P2P network classification
A both port and payload agnostic approach

P. J. Molijn
P2P NETWORK CLASSIFICATION
A BOTH PORT AND PAYLOAD AGNOSTIC APPROACH

by

P. J. Molijn

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University, faculty of Management, Science and Technology

Master Software Engineering
to be defended publicly on Tuesday September 9, 2014 at 15:00 PM.

Student number: 851437445
Course code: T75317
Thesis committee: Prof. dr. M. C. J. D. van Eekelen (chairman), Open University
Dr. ir. H. P. E. Vranken (supervisor), Open University

An electronic version of this thesis is available at http://dspace.ou.nl/.
This thesis is dedicated in loving memory of my late mother, who suddenly passed away on Jan. 8th 2014. Her guidance and encouragement have enabled me to fulfill my potential.

ELEONORE EUNICE MOLIJN
(1951-2014)
ACKNOWLEDGEMENTS

Writing this thesis would not have been possible without the support of several people whom I need to gratefully thank.

I wish to thank my supervisors prof. dr. Marko van Eekelen and dr. ir. Harald Vranken for providing me with the opportunity to write this thesis under their guidance. Words cannot describe my gratitude towards Marko and Harald. Despite their busy schedules, they reserved some moments of their valuable time, they provided me with excellent feedback and guided me into the world of academia. It has been a real pleasure working with both of you and I am looking forward to our next project together. To PhD or not to PhD? That is the question!

I would also like to thank dr. Anda Counotte-Portman for her valuable advices and assuring that there were no delays in study progress due to the Open University’s part. I am thankful to dr. Bastiaan Heeren for his availability and willingness to answer many of my questions during my study.

I would like to thank my parents. They introduced me to the wonderful world of computing by purchasing my first computer, the commodore 128. My father, Hedwig, explained complex math problems in such a way that it was a real joy to find the solution myself when I was a child. My late mother, Eleonore, who loved reading books, passed this characteristic on to me. I really enjoy reading and am thankful for that. Thanks, Mom and Dad, for believing in me and providing me with the opportunities to pursue higher education.

When my younger and only brother, Delano, achieved his BSc degree, he motivated me to pursue my Masters. Thank you for just being my brother and one of the first people of implicit or explicit motivation on my journey.

Finally, without the love and support from my wife Gracia, my son Qylan and daughter Kaylin, I would never have succeeded. Gracia, my utmost respect for keeping up with all the late nights / early mornings during my study, always there providing me with the necessary drinks and/or food to keep going, not to mention my grumpy moods you gracefully dealt with when there were some setbacks. Qylan, thanks for the time we spend together with the Martial Arts Training. Just an example of the good moments we experienced to take the mindset from research onto a different topic. Kaylin, the funny faces you can put on, combined with the most hilarious story telling. Thanks for these example moments as well. I thank you both, love you very much, am proud of you and I am proud to be your father. Although all three of you had to cope with my absence often, you seldom complained. Hopefully now, as the thesis is completed, we can spend more family time together.

P. J. Molijn
Lelystad, September 2014
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>Summary</td>
<td>viii</td>
</tr>
<tr>
<td>Samenvatting</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Network Traffic Classification</td>
<td>6</td>
</tr>
<tr>
<td>1.2.1 Approaches</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2 Methods</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Research Question</td>
<td>11</td>
</tr>
<tr>
<td>1.4 Research Method and Objective</td>
<td>12</td>
</tr>
<tr>
<td>1.5 Research Contribution</td>
<td>13</td>
</tr>
<tr>
<td>1.6 Deliverables</td>
<td>13</td>
</tr>
<tr>
<td>1.7 Thesis outline</td>
<td>13</td>
</tr>
<tr>
<td>2 Peer-to-Peer Systems</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Architectures</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 Degree of centralization</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2 Network structure</td>
<td>19</td>
</tr>
<tr>
<td>3 Traffic Classification</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Approaches</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Port</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2 Payload</td>
<td>24</td>
</tr>
<tr>
<td>3.2.3 Host behavior</td>
<td>24</td>
</tr>
<tr>
<td>3.2.4 Flow feature</td>
<td>25</td>
</tr>
<tr>
<td>3.3 Machine Learning</td>
<td>25</td>
</tr>
<tr>
<td>3.3.1 Algorithms</td>
<td>25</td>
</tr>
<tr>
<td>3.4 Performance Criteria</td>
<td>28</td>
</tr>
<tr>
<td>4 Proposed Framework</td>
<td>31</td>
</tr>
<tr>
<td>4.1 Background</td>
<td>31</td>
</tr>
<tr>
<td>4.2 System design</td>
<td>32</td>
</tr>
<tr>
<td>4.2.1 The dialogue</td>
<td>32</td>
</tr>
<tr>
<td>4.2.2 P2P traffic</td>
<td>32</td>
</tr>
<tr>
<td>4.2.3 Details</td>
<td>33</td>
</tr>
</tbody>
</table>
# Contents

