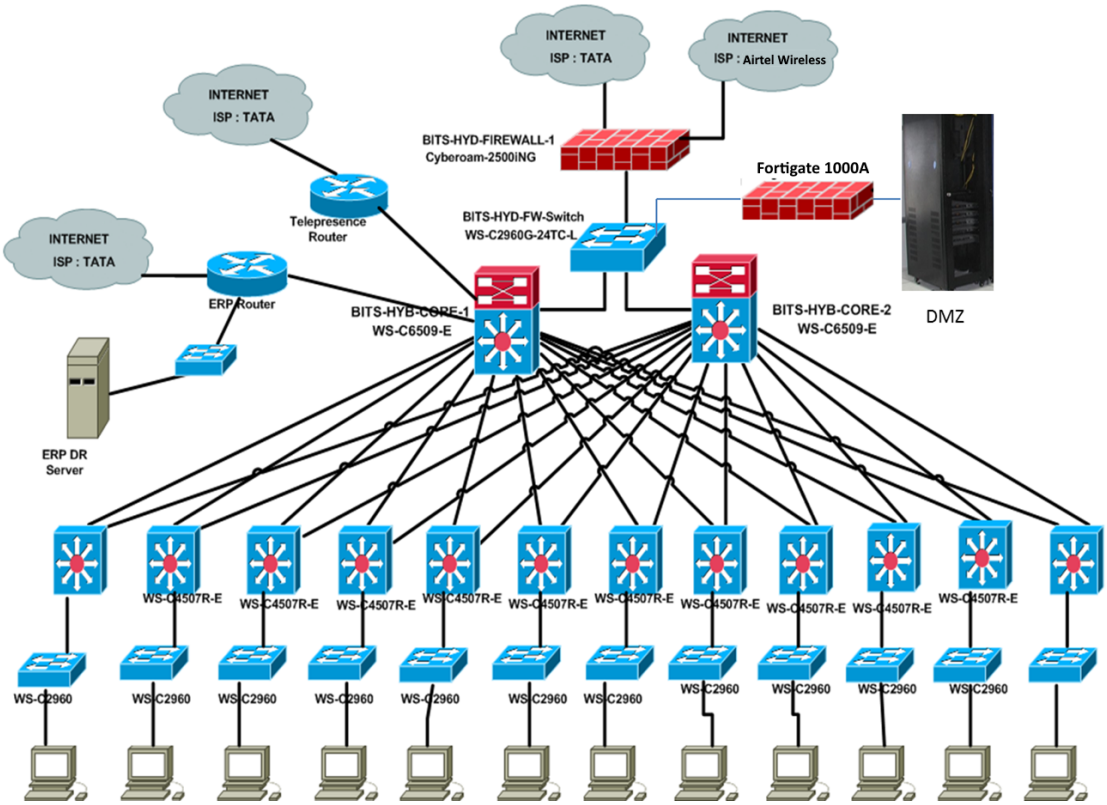


Figure 3.1: Network architecture of BITS-Pilani Hyderabad Campus

different Internet-based services hosted by the BITS-Pilani Hyderabad Campus is facilitated in a 'DMZ' (demilitarized zone). The DMZ hosts, Web (HTTP(S)) service, FTP service, and the Institutional email service (SMTP) of BITS-Pilani Hyderabad Campus.

The network is protected by a commercial Unified Threat Management (UTM) system which controls the ingress and egress Internet traffic. The UTM performs the task of traditional Intrusion Detection/Prevention Systems (IDS/IPS). Along with this, it also plays the role of performing anti-virus, anti-malware and spam-filtering services. It is instrumental in performing URL filtering and detecting potentially suspicious websites hosting malware, spyware, adware etc.

Large number of users in a large scale university contribute data to such networks

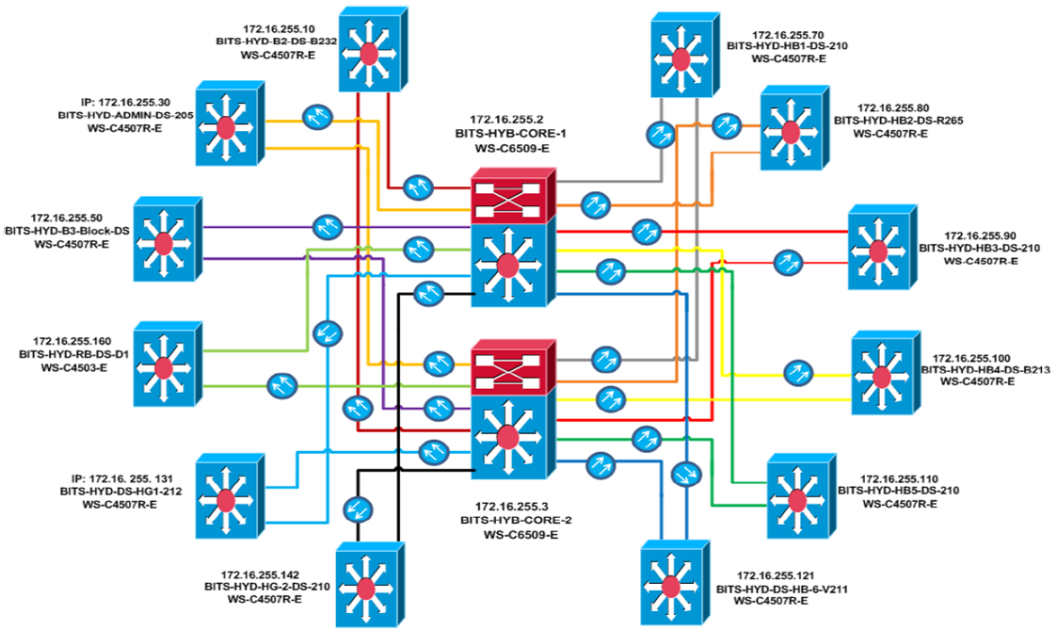
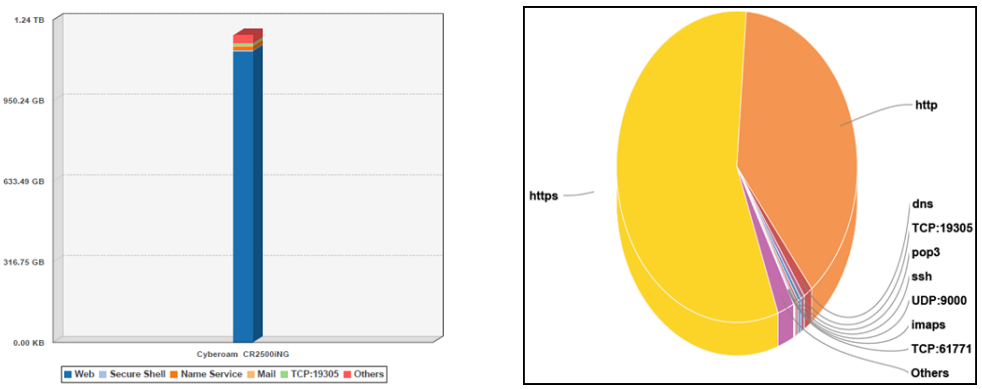


Figure 3.2: Network architecture of BITS-Pilani Hyderabad Campus

which conglomerate to huge amounts of data every day. Web-based traffic related to web browsing, social networking, search engines and education related material forms the major backbone of Internet traffic. P2P based applications,although popular amongst students, play a pivotal role in consuming large amounts of Internet bandwidth. Although these applications are restricted by the UTM, it is observed that students are still able to override the UTM and run many such P2P applications on the University network. This might be attributed to the fact that many such applications use encryption (thus,making signature detection difficult) and randomize their port numbers (thus, localizing port numbers by filtering is difficult).

The average network traffic generated is one Terabyte per day in our campus. A snapshot of the traffic logs of one day (20 August 2014) as captured by the UTM, supplemented with the distribution of different types of applications running in the network, is given in Fig. 3.3. From the graphs, it is evident that the majority



(a) Internet Usage Bar Graph (b) Distribution of protocols

Figure 3.3: Logs for Internet traffic generated on one day at the university of the authors

of the traffic generated in the campus is primarily web-based (HTTP/HTTPS).

The testbed shown in Fig. 3.4 is made up of both wired and wireless nodes at BITS Hyderabad Information Security Lab, Distributed Systems Lab and two of our hostels. Network traffic is mirrored from the border router and captured at the storage server. Due to the real-time traffic, we need to guard the sensitive information of users in the packets. Due to magnanimous amount of packets per day, processing which would otherwise take enormous amount of time can be limited by dropping packets without any privacy threat associations. To ensure a secure communication through this channel, filtering of HTTPS packets is ensured.

The capture was segregated in accordance to the external IP addresses with which the communication was taking place from an internal hosts. It was noticed that most of the traffic content consisted of online video streaming and software updates from online repositories following which this traffic was also dropped. The P2P packets' payload and extracted signatures of these applications were thoroughly analysed Using AnonTool [70], we anonymized all the packets which were mapped with extracted signatures.

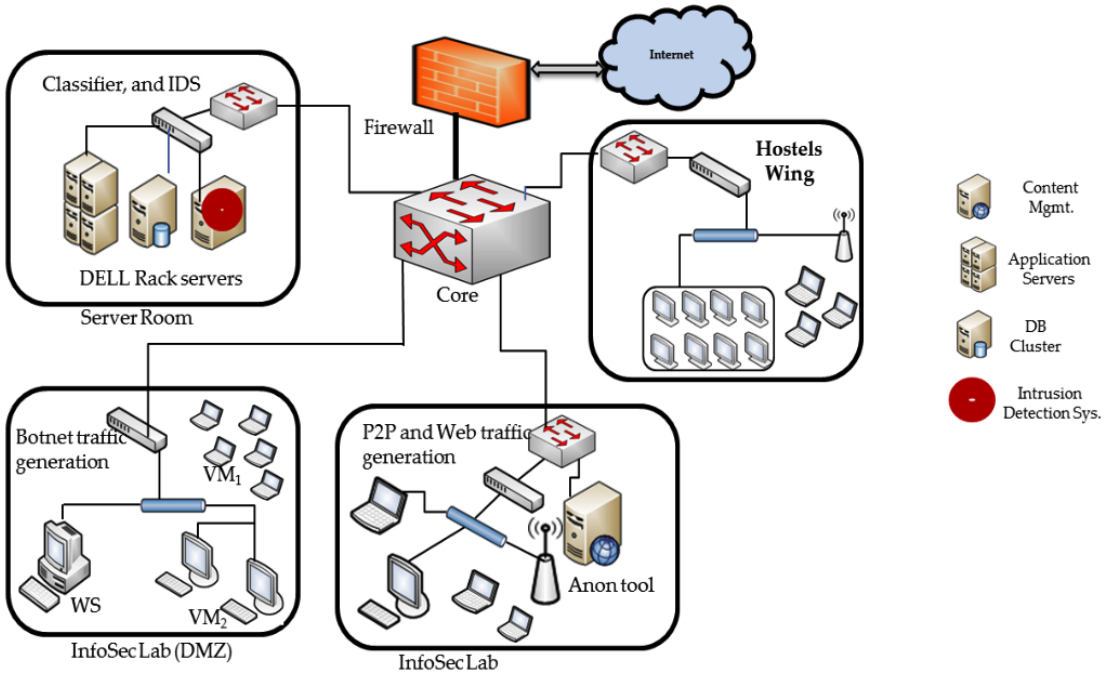


Figure 3.4: Testbed for impact analysis of P2P on IPS/IDS

In the inception, payload-based identification of P2P traffic using an open-source IDS Snort tool was carried out. To detect P2P traffic, extracted signatures from multiple P2P applications from network traces obtained from our campus testbed were prepared. A sample of these signatures is given in Table 3.2. A snapshot of our experiments and results with Snort is also shown Fig. 3.5.

Another significant research fact to be pursued in this work is to assess the impact of P2P traffic on the performance (accuracy and speed) of traditional network

Table 3.1: P2P Dataset

Application	**Amount of Data**	**Source of data**
Direct Connect	20 GB	Generated on testbed
eMule	39 GB	University of Georgia
FrostWire	11 GB	University of Georgia
μTorrent	53 GB	University of Georgia
Vuze	19 GB	University of Georgia

Table 3.2: Application Signatures

Application	TCP	UDP
DC++	$Send, $Connect, $Get, $Search, $MyINFO, $MyNick, $Hello, $Quit, $ADCGET, $ADCSND, $Supports, $dcplusplus	$SR, $Pin
BitTorrent	GET, /announce, info_hash=, peer_id=, event=, 0x13 BitTorrent protocol	-
GNUTella	GNUTELLA, GNUTELLA CONNECT, GNUTELLA OK	GND
eDonkey	Server 3A eMule	-
Fasttrack	Get /.hash	0x270000002980
eDomkey2000	0xe319010000, 0xc53f010000	0xe319010000, 0xc53f010000
Ares	GET hash:, Get sha1:	-
KazaA	kazaa, kazaaClient, KazaA, X-Kazaa, 48 54 54 50 2f 2f 31 2e 3120 32 30 30 20 4f 4b	-

security devices like firewalls and IDSs. Due to relatively large volume of P2P traffic, its impact on these devices is quite significant. However, this impact is not well deciphered and consequently, solutions to neutralize this impact does not exist. The proposed testbed (Fig. 3.4) is used to quantify the impact of P2P traffic on network security devices and develop solutions to mitigate this impact. We use this testbed as a reference throughout the thesis.

In the Information security lab, attack traffic was simulated by creating binaries of bots as well as by replaying existing bot datasets. Attacks generated included port-scans, ICMP ping of death, TCP SYN floods and UDP floods [71]. For the purpose of creating malicious traffic, binaries of bots were obtained from innumerable sources. Virtual Machines with Windows OS were deployed on Linux machines and these VMs were infected with the binaries of bots. This set-up was created in a VLAN segregated from the BITS network. It has been observed in

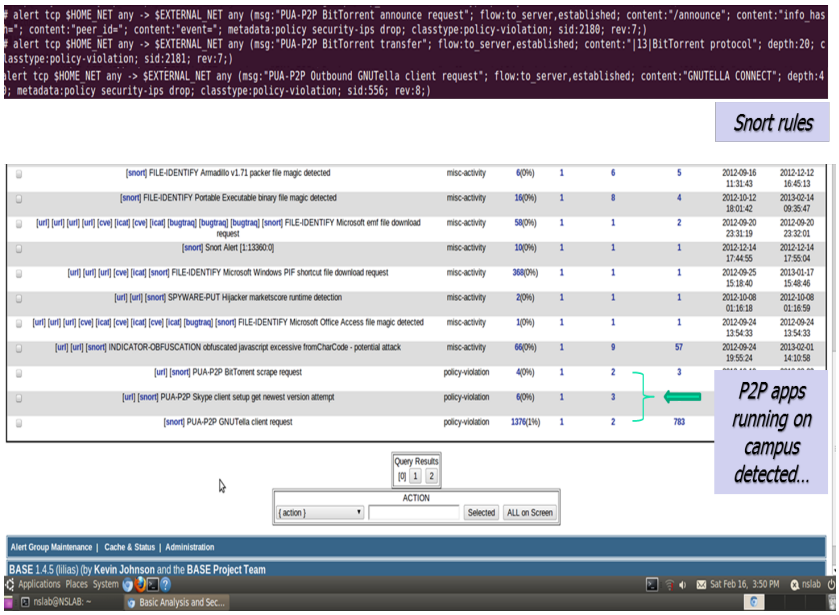


Figure 3.5: Payload-based detection of P2P traffic using Snort IDS

[72] that in order to successfully run security-related experiments in a virtualized environment, the users must be provided privileged access rights on the virtual machines to execute security or network tools. This facility was furnished to all users of the system and secure remote connectivity was also facilitated. Since bots run in a virtualized environment, and that too on a VLAN separated from the rest of the network, the testbed exercises control over the effects of the generated attack and background traffic [73]. However, the systems infected with bots were allowed communication with the Internet. In Fig. 3.6, a snapshot of this lab set-up is provided showing VMs running on a controlled environment running P2P applications and NonP2P applications. We deliberately ran machines P2P to see the efficacy of our detection algorithms which are presented in later chapters.