## 5 Data Analysis
- 5.1 Data Collection ........................................ 36
- 5.2 Attribute Importance .................................... 37
- 5.3 Classifier metrics ....................................... 38

## 6 Conclusion
- 6.1 Limitations .............................................. 44
- 6.2 Related work ........................................... 44
- 6.3 Future work ............................................. 45
- 6.4 Reflection ................................................ 45
- 6.5 MIT-License .............................................. 45

## Bibliography
- Books .......................................................... 47
- Academic Articles .......................................... 47
- Technical Documentation ................................. 50

## Acronyms

## Glossary
**List of Figures**

1.1 Botnet communication topology ................................................. 3
1.2 Traffic Classification Approaches .............................................. 7
1.3 Traffic Classification Methods .................................................. 10
1.4 Conceptual Research Model ...................................................... 13

2.1 Centralized P2P ........................................................................ 18
2.2 Decentralized P2P ..................................................................... 19
2.3 Different degrees of centralization .............................................. 20

3.1 Classification approaches trends ................................................... 23

4.1 Framework overview ................................................................. 32
4.2 Dialogues per interval ................................................................. 35

5.1 Attribute importance on training dataset ........................................ 38
5.2 Algorithm accuracy ................................................................... 39
5.3 Classifier Performance per application ......................................... 42
5.4 Classifier Comparison ............................................................... 42
# List of Tables

1.1 Comparison of botnet communication topologies [Oll09] ........................................... 4

3.1 Side-by-Side Comparison of the Approaches for Traffic Classification ....................... 26

5.1 P2P traffic dataset summary ......................................................................................... 37
5.2 Boosted REPTree Performance Per P2P Application ................................................. 40
5.3 Boosted J48 Performance per P2P Application ......................................................... 41
5.4 REPTree Performance per P2P Application ............................................................. 41
5.5 J48 Performance Per P2P Application ........................................................................ 41
5.6 Classifier Performance Per Algorithm ....................................................................... 42
The popularity of Peer-to-Peer (P2P) applications, and consequently the P2P traffic on the internet, has increased in the last years. This increase in traffic usage of P2P applications is besides benign P2P applications also due to malicious P2P software such as P2P botnets. To cope with the increasing threats imposed by malicious P2P botnets, botnets should be combated actively. A first step is to detect which internet traffic originates from P2P botnets. In this research, a start has been made by looking at whether internet traffic can be classified as either P2P traffic or non-P2P traffic, yet regardless of whether it concerns benign or malicious traffic.

Classification of P2P traffic is challenging since traditional techniques, that mainly analyze port numbers or payload data, are becoming ineffective against applications that use random ports or encryption. This research proposes, based on literature study, Machine Learning (ML) as a method for P2P traffic classification, using the algorithms J48, REPTree and AdaBoost for analysis of statistical flow features, which are both port and payload agnostic.

The classifier is trained with a data set consisting of network traffic derived from four P2P applications, two P2P botnets and non-P2P traffic. Classifier metrics were obtained by utilizing test data sets, in such a way that each individual set is disjunct with all the other sets (including training set). The results of this quantitative empirical research show that the proposed method can achieve high accuracy, outperforming comparable existing approaches for classification of P2P traffic.

The data sets and some source codes used in the thesis will be made available to the research community to enable validation and extension of the work.

**Keywords:** P2P traffic, port agnostic, payload agnostic, classification, Machine learning
De populariteit van Peer-to-Peer (P2P) toepassingen, en daarmee ook het P2P verkeer op het internet, is in de laatste jaren sterk toegenomen. Deze toename is naast het gebruik van goedaardige P2P toepassingen ook te wijten aan kwaadaardige P2P toepassingen zoals P2P botnets. Om de toenemende bedreigingen van P2P botnets te pareren, is actieve bestrijding ervan noodzakelijk. Een eerste stap daarin is om te detecteren welk internetverkeer deel uitmaakt van P2P botnets. In dit onderzoek is daarmee een start gemaakt door te kijken of internetverkeer geclassificeerd kan worden als P2P verkeer en niet-P2P verkeer, nog ongeacht of dat goed- of kwaadaardig verkeer betreft.