Distributed system laboratory at BITS, Hyderabad, runs application servers as well as some of the peers for P2P clients. This laboratory also includes Database server(s) that stores the data and monitors logs of various events (particularly

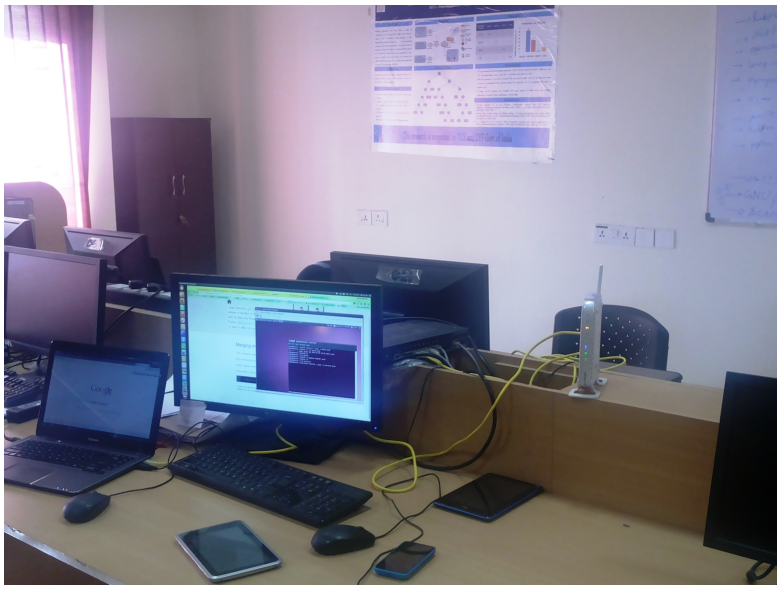


Figure 3.6: Virtualized Environment running P2P & NonP2P traffic at BITS Pilani, Hyderabad campus

Syslog service) generated from our experiments. Multiple P2P applications were executed which include file sharing applications as well as media streaming. In addition, simulation of non-P2P traffic using freely available simulators like mail server and web-server loads are also performed.

Anonymized and prioritized traffic from BITS hostels and labs is relayed to Information Security Lab for P2P and IDS analysis. Wired as well as wireless data collectors were deployed for this purpose. Info-Sec lab also has a Linux based firewall, which was integrated with IDS to thwart attacks detected by it in real time. Moreover, based on IDS triggers, solutions were formulated to blacklist compromised systems from getting access to wired/wireless network.

Inclusion of 'live Internet traffic' is taken for analysis of IDS and P2P classifiers. This traffic is mirrored from the border router and made available in the Info-Sec lab. Once an integrated solution for IPS was made available, we use our firewall (in Info-Sec lab) to carry all ISP bound traffic while enforcing the IDS/IPS

functionality in real time.

Preliminary experiments were carried out using existing PCs in the Info-Sec lab for detection of P2P traffic based on signature-based detection using Snort. Snort performs real-time protocol analysis, content searching and content matching on Internet Protocol (IP). As part of the experiment(s), few P2P applications like, multiple versions of Direct Connection (DC++) and BitTorrent traffic were taken into consideration. The reason for considering different versions of DC++ is because of the varying nature of signatures from version to version. For these two applications well defined rules set to detect its traffic were created. Snort can be configured in three modes: sniffer, packet logger and Network Intrusion Detections System (NIDS). When Snort is in NIDS, it monitors traffic, analyses it and raises alerts as per the user defined rule base. To monitor the alerts Basic Analysis and Security Engine (BASE) is used which happens to be a third party tool that interfaces with Snort. Fig. 3.5 captures our experimental results showing the alerts that were raised.

The following signatures were used to identify several applications (Torrents, KaZaA and DirectConnect) using Snort IDS:

Torrent Signatures:

```
alert $HOME_NET any  <> $EXTERNAL_NET any (msg: "P2P Tracker Site";
content: "GET"; sid:10000001: rev:1;)
```

```
alert $HOME_NET any  <> $EXTERNAL_NET any (msg: "P2P Torrent
metafile Download"; content: "d8\:announce"; flow:established,
to_server; classtype: policy-violation; sid:10000002: rev:2;)
```

```
alert udp $HOME_NET any <> $EXTERNAL_NET any (msg: "P2P torrent
DHT ping";content:"d1\:ad2\:id20\:"; classtype:policy-violation;
sid:1000003; rev:3;)
```

KaZaA Client Signature:

```
alert tcp $EXTERNAL NET any <> $HOME NET any (flow:from client;
content:"|48 54 54 50 2f 2f 31 2e 3120 32 30 30 20 4f 4b|";
offset:0; depth:15; content:"KazaaClient";session:printable;
msg:"Request of a shared file with KaZaA";sid:1000004;)

alert tcp $EXTERNAL NET any <> $HOME NET any (flow:from client;
content:"X-Kazaa";
offset:0; depth:07; content:"X-Kazaa";session:printable;
msg:"Request from P2P-Agent";sid:1000005;)

alert tcp $EXTERNAL NET any <> $HOME NET any (flow:from client;
content:"KaZaA"; offset:0; depth:05; content:"KaZaA";
session:printable; msg:"Request from X-Kazaa-Network";sid:1000006;)
```

Direct Connect Signatures:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Send";
offset:0; depth:5; classtype:dcconectionsend; sid:100201;
priority:9;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Connect";
offset:0; depth:8; classtype:dcconectionconnect; sid:100202;
priority:10;)
```

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Get";
offset:0; depth:4; classtype:dcconectionget; sid:100203;
priority:11;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Search";
offset:0; depth:7; classtype:dcconectionsearch; sid:100204;
priority:12;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$MyINFO";
offset:0; depth:7; classtype:dcconectionmyinfo; sid:100205;
priority:13;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$MyNick";
offset:0; depth:7; classtype:dcconectionmynick; sid:100206;
priority:14;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Direction";
offset:0; depth:10; classtype:conectiondirection; sid:100207;
priority:15;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Hello";
offset:0; depth:7; classtype:dcconectionhello; sid:100208;
priority:16;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Quit";
```

```
offset:0; depth:5; classtype:dcconectionquit; sid:100209;
priority:17;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$Lock";
offset:0; depth:5; classtype:dcconectionlock; sid:100210;
priority:18;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (content:"$key";
offset:0; depth:4; classtype:dcconectionkey; sid:100211;
priority:19;)

alert udp $HOME_NET any <> $EXTERNAL_NET any (content:"$SR";
offset:0; depth:3; classtype:dcconectionsr; sid:100212;
priority:20;)

alert udp $HOME_NET any <> $EXTERNAL_NET any (content:"$Pin";
offset:0; depth:4; classtype:conectionpin; sid:100213;
priority:21;)
```

Gnutella Signatures:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from client;
content:"GNUTELLA"; offset:0; depth:08;
classtype: Gnutella Connection; sid: 3000001; priority:30;)


alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from client;
```

```
content:"GNUTELLA CONNECT"; offset:0; depth:08;
classtype: Gnutella Connection Request from Client;
sid: 3000002; priority:31;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from server;
content:"GNUTELLA OK"; offset:0; depth:11; classtype:
Gnutella Connection Found; sid: 3000003; priority:32;)

alert udp $HOME_NET any <> $EXTERNAL_NET any (flow:from server;
content:"GND"; offset:0; depth:03; classtype: Gnutella Connection
Request Accepted; sid: 3000001; priority:33;)
```

Fasttrack Signatures:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from client;
content:"Get /.hash"; offset:0; depth:10; classtype: Fattrack
connection Request; sid: 5000001; priority:41;)

alert udp $HOME_NET any <> $EXTERNAL_NET any (flow:from server;
content:"|27 00 00 00 29 80|"; offset:0; depth:11; classtype:
Fattrack over UDP; sid: 5000002; priority:42;)
```

Ares Signatures:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from client;
content:"GET hash:"; offset:0; depth:08; classtype: Ares requests
hash of file list from peers; sid: 6000001; priority:51;)

alert tcp $HOME_NET any <> $EXTERNAL_NET any (flow:from client;
```

```
content:"GET sha1:"; offset:0; depth:08; classtype: Ares requests
sha1 of peers list; sid: 6000002; priority:52;)
```

3.2 Summary

In this chapter we have introduced the taxonomy of P2P traffic identification techniques using signatures applied in P2P detection in detail. The usage of P2P applications is increasing day by day and which allows more P2P applications and more users sharing their content.

The P2P packets' payload and extracted signatures of these applications were thoroughly analysed This approach accurately identifies P2P traffic. To detect the signatures, we search few strings of the packets' payload and maps against the signature database. In this thesis, we extracted the signatures from popular P2P applications. However, signature based detection also has its drawbacks: (1) if the P2P applications use tunnels or encrypted traffic, it is difficult to detect it. (2) Another limitation of this technique is at a high bandwidth it needs more resources in terms of hardware requirements like CPU speed, memory size etc. In order to alleviate the problem of high false positive rate in P2P traffic detection we propose the an multi-classifier approach which combines several Machine Learning algorithms that can have high performance and efficiency. In the next chapter, Chapter 4, we present the proposed framework for P2P traffic identification.

Chapter 4

Design of a privacy-preserving P2P traffic classifier

4.1 Network Traffic Classification

With increase of network bandwidth, network behavior becomes more complex and produces a myriad of new network applications. Traffic identification and classification are becoming increasingly important for network administrators. Network engineers' tasks comprise of many responsibilities, e.g., meeting bandwidth requirements of customers or services, monitoring the bandwidth consumption of the users,if necessary, implementing security rules and performing accurate accounting for billing issues [74]. To accurately identify network traffic it is vital to provide best-effort delivery to the end users, i.e traffic shaping/policing, traffic prioritization, and quality of service(QoS). On the other hand file sharing applications, particularly, Peer-to-Peer applications, are designed to use available bandwidth which impact the quality of Service (QoS) to other applications run-

ning in the network.

These applications are indecipherable by following traditional policies. On fixed networks, around 27% of aggregated traffic (upstream and downstream) is carried by P2P (file sharing applications) networks, as reported by Sandvine [5]. The impact of P2P network traffic is increasing day by day. It is clogging the ISPs' bandwidth and becoming the nucleus of many attacks since worms, viruses etc. are launching attacks by exploring vulnerabilities in these applications.

It is an important problem to classify the P2P traffic from the regular Internet traffic and this is a uphill task to discriminate between P2P to Non-P2P traffic because of the evolution of P2P networks and applications. The P2P applications carry the traffic which is similar to web traffic at transport layer either TCP/UDP or both. Many research papers have been published, but till now, challenges to classify the P2P traffic remain unresolved.

P2P traffic identification is bifurcated into two techniques namely:

1. Signature based technique – It maps few bytes (referred as signatures) of the payload of every packet to verify whether the packet is P2P or Non-P2P at the expense of more computation power (in-terms of CPU and RAM) and thus leading to privacy concerns which form the bottleneck of this technique.

2. Anomaly based technique – It observes the behavior of the application/network and then extracts relevant characteristics/features from the network traces. It subsequently applies statistical methods on these features to classify the traffic by using Machine Learning (ML) technologies.

4.2 System Overview

To identify P2P traffic, we employ statistical characteristics of TCP and UDP protocols. We constructed 5-tuple (src_ip, src_port, dst_ip, dst_port, protocol) flows from these two protocols and then extracted statistical features from the network traces. We attempt different feature selection techniques to reduce the number of features required to train the model i.e reducing the model build time and achieve optimized training model.

Stacking and Voting ensemble learning techniques were used to improve the prediction accuracy (Detection rate) with base classifiers modelled using Machine learning (ML). Network traffic traces were collected in the testbed at our laboratory. A Java based module with the jNetPcap library [75] is used to extract the features from the captured network traces. The performance comparison of two different feature selection algorithms - Consistency-based Subset Evaluation (CSE) and Principal Component Analysis (PCA) techniques is discussed in Section 4.2.2 and pictorial representation is shown in the Fig. 4.4 and Fig. 4.5.

Classification: Classification is a technique where we assign a decision class label to a set of objects described by a set of attributes. The set of learning examples $S = < x_1, y_1 >, < x_2, y_2 >, ..., < x_n, y_n >$ for some unknown classification function f : $y = f(x)$

$X_i =< x_{i1}, x_{i2}, ..., x_{im} >$ where m is number of attributes and y is the class label.

Need of Multiple Classifiers: Choosing an appropriate algorithm for classification is not a straightforward approach because there is no one algorithm achieving the best accuracy for all situations.

"Multiple learning systems try to exploit the local different behavior of the base learners to enhance the accuracy of the overall learning system" – G. Valentini, F. Masulli

Multiple Classifiers: Is a set of classifiers where individual predictions from each classifier are combined in some way to classify new examples. These classifiers are also called as ensemble methods, classifier fusion, combination, aggregation, etc. This approach improves the prediction or classification accuracy. However, combining identical classifiers does not give better accuracy, because the classifiers might make `uncorrelated s` with respect to one another, each classifier should perform better than a random guess.

The intuition behind diversification of classifiers; if two classifiers are diverse, they make different errors on a new object. Assume a set of three classifiers m_1, m_2, m_3 and a new object x,

- If all are identical, then when $m_1(x)$ is wrong, $m_2(x)$ and $m_3(x)$ will also be wrong.
- If the classifiers are uncorrelated, then when $m_1(x)$ is wrong, $m_2(x)$ and $m_3(x)$ may be correct $->$ a majority vote will correctly classify x.

The diversification of classifiers are diverse in:

- Different training sets(i.e different samples or splitting)
- Different classifiers (i.e trained from the same data)
- Different attribute sets
- Different parameters choices (tree pruning, BP parameters, # neighbors in KNN etc.)

- Different architectures (like topology of ANN)
- Different initializations

Ensemble Approach: Learning algorithms that output only a single hypothesis suffer from three problems that can be overcome by an ensemble approach: the statistical problem, the computational problem and the representation problem [76].

A learning algorithm that suffers from the statistical problem is said to have `high ‘‘variance’’`. An algorithm that suffers from the computational problem is described as having `computational ‘‘variance’’`. And a learning algorithm that suffers from the representation problem is said to have `high ‘‘bias’’`.

- **The Statistical Problem:** arises when the hypothesis space is too large for the amount of availability data. In such cases, there are many different hypotheses with the same accuracy on the training data and the learning algorithm chooses only one of them. There is a risk that the accuracy of the chosen hypothesis is low on unseen data. A simple vote of all these equally-good classifiers can reduce this risk.
- **The Computational Problem:** arises when the learning algorithm cannot guarantee finding the best hypothesis within the hypothesis space.
- **The Representation Problem:** arises when the hypothesis space does not contain any good approximation of the target class(es).

There are different approaches to create multiple system like homogeneous and heterogeneous classifiers.

- **Homogeneous Classifier:** Uses the same algorithm over diversified data

sets. Algorithms like bagging, boosting, multiple partitioned data, multi-class specialized systems.

- **Heterogeneous Classifiers:** Different learning algorithms over the same data set.
  - Voting or rule-fixed aggregation
  - Stacked generalization or meta-learning

Ensemble Learning This section entails about the approaches implemented in this thesis. The approaches undertaken are *Staking* and $Voting$, which are two of the most well-known ensemble approaches which use heterogeneous classifiers.

– *Stacking* : This technique consists of three layers as shown in Fig. 4.1. In the first layer, it takes the training dataset as input. The second layer, the input dataset is fed to several 'base classifiers' to obtain the individual predictions. In the third layer, a 'meta classifier' is employed for the final prediction. It is hard to analyse theoretically.

– $Voting$: This approach is similar to that of *Stacking* till the second layer as shown in Fig. 4.1. The only difference is that there is no meta classifier used in this technique. In the third layer, it combines all the probability estimations from the base classifiers and classifies an instance based on the majority vote.

The functioning of each of these layers is as described below:

- **Input Layer:** Contains a set of features as training data.
- **Base Classifier:** Multiple independent classifiers are used in this layer. This layer results in predictions of multiple classifiers. If the base learners can output probabilities, it's better to use those as input to meta learner.

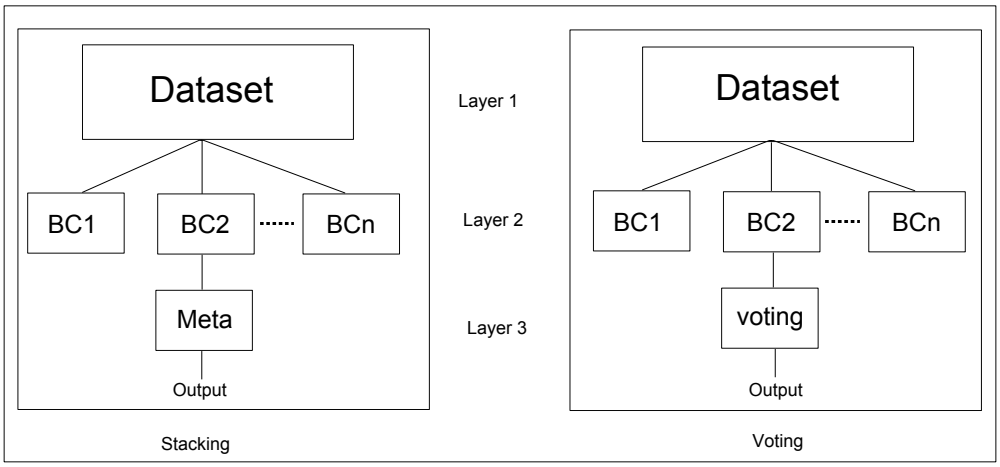


Figure 4.1: Stacked Learning

- **Meta Classifier:** The predictions of base classifiers on an extra validation set (not directly training set – apply internal cross validation) with correct class decisions $->$ a meta-level training set. An extra learning algorithm is used to construct a meta-classifier. It attempts to learn mapping between predictions and the final decision. It may correct some mistakes of the base classifiers.

In this approach, the base classifiers do most of the work and reduce risk of overfitting.

Ensemble Learning Model for Traffic Classification An ensemble technique combines multiple models whose individual predications are fused to obtain better predictive performance than the individual predictions [77]. The proposed ensemble learning model for P2P traffic classification is as shown in Fig. 4.2.

Flow and Flow description: Packets are categorized into flows using 5-tuple,i.e., source IP, source port, destination IP, destination port and protocol. As most of the time flows are bi-directional in P2P networks the direction of the flow is de-

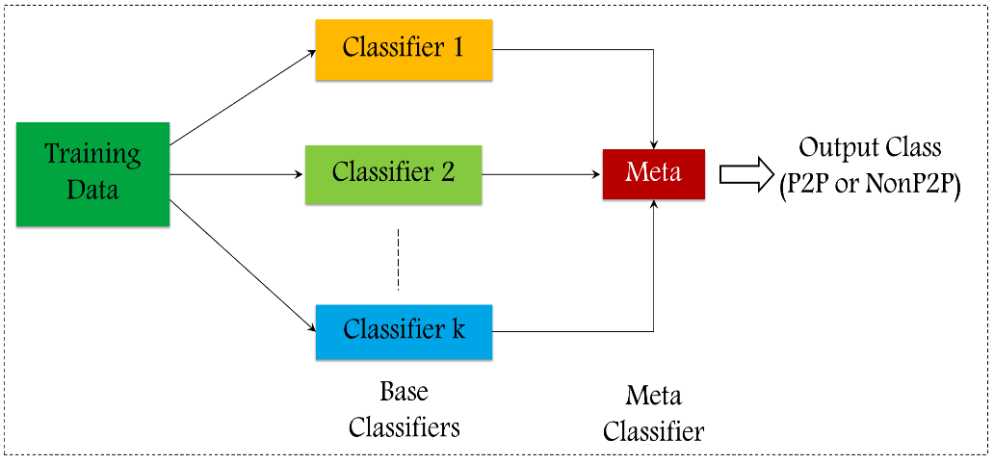


Figure 4.2: Stacked Learning

cided when the first packet is noticed. In general, a TCP flow is completed when either a connection timeouts or FIN/RST flags are set, while a UDP flow is terminated based on a heuristic value of 600 seconds [59]. Feature extraction module is implemented in Java using jNetPcap library.During the feature extraction we dropped few flows like TCP three-way handshake, flows with zero payload length etc., which do not contribute significantly to P2P traffic classification activity. However, flows with minimum of four packets are considered that describe P2P traffic behavior suitably. Algorithm 1 details the construction of flow from TCP and UDP protocols.

4.2.1 Background of Feature Selection

Feature selection is the process of reducing the number of features, with the aim of removing those features from the learning algorithm which have low impact on the classification problem. Primary motivation behind feature selection is that the training data contains many features which are either redundant to the classification problem, i.e., they provide no further information than the currently

Algorithm 1: Conversation

```
Conv c;
        HashSet< Conv > cset;
        Map< String, Conversation > ConvMap;
        Data: getConv(ArrayList < Packet > list) input is a list of
        packets.
Result: Set< Conv > convSet; conversations in convSet.
begin
    cset ← new HashSet< Conv >()
    ConvMap ← new HashMap< String, Conv >()
    for Packetp ∈ list do
        if ConvMap.containsKey(p.getSige()) then
            if p.isTcp() and (p.getTcpflag()[7]||p.getTcp − flag()[5]) then
                ConvMap.get(p.getSig()).addPacket(p)
                ConvMap.remove(ConvMap.get(p.getSig()))
            else if p.isTcp() ≠ ∅ then
                if
                 p.getTstamp() − ConvMap.get(p.getSig()).last ≥ timeOut
                 then
                    ConvMap.remove(ConvMap.get(p.getSig()))
                    c = Conv(p.getSIp(), p.getDtIp(), p.getSPort(),
                     p.getDPort(), p.isTcp())
                    c.addPacket(p)
                    cset.add(c)
                    ConvMap.put(p.getSig(), c)
                else
                    ConvMap.get(p.getSig()).addPacket(p)
            else
                ConvMap.get(p.getSig()).addPacket(p)
        else
            c ← Conv(p.getSIp(), p.getDIp(), p.getSPort(), p.getDPort(),
             p.isTcp())
            c.addPacket(p)
            cset.add(c)
            ConvMap.put(p.getSig(), c)
    for Conv co ∈ cset do
        co.updateValues()
        co.freeSpace()
    return cset
```

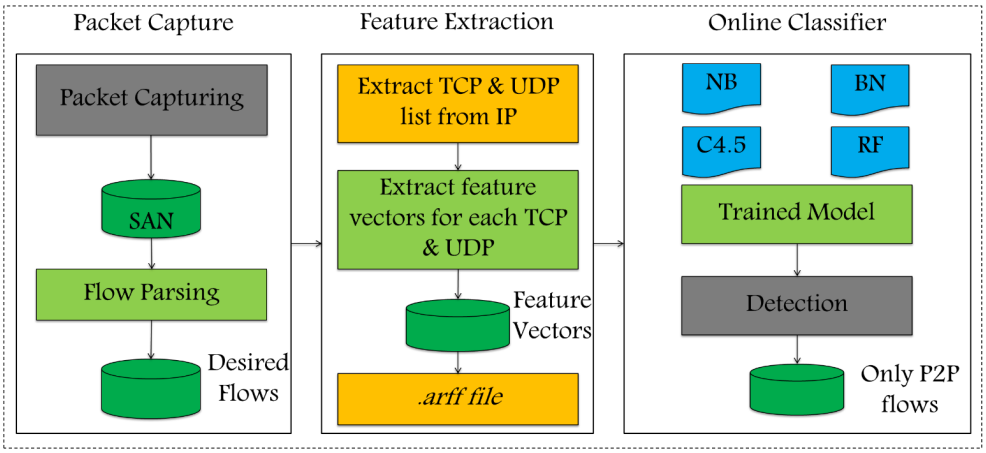


Figure 4.3: Proposed Detection Framework using ensemble learning

selected features or are totally irrelevant to the classification problem itself. Consider the simple example of features that may be extracted from network flows for the purpose of classification of network traffic. Several features can be extracted from every flow like average packet size, duration of the flow, number of unique ports used, TCP flags, TCP window size, etc. But certain features are not at all relevant to the task of classifying network traffic. For example, every TCP packet receives an acknowledgement in response. A count providing the number of ACK packets is not a good feature for classifying network traffic because it cannot help in distinguishing between differentiating applications.

Reducing the number of features provides direct benefit in terms of lesser training time for the learning model. Such 'feature selection' is also known to reduce the problem of 'over-fitting' or the variance error [26] and is also critical to overcome the class imbalance problem [27]. It should be noted that although the final accuracy of the learning algorithm will depend on the learning technique used, suitability of the original features (i.e., those obtained prior to use of feature selection) etc., feature selection techniques are effective in optimizing the performance of the classifier drastically reducing the number of features used for classification.

4.2.2 System Implementation

Our approach using ensemble learning:

– Step I: We captured the network traffic at our core switch and stored them in a Storage Area Network (SAN).

– Step II: Once desired captures are obtained, we preprocess the traces and filter only IP packets. Then, we obtained desired flows to extract features. We extracted a total of 24 features from each bi-flow (i.e. a pair of unidirectional flows between the same 5-tuple) and stored them into comma separated values (CSV) format.

– Step III: After feature extraction, we labelled each instance according to class as P2P or Non-P2P.

– Step IV: We used ensemble learning technique to classify network traffic. In the off-line phase, we experimented with Naïve Bayes, Bayesian Networks, Decision Trees as base classifiers and Random Forest as decision classifier. In on-line phase, we classified unknown samples (i.e. no class label is assigned to it) using model built from off-line phase.

A total of twenty three features are extracted from each flow. The twenty three statistical features that we considered include features shown on Table 4.1.

The proposed P2P network traffic detection framework is shown in Fig. 4.3. This hierarchical framework consists of three main components. The packet capture and feature extraction module involved processes like flow parsing, and extracting feature vectors into an `.arff` file (Weka format) which is later used to train the classifiers (Naïve Bayes, Bayesian Network, C4.5 and Random Forest). Stacking is then used as a fusion technique to improve upon the classification accuracy

Table 4.1: Flow-based features

Feature Description	**Abbreviation**
Packet Count (sent)	spkt
Packet Count (Receive)	dpkt
Flow Duration (sent)	sflowdur
Flow Duration (Receive)	dflowdur
Minimum Packet Size (sent)	sminpkt
Minimum Packet Size (Receive)	dminpkt
Maximum Packet Size (sent)	smaxpkt
Maximum Packet Size (Receive)	dmaxpkt
Mean Packet Size (sent)	smeanpktsize
Mean Packet Size (Receive)	dmeanpktsize
Standard deviation of Packet Size (sent)	sstdpktsize
Standard deviation of Packet Size (Receive)	dstdpktsize
Minimum Inter-arrival Time (sent)	sminiat
Minimum Inter-arrival Time (Receive)	dminiat
Maximum Inter-arrival Time (sent)	smaxiat
Maximum Inter-arrival Tim (Receive)	dmaxiat
Mean Inter-arrival Time (sent)	smeaniat
Mean Inter-arrival Time (Receive)	dmeaniat
Standard Deviation Inter-arrival Time (sent)	sstdiat
Standard Deviation Inter-arrival Time (Receive)	dstdiat
Total bytes (sent)	stalbytes
Total bytes (Receive)	dtalbytes
Transport Protocol	proto

by combining the predictions of NB, BN (base classifiers) and C4.5 as the meta-classifier, as per the structure shown in Fig. 4.2. To improve the output of Stacking three base classifiers NB, BN and C4.5 are used followed by RF as the meta-classifier. To further compare the performance of Stacking with other learning algorithms, Voting scheme [66] that computes average probabilities of predictions made by NB, BN, C4.5 and Random Forest is used.

Feature Selection:

Feature selection is the process of reducing the number of features, with the aim of removing those features from the learning algorithm which have low impact on the classification problem.Primary motivation behind feature selection is that the training data contains many features which are either redundant to the classification problem, i.e., they provide no further information than the currently selected features, or are totally irrelevant to the classification problem itself. Consider the simple example of features that may be extracted from network flows for the purpose of classification of network traffic.Reducing the number of features provides direct benefit in terms of lesser training time for the learning model. Such 'feature selection' is also known to reduce the problem of 'over-fitting' or the variance error [26], and is also critical to overcome the class imbalance problem [27]. It should be noted that although the final accuracy of the learning algorithm will depend on the learning technique used, suitability of the original features obtained (i.e., those obtained prior to use of feature selection) etc., feature selection techniques are effective in optimizing the performance of the classifier since the number of features used for classification are reduced. In [78] authors shown that feature reduction techniques are able to greatly reduce the feature space and at the same

Algorithm 2: Best first search algorithm

Begin with the OPEN list containing the start state, the CLOSED list empty, and BES ← start state.
Let s = arg max $e(x)$ (get the state from OPEN with the highest evaluation).
Remove s from OPEN and add to CLOSED.
if $e(s) \geq e(BEST)$ **then**
  BEST ← s.
for $childtofsnotintheOPENorCLOSElist$ **do**
  evaluate and add to OPEN.
if $BESTcahngedinthelastsetofexpression$ **then**
  select the state from OPEN.
return BEST

time increase the computation performance.

In this work we use two different algorithms to reduce the feature set: CSE and PCA. CSE evaluate different combination of features to create an optimal feature subset. The feature subset is generated using different search techniques. We use Best First Search in CSE is discussed below. With CSE and BFS, optimal features with eight feature subset (reduced from full feature set of 23 features) are obtained.

BFS: BFS [79] is an AI search strategy that allows backtracking along the search path. It generates an optimal subset based on the addition or removal of features to the current subset. However, it has ability to backtrack along the subset selection path to explore different possibilities when the current path no longer shows improvement. To prevent the search from backtracking through all possibilities in the feature space, a limit is placed on the number of non-improving subsets that are considered given in Algorithm 2.

PCA reduced the feature set from 23 to 11 features are extracted with 95% of variance which are in linear combinations is listed in Table 4.2. With these features

Table 4.2: Attribute Selection with CSE and PCA

Attribute Selection	**# of Features**	**Feature Description**
CSE	8	source max pkt length, destination mean pkt length, source min inter-arrival time (iat), source max iat, destination max iat, destination mean iat, source total volume, destination total volume