Classificatie van P2P verkeer is uitdagend aangezien traditionele technieken, die hoofdzakelijk poortnummers of payload-informatie analyseren, ineffectief zijn tegen toepassingen die willekeurige poorten of encryptie gebruiken. In het onderzoek is, op basis van literatuuronderzoek, Machine Learning (ML) gebruikt als methode voor classificatie van P2P verkeer, waarbij de algoritmen J48, REPTree en AdaBoost gebruikt zijn voor analyse van statistische flow features die zowel poort- als payload agnostisch zijn.

Het classificatie mechanisme leert P2P gedrag van een data set die bestaat uit zowel goedaardig P2P-verkeer, kwaadaardig P2P-botnet verkeer en niet-P2P verkeer. De nauwkeurigheid van de classifier op de daadwerkelijke test data bepaalt hoe effectief er onderscheid kan worden gemaakt tussen P2P en niet-P2P verkeer. De performance metrieken van de classifier zijn allen gebaseerd op het gebruik van test data sets, waarbij elke individuele set disjunct is met de overige sets(inclusief de training set). Uit de resultaten van dit kwantitatief empirisch onderzoek is gebleken dat hiermee een hoge nauwkeurigheid kan worden bereikt, die vergelijkbare bestaande benaderingen voor classificatie van P2P verkeer overtreft.

De datasets en enkele broncodes die tijdens het onderzoek werden gebruikt zullen publiekelijk ter beschikking worden gesteld om bijvoorbeeld validatie of uitbreiding van dit werk mogelijk te maken.

**Trefwoord:** P2P traffic, port agnostic, payload agnostic, classification, Machine learning
INTRODUCTION

1.1. BACKGROUND

Nowadays we live in a world where the internet plays a central role. The use of the internet and its associated applications exposes a trend of increased usage to the point at which they have turned into a necessary part of our lives. In spite of the fact that internet and internet-based applications can be extremely helpful, the utilization of these applications represents various security challenges.

The primary security risk is brought upon us from vulnerabilities in software which is then utilized by malicious software. Malicious software is also known as malware. McGraw and Morrisett [MM00] define malicious code as “any code added, changed, or removed from a software system in order to intentionally cause harm or subvert the intended function of the system.” As the internet based applications matured, malware experienced a gigantic improvement as well. Improving its attack scope, way of spreading, methods to hide its presence and versatility to dismantle attempts to name a few.

The most common malware infrastructure nowadays is a botnet [GOH11; SB11]. A botnet is a network of compromised computers which are controlled by a (mostly malicious) user, who is also known as the attacker, Botmaster or Botherder [GOH11; SB11]. The compromised computers, also known as Bots, run malicious software that successfully integrates techniques used by other previously known malware types, such as rootkits, worms, viruses, Trojans, etc. [PGL11].

A specially crafted communication path between the network of compromised computers and the botmaster is what sets botnets apart from other malware. This specially deployed path is called the Command and Control (C&C) communication channel [GOH11; SB11; Ros+13; UAH10]. Once the client’s machine is compromised, the C&C channel is used to send information from the bots to the server(s). The C&C channel provides a way for botmasters to have full control over the bots.

Common known malicious actions executed by bots are malware distribution, sending spam mails, commencing Distributed Denial of Service (DDoS) attacks, illegal content distribution, click fraud, collecting of private information (e.g. banking) and attacks on other critical infrastructure [GOH11; SB11; PGL11; SV13; Sil+13].

It has been observed that some botnets have a centralized architecture by connecting to a central C&C server. In this architecture, the computer or device acting as the C&C server
is the weakest point of the botnet as this exposes a single point of failure for the entire botnet [GOH11; SB11; Ros+13; UAH10]. To avoid the single point of failure of the centralized architecture, botmasters are also exploiting Peer-to-Peer (P2P) architectures. In a Peer-to-Peer (P2P) architecture there is no dedicated server or client role, as a P2P node can act as both a server and a client, thereby eliminating the centralized C&C channel, making P2P botnets an attractive alternative architecture for botmasters [Zei+10; Liu+10; Saa+11; Ros+13]. In between these two extremes of a single centralized C&C server towards no dedicated server, other other variants are possible [Oll09], leading to the following topologies (see Figure 1.1):

**Centralized.** The centralized topology relies upon a single central C&C server to communicate with all the bots. Each bot gets its instructions directly from the central C&C server.

**Multi-Server.** Multi-server is an extension of the centralized C&C topology, in which multiple servers are used to provide C&C instructions to the bots.

**Hierarchical.** A Hierarchical topology allows bots to propagate C&C instructions to other underlaying bots, effectively creating a hierarchical layer of command.

**P2P.** Botnets with a P2P architecture, do not have a centralized C&C server. Instead, commands are injected into the botnet via any bot.

Table 1.1 provides a quick overview with the pros and cons of the common botnet communication topologies [Oll09].

Studies of global internet traffic have shown that P2P applications were producing more traffic than all the other applications together, being responsible for 49% to 83%, on average, of all internet traffic and reaching peaks of over 95% [Gom+13]. It should be noted that not all P2P traffic is malicious, there are numerous legitimate P2P applications (Voice over IP (VoIP), videoconferencing e.g.).

A Peer-to-Peer (P2P) computer network is a distributed architecture where tasks or workloads are divided amongst different computers. Every computer in this distributed network is referred to as a *peer* or *node*. In P2P networks, clients provide resources, such as bandwidth, storage space, and computing power for example [Wan+14]. Also specific for a P2P network is its resilience capability. When a peer is either intentionally or unintentionally disconnected from the network, the P2P application will still continue to function by using other peers [Wan+14].

Despite its advantages, P2P networks introduce some problems of their own. Resource discovery introduces overhead costs [Wan+14]. When a peer \( P \) needs resource \( R \) (e.g. a file, bandwidth, computing power), peer \( P \) sends out what is called a *query message* describing \( R \). A list \( L \) is returned from the resource discovery system containing the nodes which can provide \( R \). System performance is decreased by the queries and broadcasts sent, while not necessarily resulting in improvement in resource quality. As a result of increased overhead, purely decentralized P2P-based systems scale poorly. Additionally, P2P-based systems also can be dominated by freeloaders that only consume resources, but do not contribute to the system as a whole. These peers add to the system overhead, but fail to contribute to other peers.

In addition decentralized networks introduce new security issues because they are designed so that each user is responsible for controlling their own data and resources [Wan+14].
1.1. BACKGROUND

(a) Centralized botnet topology.

(b) Multi-Server botnet topology.

(c) Hierarchical botnet topology.

(d) P2P botnet topology.

Figure 1.1: Botnet communication topology.
<table>
<thead>
<tr>
<th>Topology</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized</strong></td>
<td><strong>Speed of Control</strong></td>
<td><strong>Single point of failure</strong></td>
</tr>
<tr>
<td></td>
<td>The direct communication between the C&amp;C and the bots means that instructions (and stolen data) can be transferred rapidly.</td>
<td>If the central C&amp;C is blocked or otherwise disabled, the botnet is effectively neutralized.</td>
</tr>
<tr>
<td><strong>Multi-Server</strong></td>
<td><strong>No single point of failure.</strong></td>
<td><strong>Requires advance planning</strong></td>
</tr>
<tr>
<td></td>
<td>Should any single C&amp;C server be disabled, the botmaster can still maintain control over other bots.</td>
<td>Additional preparation effort is required to construct a multi-sever C&amp;C infrastructure.</td>
</tr>
<tr>
<td></td>
<td><strong>Geographical optimization</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multiple geographically distributed C&amp;C severs can speed up communications between botnet elements.</td>
<td></td>
</tr>
<tr>
<td><strong>Hierarchical</strong></td>
<td><strong>Botnet awareness</strong></td>
<td><strong>Command latency</strong></td>
</tr>
<tr>
<td></td>
<td>Interception or hijacking of bots will not enumerate all members of the botnet and is unlikely to reveal the C&amp;C server.</td>
<td>Because commands must traverse multiple communication branches within the botnet, there can be a high degree of latency with updated instructions being received by bots. This delay makes some forms of botnet attack and malicious operation difficult.</td>
</tr>
<tr>
<td></td>
<td><strong>Ease of re-sale</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A botmaster can easily carve off sections of their botnet for lease or resale to other operators.</td>
<td></td>
</tr>
<tr>
<td><strong>P2P</strong></td>
<td><strong>Highly resilient</strong></td>
<td><strong>Command latency</strong></td>
</tr>
<tr>
<td></td>
<td>Lack of a centralized C&amp;C infrastructure and the many-to-many communication links between bots make it very resilient to shutdown.</td>
<td>The ad hoc nature of links between bots make C&amp;C communication unpredictable, which can result in high levels of latency for some clusters of bots.</td>
</tr>
<tr>
<td></td>
<td><strong>Botnet enumeration</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Passive monitoring of communications from a single bot-compromised host can enumerate other members of the botnet.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: Comparison of botnet communication topologies [Oll09]
Since no central server monitors and corrects badly behaving peers, the peers can provide poor quality data or even unwanted data to other peers and get away.