PCA	11	linear combinations with 95% of variance

obtained from CSE and PCA, Stacking and Voting mechanisms were put to use.

Three models,in total, are obtained from each of the ensemble learning that encompasses two from feature selection algorithms and one from full feature set. To evaluate the algorithms, k-fold cross-validation is implemented(in this work, k=10). This cross-validation technique divides the training data into k subsets. Each time k-1 subsets are used for training and one of the k is used as the test data. To measure the classifier performance the overall accuracy is generally considered as an acceptable metric. However, for a better understanding of the classifier's performance one has to calculate the recall and precision for each class in addition to the overall accuracy metric.

4.3 Result Analysis

Our goal is to show the impact of multiple classifiers (Staking and Voting) with two feature reduction on the relative computational performance of our chosen ML algorithms. The results have demonstrated that the classification accuracy is not significantly degrade by the use of reduced feature sets. We reduce feature

set using CSE and PCA subset evaluation. Then, we compare the computational performance of each tested ML algorithm with and without reduced feature sets. We then determine the best subset by comparing the average accuracy against the average accuracy and across the algorithms using the full feature set.

The results obtained using stacking and voting using NB, BN and C4.5 is shown in Fig. 4.4. The accuracy of full feature set is found to be 99.2% of staking and 99.4% of Voting. A larger reduction in the feature space is achieved with relatively change in accuracy. The subset evaluation algorithm CSE is found to be 98.69% with stacking technique and 99.81% with voting technique. Similarly, for PCA the accuracy obtained is 96.3% with stacking and 97.5% with voting.

The accuracy is little high in Stacking approach than Voting even with feature subset evaluation techniques. The recall (P2P) value of CSE and PCA is 99.89% and 99.6% of Stacking. Similarly the recall value of CSE and PCA is 99.7% and 99% of Voting. The results showed that CSE is better than the PCA as the subset evaluator to identify P2P traffic. There appears to be a very good trade-off between feature space reduction and loss of accuracy. We examine the impact of feature reduction no individual algorithms, in terms of accuracy and recall. Both Stacking and Voting approaches achieve more that 99% of accuracy using full feature set and there is little change when using either of the reduced feature subsets.

Experiments were carried out on four diverse classifiers, NB, BN and C4.5 as base classifiers followed by RF as the meta-classifier. Stacking with full feature set and the detection rate and the accuracy is 99.94% and 99.9% respectively. Using Voting approach with full feature set accuracy and detection rate results were found to be 99.7% and 99.7% respectively as shown in Fig. 4.5.

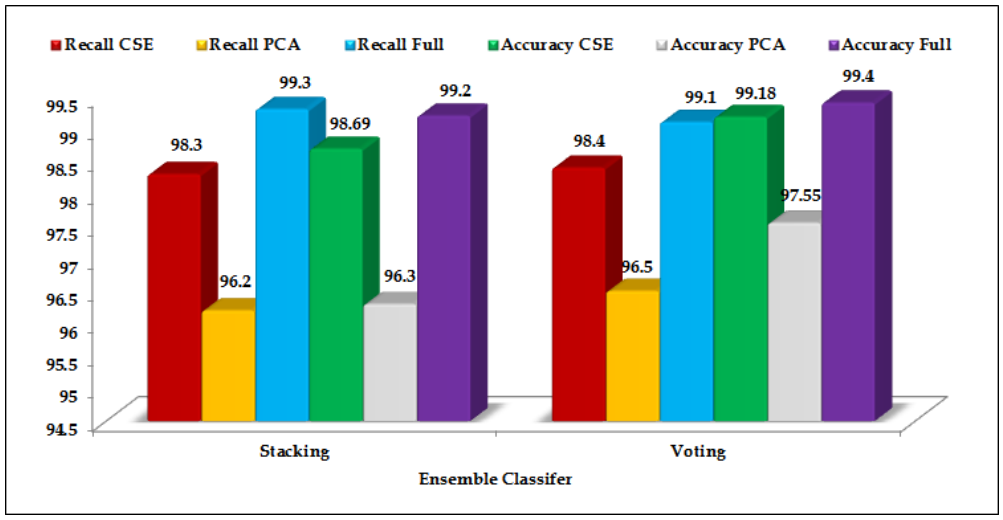


Figure 4.4: Accuracy and Recall of Stacking and Voting ensemble learning with NB, BN and C4.5

Using stacking, higher accuracy and detection rate than the Voting is ensured because the Voting scheme cannot have more number of base classifiers (in our work maximum three). It is because while predicting it combines outputs from all the base classifiers and evaluates the average probabilities to the output class which leads to either over-fitting or under-fitting which is a default limitation of Voting scheme. It is observed that the Stacking scheme performs better in this scenario than Voting (P2P traffic identification).

Tests were performed on server grade machine running Ubuntu 12.04 LTS. The computational time for building a ML model is measured with WEKA. The build time of Stacking and Voting for CSE and PCA is computed with the help of classifiers, such as, NB, BN, C4.5 and RF. The build time for Stacking and Voting is shown in Fig. 4.6. The computational time of various datasets is as follows:

- **Full feature set:** It is observed that 430 sec 99 sec for Stacking and Voting respectively.
- **CSE:** The build time of reduced subset using CSE is 156 sec for Stacking

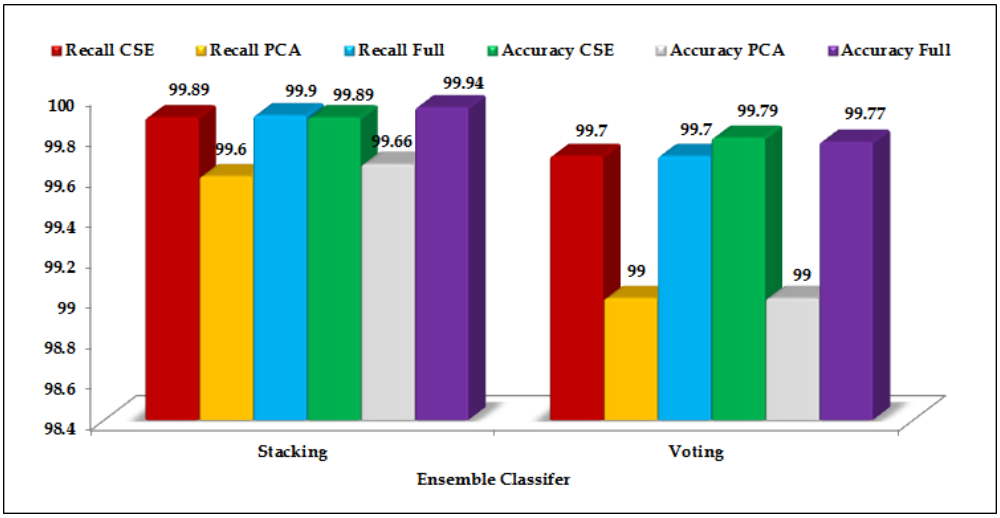


Figure 4.5: Accuracy and Recall of Stacking and Voting ensemble learning with NB, BN, C4.5 and RF

and 45 sec for Voting.

- **PCA:** The build time of reduced subset using PCA is 217 sec for Stacking and 59 sec for Voting.

4.4 Limitations

The P2P traffic classification is based on statistical properties of flows, and using ML techniques which is a promising method, these methods do not rely on port numbers and on on packet payloads. However, the success of such "statistical fingerprints" highly depends on the accuracy of the training data used. Ensuring accuracy and authenticity of the training sets is still an open issue, especially for disguised P2P flows.

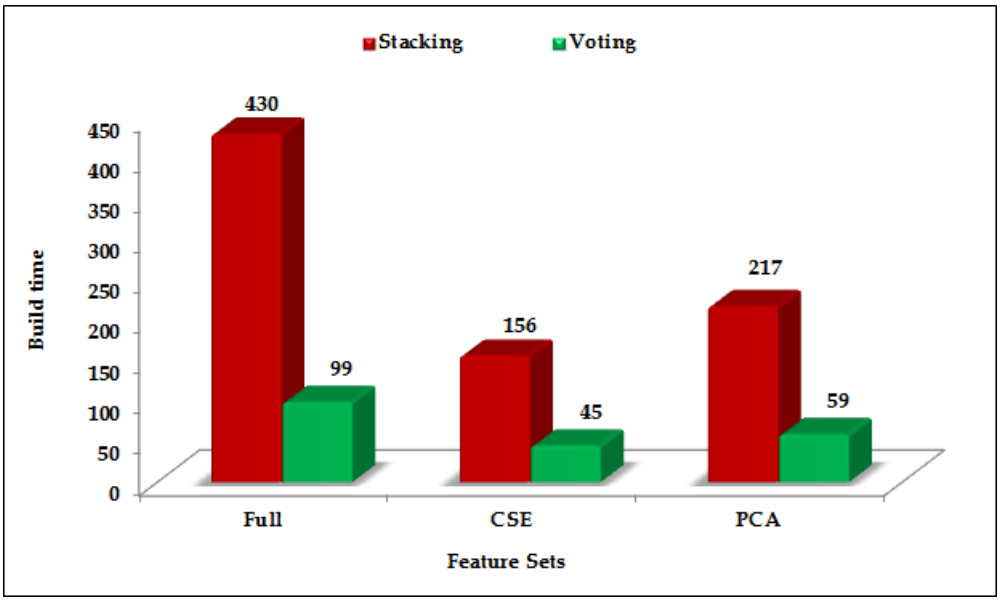


Figure 4.6: Build Time of Staking and Voting with Full and Two subset Features

4.5 Conclusion

In recent times, have seen an increased interest on development of ML techniques for P2P traffic classification. The existing research focuses on achieving higher accuracy of different ML algorithms. The process of defining appropriate feature, performing feature section, performing diverse ML algorithms and influence of this on classification and computational performance has not been studied. We obtained 24 flow features that are well accepted and simple to calculate by researchers from networking community. We evaluate the classification accuracy and computational performance of Stacking and Voting schemes with NB, BN, C4.5 and RF and with two additional feature subset section algorithms (CFS and PCA).

The experimental results, thus obtained, exhibited that stacking performs better over voting in all the scenarios. However, voting degrades its performance when more number of base classifiers are used because the final prediction is based on the

majority of the voting on the training dataset which leads to more false positives and negatives, but build time of voting is much lesser than the stacking. The result of this work has shown that multiple classifiers overcome the over-fitting of the model and also presented feature reduction greatly improves performance of the algorithms in terms of model build time and classification speed for most algorithms.

Chapter 5

Host-Based P2P Traffic Identification

5.1 Host-based P2P Traffic Identification using Heuristics

A heuristic is a technique to identify approximate solution when traditional methods fail to find exact solution. These heuristics ensure optimality, completeness, accuracy and precision, execution time. Host-based traffic identification approach allows us to improve the classification accuracy by examining the host activity. In this approach, heuristics were proposed that can evaluate the behavior of source and destination for both P2P and NonP2P traffic. The objective of the proposed technique is to improve the P2P traffic identification accuracy based on the application profile of the host. This technique observes the communication patterns of an end-host. This approach doesn't require any learning method as flow-based

approaches do.

5.2 P2P and NonP2P Applications

In this section, we discuss the behavior of two group of protocols, P2P and NonP2P. This section considered few popular P2P protocols like eMule, uTorrent, Skype and for NonP2P protocols like HTTP(S), SMTP, FTP, Dropbox. To distinguish between P2P and NonP2P, a series of heuristics are developed after understanding the behavior of these protocols.

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

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

the sever closes the second connection. eMule client and server both use UDP for keep-alive messages and for enhancing the search [80].

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

Skype is a Voice over P2P (VoP2P) application. It uses P2P networks to discover peers. It contains three main components, i.e., Skye Client (SC), Skye Server (SS), and Super Nodes (SNs) [81].

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

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

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

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

have used SN's data for the purpose of classification.

5.2.1 NonP2P

These applications typically follow a client/server behavior. The client always initiates a TCP connection and server responds to its' request. A standard web browsing connection (either HTTP or HTTPs) is accomplished by a 3-way handshake i.e. client initiates TCP's SYN flag to a web server over port 80 or 443 and then server replies with TCP's $SYN - ACK$ flag. Then client acknowledges back with ACK flag. After the 3-way handshake is completed the client requests the desired information from the server. Here, the requesting host is always a client and responding host is a server. This activity is common for most of the web applications.