The need for accurate P2P traffic classification is not only necessary to address the management and security problems given above, but the trend that botmasters are reaching out for P2P architectures to distribute their malware, also stresses out how important P2P traffic classification is [Zei+10; Liu+10; Ros+13; Saa+11; Wan+14]. With P2P traffic classification, unusual flows can be detected early to help find P2P malware [RH13].

Historically, network traffic was easily identified by matching port numbers of that traffic, with a list of officially assigned port numbers maintained by the Internet Assigned Numbers Authority (IANA)\(^1\) [Aut11]. Almost all applications nowadays can be reconfigured to use different port numbers, making this detection technique almost ineffective.

A more sophisticated approach is based on payload inspection or otherwise known as Deep Packet Inspection (DPI). This approach examines each packet and searches for some predefined application specific patterns, hence achieving a higher accuracy than port based matching. If an application communicates on non standard port numbers, DPI might still be able to detect it, assuming the payload is not encrypted.

The next approach is flow analysis. A flow is summarized data identified by a 5-tuple consisting of source IP Address, destination IP address, source port number, destination port number and protocol of the network or transport layer. Flow analysis can be done in two ways [Kor12; KPF05]:

- host behaviour
- flow feature

With host behaviour, it is assumed that many applications or groups of applications have a specific behavioral pattern when running on a host. The classification consists of matching previous patterns with the pattern from the behaviour of the host under investigation [KPF05]. This approach focuses on finding the set of hosts \(H\) that are running application \(a\) or group of applications \(A\).

With flow feature, features are computed over multiple packets grouped in flows and further used in the training process that associates sets of features with known traffic classes. The classification consists of a statistical comparison of unknown traffic with previously learned rules [Kor12].

Both host behaviour and flow feature forms of the flow analysis approach may include data mining techniques and Machine Learning Algorithms (MLAs). A MLA can divide the communication into clusters or groups where each group contains one dominant protocol.

The main drawback of behavioral and flow feature analyses is that, mostly, they produce estimates instead of accurate results [PN11]. The consequence, is that behavioral and flow feature analysis sparingly achieve 100% accuracy [PN11; DD11].

To cope with the increasing threats imposed by malicious P2P botnets, botnets should be combated actively. A first step is to detect which internet traffic originates from P2P botnets. In this research, a start has been made by looking at whether internet traffic can be classified as either P2P traffic or non-P2P traffic, yet regardless of whether it concerns

\(^1\)http://www.iana.org
benign or malicious traffic. Classification of P2P traffic is challenging since traditional techniques, that mainly analyze port numbers or payload data, are becoming ineffective against applications that use random ports or encryption. This research proposes, based on literature study, Machine Learning (ML) as a method for P2P traffic classification, using the MLAs J48, REPTree and AdaBoost for analysis of statistical flow features, which are both port and payload agnostic.

1.2. Network Traffic Classification

In this section, the current approaches and methods for protocol classification are described. Network traffic classification is according to Korczynski [Kor12]: “Methods of classifying traffic data sets based on features passively observed in the internet traffic according to classification goals.”

1.2.1. Approaches

This section describes the classical approaches used for protocol classification. The four existing techniques in network traffic classification are divided into two content based approaches and two flow analysis based approaches. The two content based traffic classification approaches are:

- Port based approach
- Payload based approach

The two flow analysis based traffic classification approaches are:

- Host behaviour based approach
- Flow feature based approach

Figure 1.2 provides a visual representation of network traffic classification approaches.

Port Based

Port number based approaches, were the first techniques to detect P2P traffic [Wan+14]. This type of classification is the oldest one, mostly due to its ease of use when collecting and analysing network data [PN11]. In the early days of the internet, most applications were assigned a specific port number. The detection is done by capturing TCP [Pos03] or UDP [Pos80] packet headers, and comparing the port numbers with the official list of port numbers maintained by IANA[Aut11]. However, more and more P2P applications do not use standard ports anymore to circumvent detection [Wan+14; PN11]. This situation causes that port based protocol detection effectiveness is deteriorating downwards to an ample 30% and less of internet traffic [MP05; SSW04; MW06].

Payload Based

The port based approach examines the packet header only, while payload based detection takes a look at the complete packet. Payload analysis is also known as Deep Packet Inspection (DPI). The packet payload has more data for a detection technique to utilize. DPI methods have a high degree of exactness and are not dependent on port numbers. An essential part in DPI detection is the existence of a pattern or signature database. By
way of comparison, it has the same meaning as the list of well known port numbers of IANA [Aut11] as most protocols have some identifying byte string patterns which are unique for them. The payload detection mechanism relies heavily on a properly maintained pattern database, as new protocols are invented or when there are significant differences between versions of one protocol. In 2004, Sen et al. [SSW04] described the accuracy, feasibility and robustness of signature based P2P traffic detection. Their experiments achieved accuracy from 90% to 100% depending on the protocol. This showed that P2P protocols could be detected by deep packet inspection in high-speed networks at that time [PN11].

A drawback of using DPI techniques is that it requires a lot of computational resources, cannot cope with encrypted data and does not detect new P2P applications with unknown characteristics (not in the pattern database) and have high maintenance costs (keeping the pattern database up to date).

**HOST-BEHAVIOUR BASED**

Flow analysis based approaches can address some limitations of content based approaches [KPF05; Kor12]. The first flow analysis based approach, the host behaviour approach, focuses on the analysis of the behaviour of network hosts, allowing observations of encrypted payload data. Examining network traffic patterns of hosts is the approach taken by behavioral analysis [XZB05]. The patterns can be for example number of incoming / outgoing host connections, number of different ports used, number of transferred bytes, number of received bytes, etc. As this does not require DPI, this technique can be applied for all types of networks. However, behavioral analysis is able to divide protocols into classes but, unlike the content based approaches, it is not capable to determine among applications in the same class. The classes defined by Moore [MZ05] are:

- Bulk
- Database
• Interactive
• Mail
• Services
• WWW
• P2P
• Attack
• Games
• Multimedia

For the scope of this research the class P2P as defined by [MZ05] is used as P2P, all other classes as non-P2P.

Since the information suitable for protocol detection is gathered from statistics about connections, there is no need to find patterns in packet payload. Behavioral analysis will sparingly reach 100% accuracy in its classification. As it was mentioned earlier, there is no strict identification of protocols like a byte string inside a packet.

Although a host-behaviour based approach sounds interesting enough for P2P classification, the limiting scope of only looking at a single host for analysis is seen as too limiting for this research. A host behaviour approach would allow for the detection of current P2P applications (regardless of intent, benign or malicious), only if the behaviour of these applications is known a-priori or created before the attempt to classify traffic for a host under investigation. The next approach, flow-feature based, attempts to overcome this limitation.

**Flow-feature based**

Another approach for network traffic analysis is to extend the behavioral characteristics in a flow to go past the reach of a single host perspective. The main difference with host behaviour and flow-feature is that with host-behaviour the behavioral signature of each single host needs to be known beforehand. The host under investigation can then be compared with this signature. Flow-feature analysis is first given a set of known traffic classes and attempts to deduct rules from the training data, such that given a different set of data it can associate it with a specific traffic class. Features are computed over multiple packets grouped in flows and further used in a training process that associates sets of features with known traffic classes. The classification consists of a statistical comparison of unknown traffic with previously learned rules [Kor12]. Features could contain the set used by host behavioral analysis as well as e.g. number of packets, flow length, inter-arrival times, inter-packet gaps, etc [PN11].

Since this method analyzes aggregated data and does not inspect packet payload, this approach is suitable for high-speed networks. In line with host behaviour analysis, flow analysis is not likely to produce 100% accuracy since this approach looks for patterns in flow behaviour that may vary between measurements [PN11].

Moore et al. [MZC05] defined 248 differentiators for flow characterisation. They consist of statistics about packet sizes, inter-packet timing and information derived from the transport protocol (TCP), such as SYN and ACK counts [PN11]. Some statistics are collected
directly by counting packets or packet sizes. Port numbers, size of TCP segments and other are derived from packet headers [PN11].