There are few applications which are slightly different in terms of using port numbers 80 and 443. File transfer protocol (FTP) also is a client/server protocol. FTP uses two control channels, one channel for control information that the server accepts from the client over port 21 and the other channel is utilized for data transfer from server to client over port 20. Dropbox is a cloud based application that also adopts client/server behavior. A client host requests Drobox server to access the files over HTTPs and these files are then downloaded into the local host machine. This protocol is different from FTP service in a way where it can *sync* any shared file in the Local Area Network (LAN) without connecting to the Dropbox server. It can also broadcast the network over port number 17500 if the host enables the LAN *sync* option.

5.2.2 P2P

P2P applications are different from the traditional client-server applications. The motivation behind design of these applications is to share files amongst the collaborating peers over the IP layer. In this, a host can act as a client and also as a server to maximize their file sharing benefits. In this section we briefly discuss about the most common P2P networks like Gnutella, eMule, Skype and μTorrent. These protocols never generate any DNS queries once these are executed over a P2P network which is a major difference between P2P and NonP2P.

5.2.3 Framework

We imported packet information into the MySql database. Then, connection patterns were obtained from P2P and NonP2P. Fig. 5.1 shows the framework of our approach to develop the heuristics. A Java interface is implemented to query the database and develop heuristics as given in Algorithm 3.

In later sections, the usage of our heuristics are discussed vividly. Then flows are constructed using 5-tuple i.e., source IP, source port, destination IP, destination port and protocol. TCP flows are separated by `FIN`, `ACK` or `RST` flags, whereas UDP flows are terminated based on Timeout of 600 seconds [59].

Network traces were independently collected from our campus LAN that is connected to the Internet by a 155 Mbps STM link. NonP2P (HTTP, HTTPs, and SMTP) applications traffic and P2P applications traffic (Torrents, Gnuetlla) were captured using Wireshark [82] tools. Due the privacy concerns, first 130 Bytes were captured of each packet. Around 250 GB of data is collected on our testbed. FTP dataset was obtained from Lawrence Berkeley National Laboratory

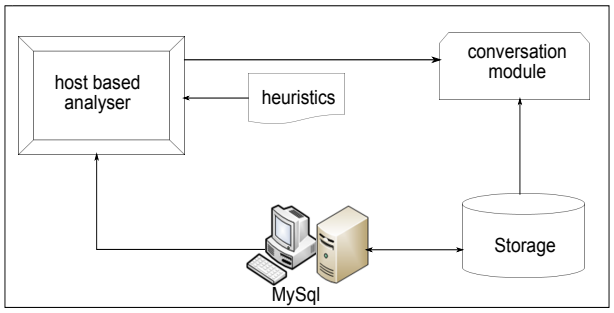


Figure 5.1: Framework for P2P Traffic Identification

Algorithm 3: hostAnalyser

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

(LBNLab) [83] with no payload information and the IP addresses obtained were anonymized. Part of the P2P traces used in this work was processed data from University of Georgia (UGA) [84]. Statistics of NonP2P and P2P application traffic in the datasets provided is shown in Table 5.1.

5.2.4 Proposed Heuristics

The heuristics are proposed when the flow based approach is failing to detect P2P traffic. 1. When the flow based approach failing the exact solution, heuristics solve more quickly than the flow based approach. 2. It finds the approximate solution to classify P2P traffic. 3. When compared to flow based, heuristics generates

Table 5.1: Application wise statistics

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

tiny rules that the system is robust enough to classify the traffic. 4. The flow based approach should accumulate all the packets and calculate the behavioural features, whereas the heuristics can be generated at individual packets.

In this section P2P traffic heuristics are explained based on the host behavior that were observed from the dataset collected. Heuristics were formulated to contain some well-known port number information as well. Tunable thresholds were derived that provide trigger to application of the heuristics used. The P2P traffic has been classified from the proposed heuristics and the false positives (FP) were identified. The fact that this piece of work differs from other related works is in terms of using lesser number of packets in the flows. These heuristics are tested in real-time by giving one minute intervals by running both P2P and NonP2P applications in the campus backbone. Few other types of traffic seen on the campus LAN were filtered, i.e., services like Network Basic Input/Output System (NETBIOS), Network Time Protocol (NTP), Dynamic Host Configuration

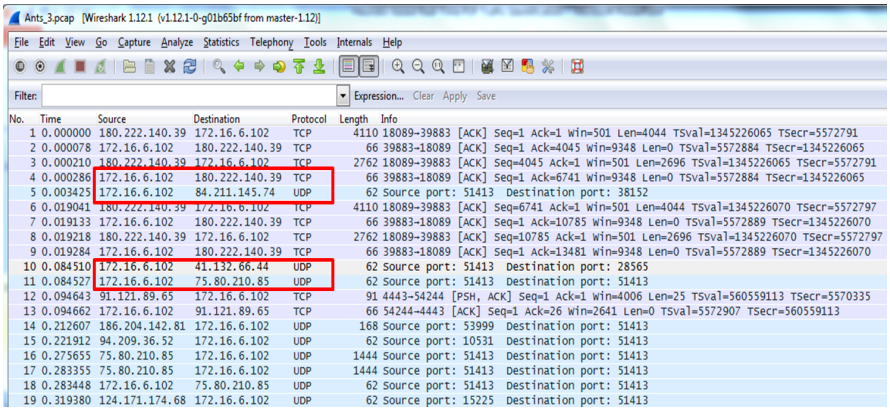


Figure 5.2: Heuristic A

Protocol (DHCP), Simple Service Discovery Protocol (SSDP), and Link local from both TCP and UDP protocols. The heuristics used are described as below:

A: **P2P TCP/UDP protocols:** This heuristic is based on the fact that the P2P applications like Gnutella, Skype, etc. use both TCP and UDP protocols. In most of the scenarios the TCP is used for data transfers whereas UDP is used for signalling messages. There may be an FP with this heuristic with UDP like default LAN services. However NonP2P applications make communication parallel over TCP. This is shown in Fig. 5.2.

B: **P2P UDP ports:** This heuristic exploits port based classification where the peers in the P2P network use well-know ports like port 80 and 443 over UDP to communicate outside the servers which can bypass the firewall. In general, these ports are being used with TCP connections for retrieving web contents. If any host uses these port numbers to and from, that host is marked as P2P and all these flows are classified as P2P. Experimental results reveal that the P2P hosts use these ports quite often in their lifetime. This is shown in Fig. 5.3.

C: **P2P Source Port and Destination Port:** In general all most all the P2P

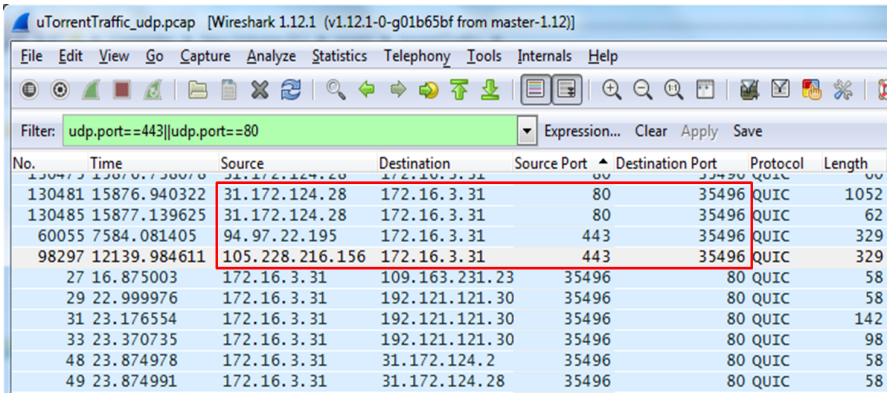


Figure 5.3: Heuristic B

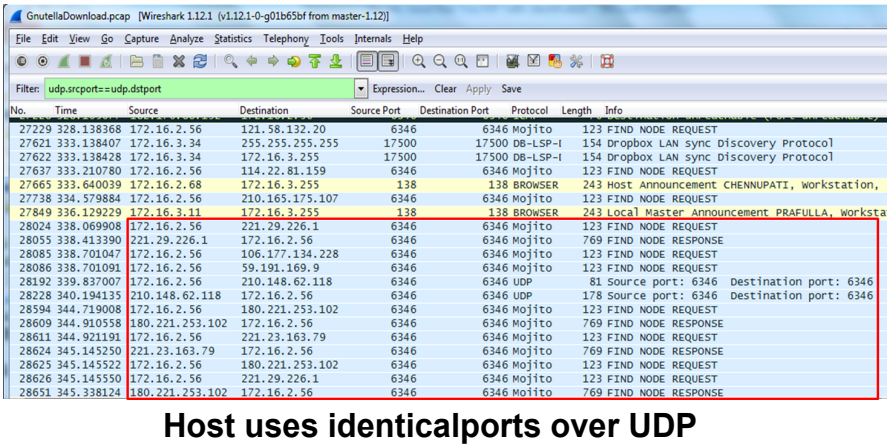


Figure 5.4: Heuristic C

applications allow change in port numbers or the application itself will use a random port to signal the peer in the overlay. However, the P2P application has the following property. Both source IP and destination IP communicate with the same port over UDP protocol. This property is unique for UDP, but not for TCP. All flows to and from these hosts are classified as P2P. This is shown in Fig. 5.4.

D: **P2P UDP/TCP port pairs:** The fact that, P2P applications use UDP for signaling or control messaging, any host can communicate using a default port or an ephemeral port over UDP on the destination side, the same port can

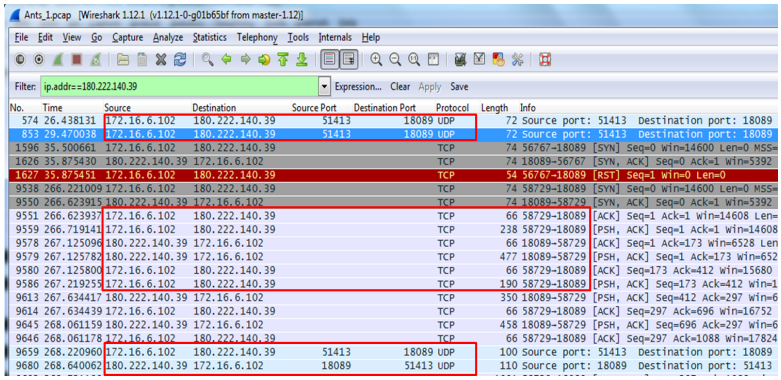


Figure 5.5: Heuristic D

be assigned to TCP for data transfer. This property is unique in NonP2P applications. All the TCP and UDP flows directed to and from a host are classified as P2P hosts. This heuristic can be applicable for both source IP and destination IP pairs, if both of them are involved in sharing of files in the overlay network. For example, a host (source/destination) communicates on port (for example: 3222) over UDP and for data transfer the other end opens same port (for example: 3222) to TCP. This heuristic when used classified Dropbox traffic wrongly as P2P traffic. This is because of the Sync behavior of Dropbox. This is shown in Fig. 5.5.

E: **P2P Unique IPs and Unique Ports:** When the P2P client initiates a connection in the network it signals multiple peers. At this point, it opens multiple port numbers to get the information about other peers over UDP and this will continue till the desired information is obtained. In our analysis, if a host is a P2P host, it must use TCP and UDP and the number of unique destination IPs are less than the number of unique ports and all flows are classified as P2P. On the other-hand NonP2P applications typically use multiple connections with a web server over TCP. Typically, FTP and SMTP

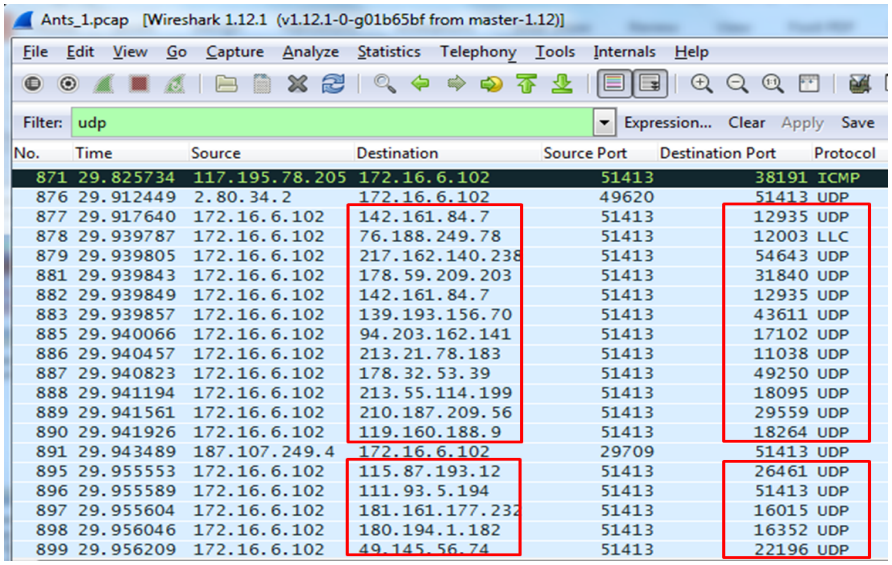


Figure 5.6: Heuristic E

servers accept multiple connections over the same port, but geographically, the servers communicate with distinct IPs and distinct port numbers over TCP, which are marked as NonP2P hosts, i.e. the host must use TCP and the number of unique IPs is less than or equal the number of unique ports. This is shown in Fig. 5.6.

The performance of our heuristics has been validated to identify P2P hosts in real-time. To justify this approach, we implemented the algorithms in Java using jNetPcap library [75]. The average detection rate is found to be more than 99%. The amount of unknown traffic is around 0.2%. The total number of unclassified flows, for Vuze is about 0.01%, μTorrent is about 0.01%, Skype is about 0.84%, Frostwire is about 0.02% and eMule is about 0.04% as shown in Fig. 5.7. Total 594 flows remain unclassified as P2P traffic out of 20,91,879.

Real-Time Analysis objective is to identify P2P and nonP2P flows even if a host uses both applications simultaneously. In this section we discuss the effec-

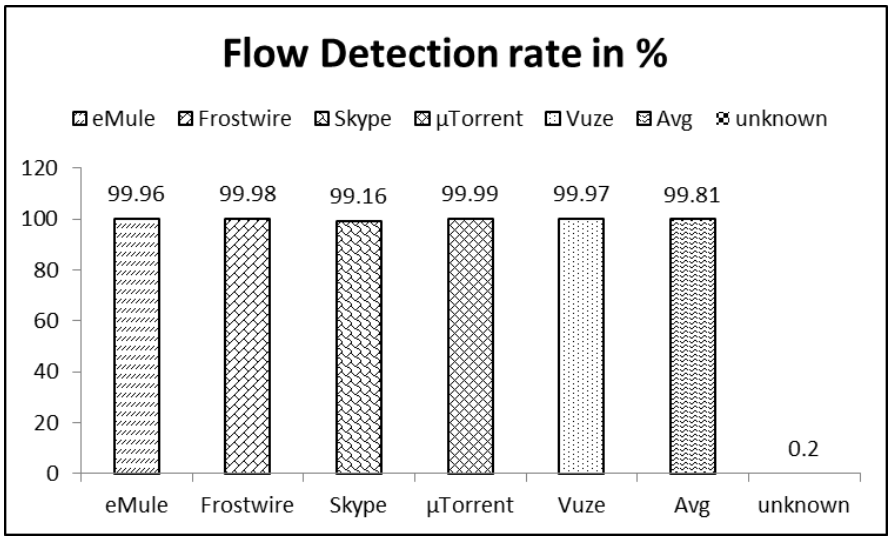


Figure 5.7: Detection rate of P2P traffic

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

5.3 Flow Based Vs Host Based approaches

In this section, the flow and host based traffic classifications according to different parameters were compared. In the experiments undertaken, it is observed that

Table 5.2: Flow vs Host based approaches

Parameter	**Flow Based**	**Host Based**
Extraction	Based on 5-tuples	based on IP address
Learning	is required	no learning
Volume of Data	large (approx. in GBs)	less (approx. in MBs)