In a follow up study, Li and Moore [LM07] showed that from these 248 attributes, 11 would be sufficient for accurate protocol detection.

These attributes are:

1. server port
2. client port
3. count of all packets with push bit set in TCP header (server → client)
4. count of all packets with push bit set in TCP header (client → server)
5. count of packets with at least 1 byte of TCP data payload (server → client)
6. the total number of bytes sent in initial window (server ↔ client)
7. variance of total bytes (client → server)
8. median of total bytes (client → server)
9. average packet size: data bytes divided by packets count (client → server)
10. minimum packet size observed (server → client)
11. total of Round-Trip Time (RTT) (server ↔ client).

RTT is the time required for packet to travel from client to server and back again.

P2P applications are not bound to specific ports, therefore the server and client port are not used as an attribute for detection in this research. P2P applications are shifting from TCP traffic only to hybrid TCP / UDP and even UDP traffic only applications. The result, for this research, is the omission of attributes regarding TCP header only.

Narang et al. [NRH13] found that the attributes for P2P protocol detection should incorporate both TCP and UDP traffic and should not rely on ports.

The features used in this research are based on the work of Narang et al. and are:

- minimum packet size (client → server)
- maximum packet size (client → server)
- maximum packet size (server → client)
- average packet size (client → server)
- average packet size (server → client)
- maximum inter-arrival time (client → server)
- flow duration (client ↔ server)
- volume bytes (client → server)

\(^2\) the push bit forces immediate delivery of data
1.2.2. Methods
As the more recent flow analysis approach for traffic classification relied on MLAs, the content-based approaches mainly relied on simple pattern matching [Kor12]. In this section, the Pattern Matching and Machine Learning methods (see Figure 1.3 [Kor12]) used in classification approaches as described in the previous section are discussed.

Pattern Matching
Previously, simple pattern matching combined with content-based approaches was one of the most accurate classification methods. However, in the case of encrypted traffic, pattern matching based on identifying the application level signatures is less effective (if possible). Sen et al. [SSW04] provided an efficient method for identifying five popular p2p applications through application level signatures. All of the proposed signatures, however, become useless once traffic encryption or tunneling methods are applied [Kor12].

Machine Learning
This section provides a general idea of machine learning.

Machine Learning (ML) is a collection of techniques for data mining and knowledge discovery which searches for useful structural patterns in data [PN11]. ML techniques can be divided into two groups according to types of learning [PN11]:

Supervised learning uses training data, from which classification rules are extracted to classify unseen examples.

Unsupervised learning does not rely on training data and groups instances that have similar characteristics into clusters.
With supervised learning, all training data is labelled to contain the target value. To predict a numerical value for a given set of input values is called Regression. Regression models can answer a question with a numerical answer. Classification is when each entry in the dataset is assigned a specific class, with the goal of determining the class value from new data.

Unsupervised learning methods do not have a labelled training set and are used for grouping data that expose similar characteristics into clusters or estimating densities.

This research proposes, based on literature study, ML as a method for P2P traffic classification, using MLAs for analysis of statistical flow features, which are both port and payload agnostic.

1.3. Research Question

To cope with the ever increasing threats imposed by P2P botnets, innovative detection and elimination techniques are needed. P2P botnet mitigation can start by taking the first step: detecting the existence of P2P Traffic. Detection is one of the most important elimination techniques as it offers an initial indication of the existence of compromise. Malware detection is, in fact, the main prerequisite of all other neutralization actions[SP14].

Using MLA to identify malware on both network and client levels is sketched by Masud et al. [MKT11] and Dua and Du [DD11]. In [MKT11; DD11] the general role of machine learning in relation with cyber security is studied.

The research proposal focuses on determining the distinguishing network traffic characteristics to identify P2P traffic, preferably using MLA techniques.

The research question is narrowed down to:

**How do we classify network traffic into P2P and non-P2P traffic?**

To answer this question, we would need to answer the following sub questions:

- **RQ1: Which algorithm is suitable for P2P traffic classification?**
  Armed with a dataset of captured network traffic containing P2P and non-P2P packets, it is assumed that a pattern can be learned from this set to predict future packets.

- **RQ2: What relevant features are needed for P2P traffic classification?**
  Attributes are needed for the algorithm found in RQ1. These attributes are mapped onto the network dataset as characteristics of the network communication between two hosts. Features relevant for P2P communication need to be extracted from the dataset.