Time	more build time	no build time
Performance	accurate, when enough data	accurate, when patterns are known

performance of host based approach is much better when compared to flow based approach discussed in previous sections in terms of metrics. Comparison of flow based and host based approaches is shown in Table 5.2.

5.4 Summary

In this work we proposed a new set of heuristics to identify a host as a P2P host based on its connection patterns. The proposed heuristics do not require any payload signatures. Our experimental result shows that the proposed rules are able to classify the P2P hosts effectively and suggested heuristics are promising. The dataset used as realistic in nature and we verified our approach in a real-time scenario too. We also presented the comprehensive behavior analysis of our P2P hosts. Further, our approach has minimal heuristics which can be deployed easily in real-time. The unclassified traffic is about 0.2% of the P2P traffic. Next Chapter 6 of this thesis covers a novel technique to identify P2P and NonP2P traffic using Fuzzy recognition based approach.

Chapter 6

P2P Traffic Identification: A Fuzzy Approach

6.1 Proposed Fuzzy Recognition System for P2P Traffic Detection

When distinctive features of the patterns are identified correctly, the classes can easily be distinguished in the feature space. With the rapid development of P2P applications, the patterns are overlapping with NonP2P traffic, thereby leading to an ambiguity in P2P traffic recognition. To overcome ambiguity, a well-know approach is fuzzy pattern recognition, capable of identifying patterns by deriving soft boundaries. The patterns can be classified into one or more classes with a certain degree of membership for TCP and UDP traffic. An algorithm for fuzzy pattern recognition is numerically demonstrated, in recognition of P2P traffic.

The previous chapters are proposed P2P traffic detection based on behavioral

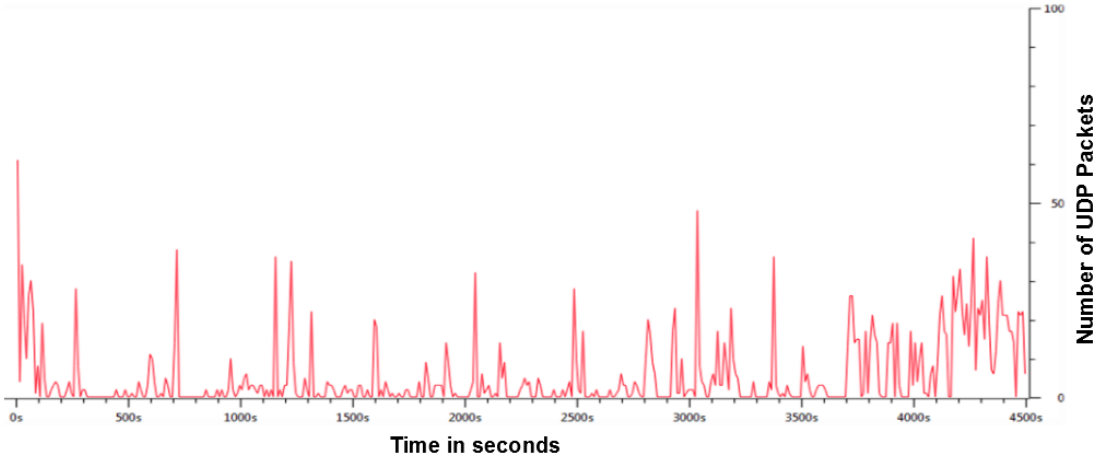


Figure 6.1: UDP packets sent by NonP2P applications.

characteristics obtained from the network traces. The proposed solution should detect P2P traffic as accurately as possible in reasonable amount of time. To achieve these objectives, our proposed system has 3 stages – packet accumulation, feature extraction, and fuzzy pattern recognition. Given network traces, the first stage accumulates only IP (IPv4 and IPv6) packets associated with every host and discards rest of the packets. The behavior-based features are extracted in the second stage whose details are described in the next section 6.1.1. Finally, the feature vector is passed through proposed fuzzy pattern recognition stage to detect P2P traffic.

6.1.1 Characterization of P2P Traffic

Our goal in this part of the work is to understand the behavior of P2P applications and to explore how these applications impact the underlying network characteristics when they are active. In our study we found most of the P2P applications use both UDP and TCP transport protocols over the network. We observed that decentralized P2P hosts like Freenet, eMule never use any DNS queries over UDP

protocol to discover the peers, on the other-hand hosts running torrents initiate DNS queries to find the domain of the tracker server. Whereas NonP2P hosts, e.g, hosts running http and https services resolve the domain name of the server. Fig. 6.1 shows an example of DNS query packets and Fig. 6.2 shows an example of peer discovery without DNS seen in our testbed. Similarly, both P2P and NonP2P hosts use TCP to transfer application data. We observed that the number of packets transferred (from server to client) in NonP2P hosts is more, whereas P2P hosts obtain packets from multiple sources concurrently. The behavior of P2P and NonP2P over TCP are shown in Fig. 6.3 and Fig. 6.4 respectively.

With these observations we extracted the following behavioral metrics to characterize P2P and NonP2P traffic. Given an input trace, our feature extraction module parses only IP, TCP and UDP headers and discards rest of the packets. Once these headers are checked for correctness then the module accumulates all the packets associated with every host. Thereafter the extraction module extracts the behavioral features for every host from the IP address list and stores them in a comma separated value (CSV) format. The feature extraction procedure is shown in Fig. 6.5.

A: **Bytes per Second:** Since P2P networks are mainly used for file sharing applications (text, video, audio, etc.), the volume of the file is larger per flow compared to NonP2P applications. Thus we measured the aggregate data transmitted between different P2P hosts.

B: **Packet Inter Arrival Time:** The Inter Arrival Time (IAT) of a packet is defined as the time gap between packets sent and received at a host. We observed high churn in P2P networks to discover the peers in the network i.e., the peers are joining and leaving the network. It is viewed as abnormal

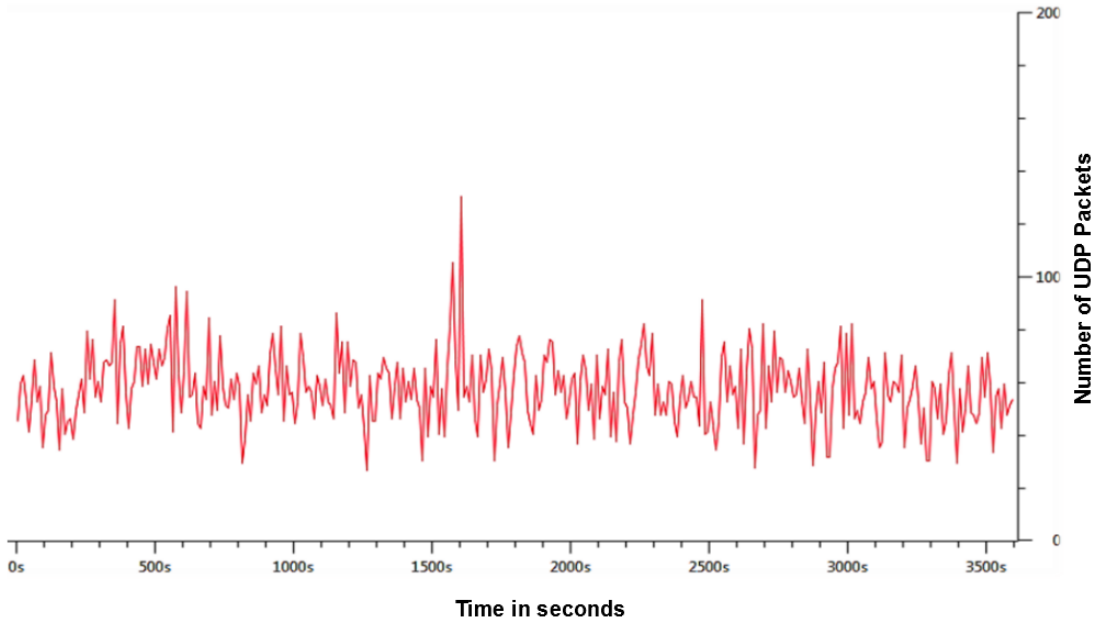


Figure 6.2: Peer Discovery/control messages sent by P2P Application over UDP.

behavior compared to a NonP2P host's behavior.

C: **Host Connectivity:** We observed the total number of unique IP addresses a host contacted in a day. P2P hosts make concurrent connections with other peers in the network whereas NonP2P hosts make parallel connections to the server.

6.1.2 Behavior-based P2P Traffic Detection

To detect P2P traffic, we employ fuzzy pattern recognition approach. In this process we extract the features from both TCP and UDP headers of a specific host. Our approach uses maximum membership function principle to characterize P2P behavior. According to this principle, from a given feature set, if the membership function for P2P traffic derives high values, the host is detected as P2P. Similarly, if the membership function for NonP2P derives higher values, the host is detected

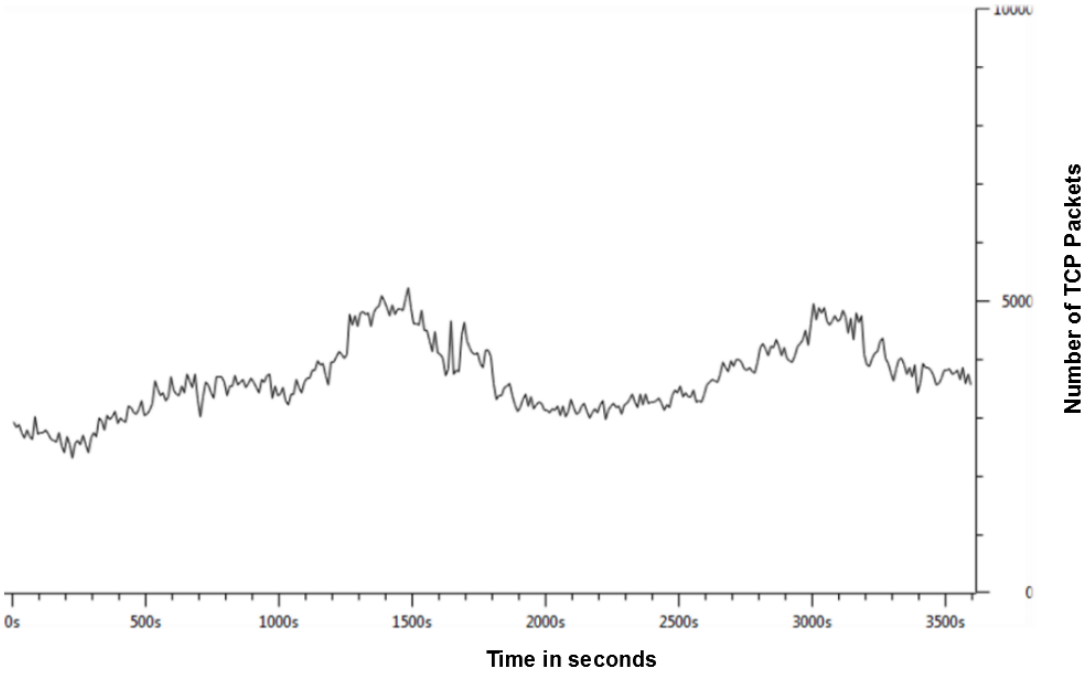


Figure 6.3: TCP behavior observed in P2P applications.

as NonP2P. Table 6.1 shows the summary of host-behavior features used for identifying the hosts as P2P or NonP2P. The membership functions are discussed in the following section.

6.1.3 Membership Functions for UDP and TCP Features

In this section we discuss fuzzy membership functions which detect P2P hosts. These membership functions are derived from a particular network trace. We define the membership functions for both TCP and UDP protocols as below:

i: **Normalized total number of sent and received packets:** A P2P host sends a large number of TCP and UDP packets to search for a file in the overlay network. Based on our analysis, we defined a membership function

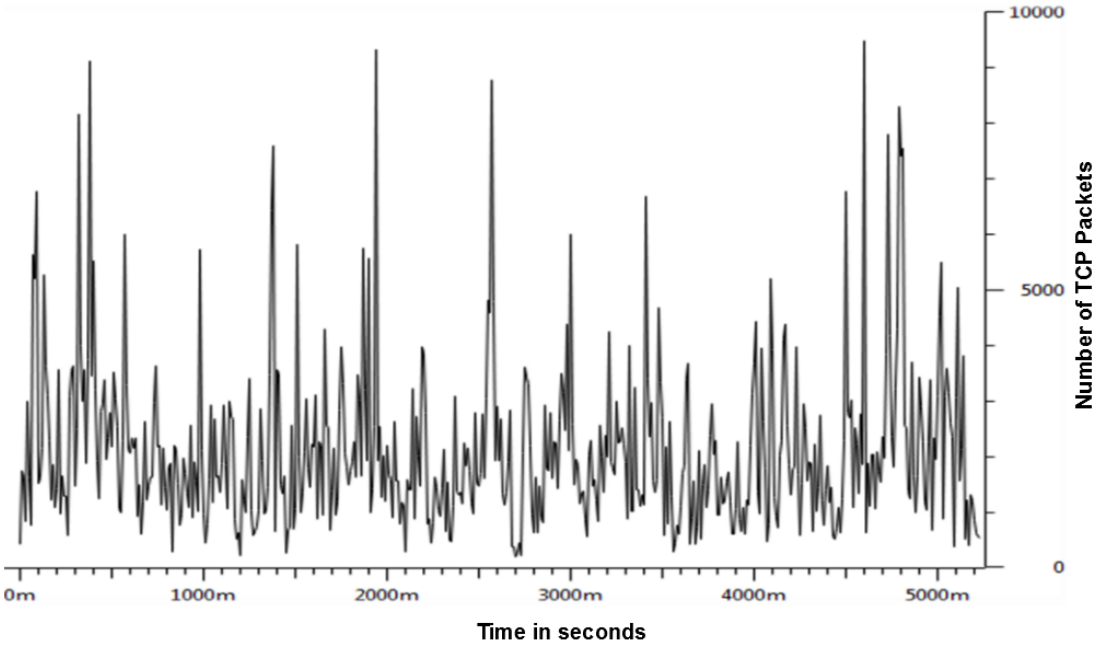


Figure 6.4: TCP behavior observed in NonP2P applications.

F_1 and F_3 to calculate the normalized packets sent and received as follows:

$$F_{1,3}(x) = \begin{cases} \frac{\left(\frac{S_p}{R_p}\right)}{x_{t1}}, & 1 < \frac{\left(\frac{S_p}{R_p}\right)}{x_{t1}} < 4.5 \\ 1, & \frac{\left(\frac{S_p}{R_p}\right)}{x_{t1}} >= 4.5 \\ 0, & otherwise \end{cases} \tag{6.1}$$

where S_p and R_p are the total number of sent and received packets in a given trace, and x_{t1} is the threshold for P2P host sent and received packets.

ii: **Normalized average of the total payload bytes sent:** In general P2P hosts send large number of packets in short period of time; we observed that the size of payload carried by each packet is small with equal sized packets. Therefore, we defined a membership function F_2 and F_4 for calculating

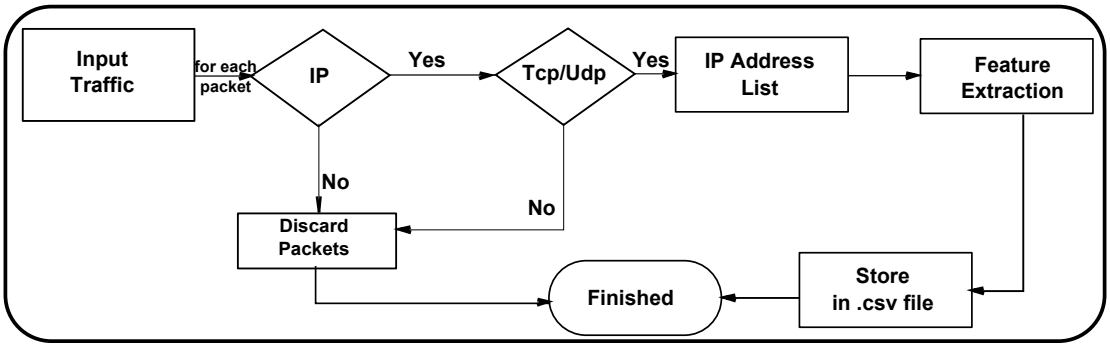


Figure 6.5: Procedure to extract Host feature

normalized average of the total payload bytes sent as follows:

$$F_{2,4}(x) = \begin{cases} \frac{\left(\frac{B_s}{n_i}\right)}{x_{t2}}, & 0.2 < \frac{\frac{B_s}{(S_p+R_p)}}{x_{t2}} < 1 \\ 1, & \frac{\frac{B_s}{(S_p+R_p)}}{x_{t2}} >= 1 \\ 0, & otherwise \end{cases} \tag{6.2}$$

$n_1 = S_{pt} + R_{pt}$

$n_2 = S_{pu} + R_{pu}$

where $i = 1, 2$ and B_s is bytes sent by a P2P host in a short period time, n_1 is the total number of TCP packets in the trace and n_2 is the total number of UDP packets in the trace and x_{t2} is the threshold for P2P host sent bytes.

The above membership functions define the P2P host activities over TCP and UDP. We define the membership function to calculate the probability of NonP2P host as below.

$$X = 1 - max(F_1(x), F_2(x), F_3(x), F_4(x)) \tag{6.3}$$

Table 6.1: Feature Description

Protocol	Feature Description	Feature
UDP	# of UDP sent and receive packets	F_1
	# of UDP bytes sent	F_2
TCP	# of TCP sent and receive packets	F_3
	# of TCP bytes sent	F_4

6.2 Performance Evaluation

6.2.1 Dataset Collection

To evaluate the performance of our proposed method, we collected two different datasets: P2P traffic generated by popular P2P applications and NonP2P traffic in the Information security laboratory at our department. The summary of the traffic collected is shown in Table 6.2. We collected the P2P dataset in a fully controlled environment. We choose the two popular P2P applications namely: eMule and μtorrent. Each application was run in different times and variety of files were uploaded and downloaded in order to increase the diversity of our dataset. NoP2P traffic was collected for the three different protocols namely http, https and ftp. We collected NonP2P traffic based on their standard port numbers at our Institution's border gateway.

6.2.2 Detection Accuracy

We used the collected dataset to evaluate the proposed approach. Table 6.2 summarizes the detection accuracy. We achieved low false positive rate of about 1.11% for data set 1, low false positive rate of about 2.49% for data set 2 and

Table 6.2: Detection Accuracy

	Data Set 1	Data Set 2	Data Set 3
Application Type	P2P (eMule)	P2P (uTorrent)	NonP2P (http, https, ftp)
# of Hosts	271	443	2406
# of Packets	2143 K	3876 K	1904 K
Capture Duration	3600 Sec	3600 Sec	318681 Sec
Correctly Classified	268	432	2405
Incorrectly Classified	3	11	1
Accuracy	98.89%	97.51%	99.96%
Error Rate	1.11%	2.49%	0.04%

low false positive rate of about 0.04% for data set 3. Among the total 11 false positives in the data set 2, we found that those are HTTP instances which made several connections to tracker servers.

6.3 Conclusion

In this paper, we presented a behavior-based P2P host identification approach using fuzzy patterns. This work improves our previous work in terms of detection accuracy for p2p traffic identification. The detection rate is improved by using fuzzy pattern recognition approach. Experiments show that our approach is able to detect more than 98% of P2P traffic and ≈2% of false positive rate was observed. However, our work is focused only on two P2P applications. Our future work will focus on further investigation of more P2P applications and define more fuzzy membership functions to even reduce the false positive rate further.

Chapter 7

Design and implementation of a P2P-aware firewall

7.1 P2P-aware firewall module

P2P-aware firewalls is proposed to monitor deeper into P2P traffic in terms of behavioural features than that of regular security system configurations. The regular firewalls can only monitor based on port numbers and (or) the payload signatures. P2P-aware firewalls adopted the regular security system features and unified flow based and behavioural features to classify P2P traffic on the fly. The drawback with P2P-aware firewalls are bit slower than the other firewalls systems.

The P2P classification module is used as "preprocessors" to enhance an IPtables based firewall. P2P-aware classification module was deployed to receive mirrored traffic from the border router of the BITS network. As such, they do not sit 'inline'. In order to take action on real-time traffic (in the form of blocking suspicious traffic, rate limiting P2P connections, etc.), an inline module is necessary.

The firewall modules was set-up as an inline module for this purpose.

IPtables is the default firewall available with Linux distributions. The firewall can take actions based on multiple parameters, such as – reject all traffic to a destination, drop all traffic to a destination, limit the traffic originating from P2P hosts, etc.

Traffic classification is performed by the classification modules. The results thus obtained are used by the 'firewall' module to generate automated, dynamic firewall rules. For the purpose of this thesis, IPtables based firewall was set-up on a server-grade Linux machine. This server was implemented as the gateway for the systems in the Information Security lab at BITS Hyderabad campus. The firewall rules are applied to this 'gateway' server. Being the gateway, this server sits 'inline' and can take real-time action with the firewall rules.

The 'firewall' module utilizes IPtables and bash scripts to create automated and dynamic firewall rules for hosts identified as running P2P applications. For the prototype implementation of our module, we implemented firewall rules for top 20 IPs inside and outside BITS campus identified as P2P. Apart from creating firewall rules pertaining to these set of hosts, we also created generic firewall rules to rate-limit connections of popular P2P applications. Our firewall rules have a dynamic nature. We do not block/reject P2P traffic on weekends. On working days, P2P traffic is disallowed during the working hours (9:00 AM – 5:00 PM). Apart from working hours, P2P traffic is allowed. Bot traffic, however, is shown zero tolerance and is rejected under all circumstances.

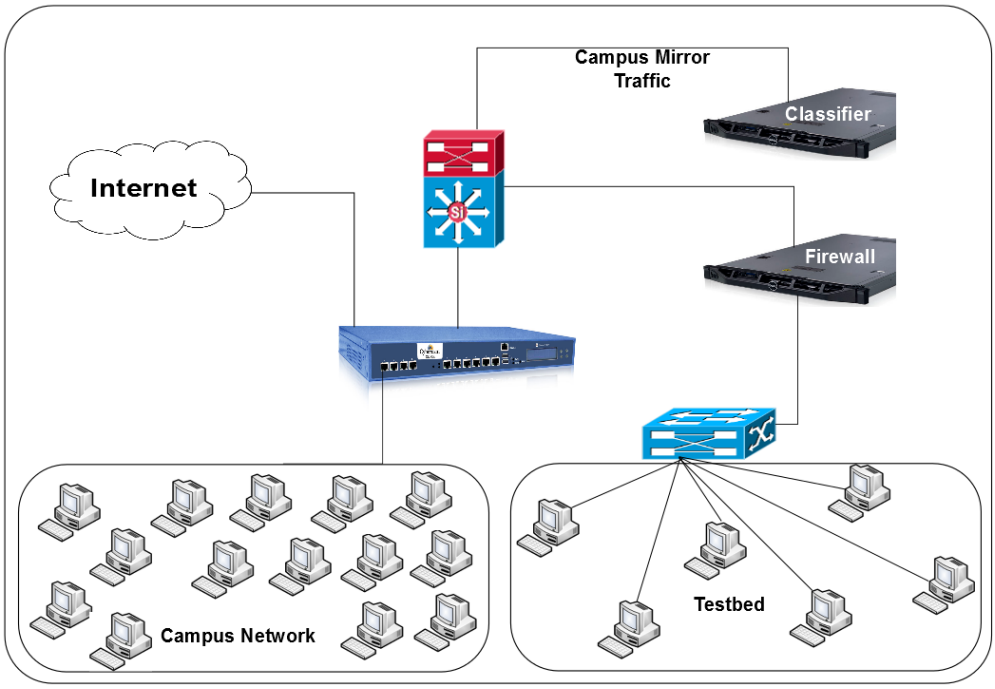


Figure 7.1: Overview of System Integration

7.2 System Integration

The P2P traffic classification module was integrated in the form of a live P2P traffic classifier and the output further goes to the firewall. The firewall module generates real-time, dynamic rules to block and/or rate-limit P2P traffic. A diagram of the entire integrated system is given in Fig. 7.1.

The integrated module has been deployed at the BITS network. The module receives mirrored traffic from the border router of the University. This mirrored traffic is captured on the server using a 'dumpcap', a libpcap (packet capture library) based tool. The `.pcap` files thus generates from dumpcap are limited to size of 50 MB. This size was chosen so as to achieve high speed of processing. Bigger file sizes will slow down the processing speed.

To achieve high speed of processing, we utilizes a parallelized implementation of

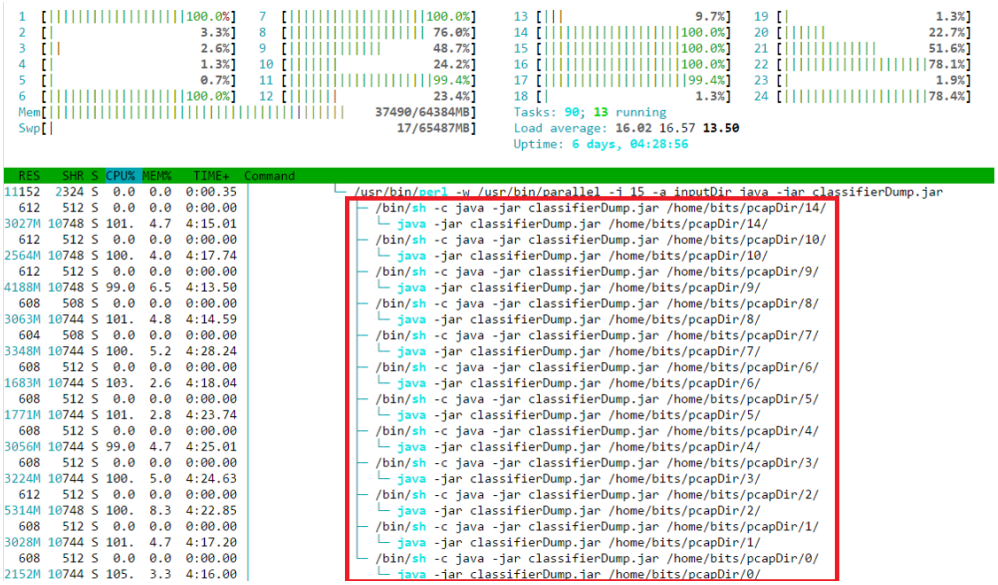


Figure 7.2: A snapshot of the P2P classification module invoked in parallel

different modules such as the P2P traffic classification module, `.pcap` file parsing module, etc. Multiple 'processes' of each module are invoked on different CPU cores of the server-grade machine. In this way, we are able to efficiently harness the processing power of server-grade machines.

The P2P classification module is invoked in parallel using the `GNU Parallel` utility. Fig. 7.2 captures the run of the classification module in parallel (marked in red). Since multiple modules run in parallel, we are able to process the `.pcap` files at high speed. One 50 MB files takes approximately 5–6 minutes to get processed. At the current network speed of the BITS network, a single 'process' of this module processes around 1,100 packets per second. Currently, owing to CPU core limitations, 15 processes of the P2P classifier are run. This amounts to a classification speed of around 16,500 packets per second. A better picture of this classification speed will come out when we compare our module with open-source IDS Snort in the next section.

The P2P classification module uses ensemble learning approach to classify flows as 'P2P' or 'Non-P2P'. Non-P2P flows are discarded, while P2P flows are written out to the database.

This integrated system was compared for performance against open-source IDS Snort. Network traces totalling more than 200 GB were collected from BITS network by regular sampling over a period of more than one month. Our modules was compared to Snort on the basis of multiple parameters, as mentioned below:

- P2P detection: Both Snort and our modules were successful in accurate detection of P2P applications running in the campus.
- Payload-oblivious classification: Since content-matching in the payload lies at the heart of Snort's detection engine, Snort cannot perform privacy-preserving, payload-oblivious classification. We tested Snort capability by supplying a 100 GB of pcap files to it which were collected from the BITS network. These pcap files were purposefully captured with payloads truncated to first 150 bytes. This was done because it is commonly seen that many organizations, owing to privacy concerns, share data only after truncating the payloads. Therefore it becomes important that a detection module can handle such data. Snort's detection engine failed to understand this data, and around 50% of the data was simply discarded by Snort. Our modules, on the other hand, face no problem in dealing with such data. Our modules extract feature from packet headers only, and are thus privacy-preserving.
- Speed: Snort is able to process network traces at very high speeds. Snort directly listens for traffic at the network interface, processes data packet-by-packet, and writes out data in binary format. This way, it is highly optimized

and thus attains high speed. With the dataset fed to Snort, we observed a processing speed of 34,268 packets per second. Our module in contrast to Snort, does not process data packet-by-packet. We extract several statistical features discussed in Chapter 4 from a flow. This creates some time latency. The classification speed of a single process of P2P classification module is about 1,100 packets per second. However, we have implemented our modules to be 'multi-process'. This speed can be scaled up as a factor of CPU cores available on the machine. At present, 15 cores are being used by the classification module, giving a speed of 16,500 packets per second. With more CPU cores, our module can outperform Snort.

The entire integrated system was deployed at the Hyderabad campus of BITS Pilani. The system is running continuously, with the P2P classification module detecting suspicious activities, the firewall module generating dynamic firewall rules.

Given below are the rules written using IPtables.

All traffic to the top 20 P2P IPs (as detected by our module) outside the BITS network is outrightly rejected so that any attempt from inside the BITS network attempting to connect to them is denied:

```
iptables -I FORWARD -t filter -d $ip -j REJECT
```

Further, the traffic from top 20 P2P hosts inside the BITS network is limited to 3 connections per host:

```
iptables -I FORWARD -s $ip -m connlimit --connlimit-above 3
-j REJECT
```

All P2P traffic is permitted during weekends, and blocked during the working

hours of weekdays. P2P traffic is blocked by rejecting all traffic on known standard ports of popular P2P applications. After working hours, this traffic is again permitted. Since IPtables processes rules in the order in which they are written. This rule is given here:

```
iptables -I FORWARD -p tcp -m time --timestart 09:00:00
--timestop 17:00:00 --weekdays Mon,Tue,Wed,Thu,Fri -m multiport
--sports 411,412,2323,6347,1214,6346,4662,6881,6889,6699 -j REJECT
```

Certain P2P applications may escape this rule by using ports other than those mentioned above. For this purpose, we reject all traffic to the top 20 P2P IPs during the working hours of weekdays:

```
iptables -I FORWARD -t filter -d $ip -m time --timestart 09:00:00
--timestop 17:00:00 --weekdays Mon,Tue,Wed,Thu,Fri -j REJECT
```

The rules given above are explained in Algorithm 4 below:

Algorithm 4: IPtables pseudo-code

$Weekday \leftarrow [Mon, Tue, Wed, Thu, Fri]$;
$P2P\_ports \leftarrow [411, 412, 2323, 6347, 1214, 6346, 4662, 6881, 6889, 6699]$;
begin
  while do
    if IP *in* $Top20\_P2P\_IPs$ **then**
      $REJECT$ traffic to IP
    else if IP *in* $Top20\_local\_P2P\_IPs$ **then**
      $Rate\_limit$ traffic to IP to 3 connections
    else if $Time$ *in* $[9:00-17:00]$ *and* $Weekday$ **then**
      $REJECT\ TCP$ traffic on $P2P\_ports$
    else if IP *in* $Top20\_P2P\_IPs$ **then**
      if $Timein \quad [9:00-17:00]$ *and* $Weekday$ **then**
        $REJECT$ traffic to IP
    else
      $ALLOW$ traffic

7.3 Summary

In order to understand the effectiveness of the P2P-aware classifier, it was important to compare it with known existing solutions. Snort IDS was chosen for this comparison since Snort is completely open-source and it has been the *de facto* IDS for academic research for several years. Moreover, many other solutions employed by past research are not open-source or freely available.

Snort failed to detect P2P traffic in the following censorious are tested on Information Security Lab., BITS-Pilani Hyderabad Campus. 1. It doesnt work at flow based features. 2. It also failed to detect the truncated payloads. Snort can only parse, packet heads, payload signatures and also behavioural aspects are implemented 3. Snort need to have signatures are stored in the system. It has a low detection rate when new type of P2P application is designed. 4. Snort can also outperform with more CPU cores, but with all the above draw backs it cant detect the P2P traffic.

Chapter 8

Conclusion and Future scope of work

This thesis proposed novel mechanisms for P2P traffic detection. Chapter 2 described the past research work on detection of P2P traffic approaches. A significant portion of this thesis dealt with the detection of P2P traffic using machine learning techniques proposed in Chapter 4. A portion of the thesis also proposed behavior based and Fuzzy pattern based approaches to overcome the drawbacks with ML techniques to detect P2P traffic in real-time, which is discussed in Chapter 5 and 6.

8.1 Conclusion

In Chapters 4, 5 and 6, we proposed our novel approaches for the detection of P2P traffic from the regular web traffic. Our approaches don't rely on signatures or Deep Packet Inspection (DPI) mechanisms which are bypassed by P2P

applications using encryption.

In Chapter 4, we presented *multi classifiers* approach. This approach differentiates between P2P and NonP2P traffic with high accuracy. P2P traffic identification modules were developed by extracting features from network flows. To identify P2P traffic, we employed statistical characteristics of TCP and UDP protocols. We constructed 5-tuple (src ip, src port, dst ip, dst port and protocol) flows from these two protocols and then extracted statistical flow features from the network traces. We used different feature selection techniques to reduce the number of features required to train the model i.e., reduce the model build time to achieve optimized training model. An ensemble learning model was used to build detection models with supervised machine learning algorithms. Experiments were carried out on four diverse classifiers, NB, BN and C4.5 as base classifiers followed by RF as the meta-classifier. Stacking with full feature set and the detection rate, the accuracy obtained is 99.94% and 99.9% respectively. Using Voting approach with full feature set accuracy and detection rate, obtained results were found to be 99.7% and 99.7% respectively.

Another approach was proposed for P2P traffic identification which performs host-based detection using heuristics proposed in Chapter 5. This approach can evaluate the behavior of source and destination for both P2P and NonP2P traffic. The objective of this approach was to improve the P2P traffic identification accuracy based on the application profile of the host. This technique observes the communication patterns of an end-host, and it doesn't require any learning methods as flow-based approaches do. Six heuristics were proposed which could identify P2P traffic based on host behavior. Heuristics could identify a P2P host with high accuracy and low false positive rate. Heuristics were formulated to contain

some well-known port number information as well. Tunable thresholds were derived that provide trigger to application of the heuristics used. The P2P traffic has been classified from the proposed heuristics and the false positives (FP) were identified. According to experimental results, we obtain the average detection accuracy of more than 96.55% and the false positive rate of 2.5%.

In Chapter 6, we proposed a fuzzy pattern recognition system to detect P2P traffic aiming to accurately detect the traffic. In our study we found most of the P2P applications use both UDP and TCP transport protocols over the network. We observed that decentralized P2P hosts like Freenet, eMule never use any DNS queries over UDP protocol to discover the peers, on the other-hand hosts running torrents initiate DNS queries to find the domain of the tracker server. Whereas NonP2P hosts, e.g, hosts running http and https services resolve the domain name of the server. With these observations we extracted the following behavioral metrics to characterize P2P and NonP2P traffic. Our feature extraction module parses only IP, TCP and UDP headers and discards rest of the packets. Once these headers are checked for correctness then the module accumulates all the packets associated with every host. Thereafter the extraction module extracts the behavioral features for every host. Our approach uses maximum membership function principle to characterize P2P behavior. According to this principle, from a given feature set, if the membership function for P2P traffic derives high values, the host is detected as P2P. Similarly, if the membership function for NonP2P derives higher values, the host is detected as NonP2P. We defined several membership functions, two for UDP and two for TCP: (1) Normalized total number of sent and received packets, (2) Normalized average of the total payload bytes sent. The proposed approach achieved low false positive rate of about 1.11% for dataset 1, low false positive rate of about 2.49% for dataset 2 and low false positive rate

of about 0.04% for dataset 3. Among the total 11 false positives in the dataset 2, we found that those are HTTP instances which made several connections to tracker servers.

8.2 Future Scope of the work

Research on Internet traffic classification has produced creative and novel approaches, but the landscape is foggy, fragmented, and inconsistent. We outline the research directions that could improve the effectiveness of the traffic classification system.

- The increasing speed of network links requires thorough investigation of scalability trade-offs in traffic classification.
- Appropriate novel approaches for highly parallel low-cost architectures promise significant scalability improvements.
- Traffic classification techniques and algorithms to be presented with rigorous empirically grounded analysis of efficiency and performance.

We plan to explore further above aspects of traffic classification with respect to P2P traffic.

Bibliography

[1] M. Zhang, W. John, K. Claffy, and N. Brownlee, "State of the art in traffic classification: A research review," in *PAM Student Workshop*, 2009, pp. 3–4.

[2] "Ipoque internet study 2008/2009," http://www.ipoque.com/en/resources/internet-studies, accessed on 4 January 2014.

[3] H. Schulze and K. Mochalski, "Internet study 2008/2009," 2009.

[4] "Bittorrent," http://www.bittorrent.com/, accessed on 17 December 2013.

[5] "Sandvine: Global internet phenomena report on asia-pacific and europe 2015," https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-apac-and-europe.pdf/, accessed on 31 January 2016.

[6] "Napster," http://www.napster.com/.

[7] "Gnutella," http://rfc-gnutella.sourceforge.net/.

[8] "Kazaa," http://www.kazaa.com/.

[9] "edonkey," http://www.edonkey.com/.

[10] "Freenet," http://www.freeenet.com/.

[11] "Skype," https://www.skype.com/en/.

[12] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *acmcs*, vol. 36, no. 4, pp. 335–371, dec 2004. [Online]. Available: http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html

[13] "Directconnect++," http://dcplusplus.sourceforge.net/, accessed on 10 August 2013.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*. ACM, 2001, vol. 31, no. 4.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350.

[17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[18] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Peer-to-Peer Systems*. Springer, 2002, pp. 261–269.

[19] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science, vol. 3485. Springer, 2005.

[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Electronic Imaging 2002*. International Society for Optics and Photonics, 2001, pp. 156–170.

[21] J. Buford, H. Yu, and E. K. Lua, *P2P Networking and Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[22] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Peer-to-Peer Systems IV*. Springer, 2005, pp. 205–216.

[23] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 46–66.

[24] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.

[25] R. A. Wilson and F. C. Keil, *The MIT encyclopedia of the cognitive sciences*. MIT press, 2001.

[26] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[27] P. Van Der Putten and M. Van Someren, "A bias-variance analysis of a real world learning problem: The coil challenge 2000," *Machine Learning*, vol. 57, no. 1-2, pp. 177–195, 2004.

[28] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[29] M. Borda, *Fundamentals in information theory and coding*. Springer Science & Business Media, 2011.

[30] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial intelligence*, vol. 151, no. 1, pp. 155–176, 2003.

[31] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[32] P.-N. Tan *et al.*, *Introduction to data mining*. Pearson Education India, 2007.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[34] L. A. Zadeh, "Information and control," *Fuzzy sets*, vol. 8, no. 3, pp. 338–353, 1965.

[35] P. Hajek, "Fuzzy logic," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., 2010.

[36] "Iana - internet assigned numbers authority port numbers," http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[37] A. Madhukar and C. Williamson, "A longitudinal study of p2p traffic classification," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on.* IEEE, 2006, pp. 179–188.

[38] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 314–329.

[39] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 512–521.

[40] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE.* IEEE, 2008, pp. 160–167.

[41] L. Gheorghe, *Designing and Implementing Linux Firewalls with QoS using netfilter, iproute2, NAT and l7-filter.* Packt Publishing Ltd, 2006.

[42] "Cisco's network-based application recognition," http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/network-based-application-recognition-nbar/prod_case_study09186a00800ad0ca.pdf.

[43] S. Sorensen, "Competitive overview of statistical anomaly detection," *White Paper, Juniper Networks*, 2004.

[44] "Qosmos: Deep packet inspection and metadata engine," http://www.qosmos.com/products/deep-packet-inspection-engine/, 2012.

[45] "ndpi: Open and extensible lgplv3 deep packet inspection library," http://www.ntop.org/products/deep-packet-inspection/ndpi/.

[46] O. Beaudoux and M. Beaudouin-Lafon, "Opendpi: A toolkit for developing document-centered environments," in *Enterprise Information Systems VII.* Springer, 2006, pp. 231–239.

[47] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *LISA*, vol. 99, no. 1, 1999, pp. 229–238.

[48] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, "Transport layer identification of p2p traffic," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. ACM, 2004, pp. 121–134.

[49] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 12, no. 2, pp. 219–232, 2004.

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

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

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

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

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

[55] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008.

[56] J. Li, S. Zhang, Y. Lu, and J. Yan, "Real-time p2p traffic identification," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008.* USA: IEEE, 2008, pp. 1–5.

[57] Y. Zhang, H. Wang, and S. Cheng, "A method for real-time peer-to-peer traffic classification based on c4. 5," in *Communication Technology (ICCT), 2010 12th IEEE International Conference on*. IEEE, 2010, pp. 1192–1195.

[58] H. Xu, S. Wang, R. Wang, and D. Zhao, "Research of p2p traffic identification based on naive bayes and decision tables combination algorithm," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 6. IEEE, 2010, pp. 2875–2879.

[59] D. Arndt, "How to calculating flow statistics using netmate," 2011.

[60] J.-j. ZHAO, X.-h. HUANG, S. Qiong, and M. Yan, "Real-time feature selection in traffic classification," *The Journal of China Universities of Posts and Telecommunications*, vol. 15, pp. 68–72, 2008.

[61] H. Chen, X. Zhou, F. You, H. Xu, C. Wang, and Z. Ye, "A svm method for p2p traffic identification based on multiple traffic mode," *Journal of Networks*, vol. 5, no. 11, pp. 1381–1388, 2010.

[62] "Omnipeek: Network analyzer," 2013.

[63] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*. ACM, 2006, pp. 281–286.

[64] C. Gu and S. Zhuang, "A novel p2p traffic classification approach using back propagation neural network," in *Communication Technology (ICCT), 2010 12th IEEE International Conference on*. IEEE, 2010, pp. 52–55.

[65] H. H. Ang, V. Gopalkrishnan, S. C. Hoi, and W. K. Ng, "Adaptive ensemble classification in p2p networks," in *Database Systems for Advanced Applications*. Springer, 2010, pp. 34–48.

[66] S. Dong, D. Zhou, and W. Ding, "Traffic classification model based on integration of multiple classifiers," *Journal of Computational Information Systems*, vol. 8, no. 24, pp. 10 429–10 437, 2012.

[67] R. Wang, L. Shi, and B. Jennings, "Ensemble classifier for traffic in presence of changing distributions," in *Computers and Communications (ISCC), 2013 IEEE Symposium on*. IEEE, 2013, pp. 000 629–000 635.

[68] L. Shi, R. Wang, and B. Jennings, "Training traffic classifiers with arbitrary packet sets," in *Communications Workshops (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1314–1318.

[69] D. Zhao, R. C. Wang, and H. Xu, "P2p traffic identification model based on ensemble learning," *Journal of Nanjing University of Posts and Telecommunications(Natural Science)*, 2011-04.

[70] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, P. Trimintzios, and M. Fukarakis, "CRAWDAD toolset tools/sanitize/generic/anontool (v. 2006-09-26)," Downloaded from http://crawdad.org/tools/sanitize/generic/AnonTool/20060926, Sep. 2006.

[71] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Computers & Security*, vol. 24, no. 1, pp. 31–43, 2005.

[72] J. Son, C. Irrechukwu, and P. Fitzgibbons, "A comparison of virtual lab solutions for online cyber security education," *Communications of the IIMA*, vol. 12, no. 4, p. 81, 2012.

[73] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experience with deter: a testbed for security research," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*. IEEE, 2006, pp. 10–pp.

[74] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[75] "jnetpcap." [Online]. Available: http://jnetpcap.com/

[76] T. G. Dietterich, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[77] D. Thomas G, "Machine-learning research," *AI magazine*, vol. 18, no. 4, pp. 97–136, 1997.

[78] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classi-

fication," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[79] E. Rich and K. Knight, "Artificial intelligence," *McGraw-Hill, New*, 1991.

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

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

[82] "Wireshark - sniffing tool," http://wiki.wireshark.org/Tools/.

[83] "Lbnl/icsi enterprise tracing project (2005, jan.)," http://www.icir.org/enterprise-tracing/download.html.

[84] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194 – 208, 2014.

List of Publications

The following list of peer-reviewed publications have come out during the course of this project:

Peer-reviewed journals/Book Chapters:

1. Chittaranjan Hota, Pratik Narang and **Jagan Mohan Reddy**, "Unwanted Traffic Identification in Large-scale University Networks: A case study", Big Data Analytics: Methods and Applications, Springer, March 2016.

Peer-reviewed conferences:

1. **Jagan Mohan Reddy** and C. Hota, Information Systems Design and Intelligent Applications: Proceedings of Third International Conference INDIA 2016, Volume 2. New Delhi: Springer India, 2016, ch. "Attack Identification Framework for IoT Devices", pp. 505-513. [Online]. Available: http://dx.doi.org/10.1007/978-81-322-2752-6_49

2. **Jagan Mohan Reddy** and C. Hota and M. Rajarajan, "Behavior-based P2P traffic identification using fuzzy approach,"IEEE International Conference on Applied and Theoretical Computing and Communication Technologies (iCATccT), Oct 2015, IEEE-Xplore.

3. **Jagan Mohan Reddy** and C. Hota, "Heuristic-Based Real-Time P2P Traffic Identification,"The 2nd International Research Conference on Emerging Information Technology and Engineering Solutions (EITES 2015), Feb 2015, Pune, IEEE, CPS, pp. 38-43, ISBN:978-1-4799-1838-6, IEEE-Xplore.

4. **Jagan Mohan Reddy** and Chittaranjan Hota. 2013. "P2P traffic classification using ensemble learning". In Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop (I-CARE '13). ACM, New

York, NY, USA, , Article 14 , 4 pages.
doi=http://dx.doi.org/10.1145/2528228.2528243

5. Pratik Narang, **Jagan Mohan Reddy**, and Chittaranjan Hota. 2013. "Feature selection for detection of peer-to-peer botnet traffic." In Proceedings of the 6th ACM India Computing Convention (Compute '13). ACM, New York, NY, USA, , Article 16 , 9 pages.
DOI=http://dx.doi.org/10.1145/2522548.2523133

6. **Jagan Mohan Reddy**, Pratik Narang, and Chittaranjan Hota, Prafulla Kumar, "P2P traffic classification for Intrusion Detection Systems," Security and Privacy Symposium 2013, IIT Kanpur, Feb 28th – March 2nd, 2013 (Poster).

7. **Jagan Mohan Reddy**, A. Thakur, and Chittaranjan Hota, "Approaches for Measuring P2P Classification Efficiency for Intrusion Detection and Prevention Systems," First National Conference on Cyber Security, NCCS 2012, Defence Institute of Advanced Technology(DU), Pune, India, June 7-8, 2012.

Biographies

Brief Biography of the Candidate Jagan Mohan Reddy is a full-time PhD Scholar at Department of Computer Science and Information Systems in BITS Pilani, Hyderabad Campus. His research work has been funded by Tata Consultancy Services (TCS), India. He received his Master's in Computer Science from Jadavpur University, Kolkata in 2010. He did his B.Tech in Computer Science from VNR VJIET, Hyderabad in 2008. His research interests are in the area of Network Security and Big-data analytics using applied Machine Learning algorithms.

Brief Biography of the Supervisor Chittaranjan Hota is a Professor and Associate Dean (Admissions) at Birla Institute of Technology and Science-Pilani, Hyderabad, India. He is also responsible for managing the Information Processing Unit at BITS-Hyderabad that takes care of ICT needs of the entire institute. He was the founding Head of Dept. of Computer Science at BITS, Hyderabad. Prof. Hota did his PhD in Computer Science and Engineering from Birla Institute of Technology & Science, Pilani. He has been a visiting researcher and visiting professor at University of New South Wales, Sydney; University of Cagliari, Italy; Aalto University, Finland and City University, London over the past few years. His research work has been funded by University Grants Commission (UGC), New Delhi; Department of Electronics & Information Technology (DeitY), New Delhi; Tata Consultancy Services (TCS), India; and Progress Software, India. He has guided PhD students and currently guiding several in the areas of Internet of Things, Cyber security, and Big-data analytics He is recipient of Australian Vice Chancellors Committee award, recipient of Erasmus Mundus fellowship from

European commission, and recipient of Certificate of Excellence from Kris Ramachandran Faculty Excellence Award from BITS, Pilani. He has published extensively in peer-reviewed journals and conferences and has also edited LNCS volumes. He is a member of IEEE, ACM, CSI, IE, and ISTE.