- **RQ3: How do we apply a port agnostic, payload agnostic classification technique into a P2P traffic classification approach?**
  Since P2P systems, whether benign or malicious, are using dynamic port numbers and encrypting their payload more often, a P2P classification technique which is port agnostic and payload agnostic can be more effective.

- **RQ4: How effective is this P2P traffic classification approach?**
  Using classification metrics, this classifier is compared with other P2P classifiers.
1.4. **Research Method and Objective**

The research is based on the empirical research method combined with experiments. With empirical research, knowledge is gained by observing or experience. The artifacts for empirical research are used for quantitative or qualitative analysis.

This research started with a literature study on the following areas:

- P2P Botnets
- P2P Architectures
- Network traffic classification
- Machine Learning Algorithms

First the general Botnet architecture is reviewed, zooming in on P2P Botnets in particular. This is done to get background information on Botnets in general as well as the more recent P2P botnet architectures. As Botnet systems evolve by using P2P architectures, the literature study extended with the study on P2P systems. Information to model a classifier was gained by studying network classification theories. To address the research subquestion about a suitable algorithm about P2P traffic classification, Machine Learning theories were studied.

To get a better understanding of the problem domain a conceptual framework for P2P traffic classification is proposed. The results of the literature study and the development of the conceptual framework lead to the implementation of a method for P2P traffic classification. This framework was tested and validated with at least 2 datasets. Figure 1.4 provides the conceptual research model for this research.

The research will utilize public available datasets or datasets provided by others with the consent of usage within this research. The following network traffic datasets have been obtained and used (either partially or fully):

- A dataset from Shiravi et al. [Shi+12].
  Dataset[^1], consists of labelled network traces, including full packet payloads that are publicly available to researchers. The main dataset of interest for this research is the dataset of general Non-P2P internet usage.

- A dataset from Rahbarinia et al. [Rah+14].
  This consists of two main datasets: a dataset of P2P traffic generated by a variety of P2P applications and a dataset of traffic from three modern P2P botnets.

The dataset by Rahbarinia et al. is used as it includes P2P traffic in both benign and malicious forms. The dataset by Shiravi et al. is used for the non-P2P traffic counterpart.

The research objective is to determine if P2P traffic can be distinguished from offline network trace files, hereby trying to improve classification accuracy of existing P2P classification mechanisms.

[^1]: [http://www.iscx.ca/datasets](http://www.iscx.ca/datasets)
1.5. RESEARCH CONTRIBUTION

The main contribution is a network classification method specifically aimed towards the classification of P2P traffic. The classification is therefore a generic P2P network traffic classifier, which segregates traffic into two bins: P2P and non-P2P Traffic. The question whether the P2P traffic is malicious or not is out of scope of this research, although the work done here could be extended for identifying malicious P2P traffic as well.

1.6. DELIVERABLES

The deliverables of the research project are the following:

- Tool(s) for flow feature extraction.
- Algorithm for P2P Traffic classification.
- Definition of relevant features for P2P traffic classification.
- A traffic classification approach not relying on port nor payload combined with flow analysis.

1.7. THESIS OUTLINE

This thesis is organized in the following manner:
Chapter 2 provides background information on P2P Systems.
Chapter 3 covers the basics on traffic classification, which is important to understand the methodologies proposed in this thesis.
Chapter 4 elaborates the framework for P2P traffic classification which can separate P2P traffic from Non-P2P traffic. Chapter 5 provides insights into the used datasets and analysis results of the framework. Chapter 6 concludes, presents related work and provides directions towards future work.
This chapter provides background information regarding P2P systems. The most important classification of P2P systems is their degree of centralization and their network structure. A brief description of each of the P2P classifications along with their advantages and disadvantages are described.

2.1. **INTRODUCTION**

Peer-to-Peer (P2P) is a distributed computer architecture that facilitates the direct exchange of information and services between individual nodes (called peers) rather than relying on a centralized server. P2P forms the basis of many distributed computer systems, permitting each peer node to act as both a client and a server, consuming services from other available peers while providing its own service to the rest of the network [Bas+13]. Peers within a P2P network communicate directly with their known neighbors, in order to submit requests and serve responses. The definition of what specifically constitutes a P2P system varies within the literature. Generally, in theory, a P2P system is envisioned as having no centralized authority, when in reality many existing P2P applications rely on one or multiple. Some versions of the BitTorrent protocol (a P2P protocol) for example required some kind of index also known as a “tracker” which is able to link the peers together and to perform management of the swarm (a swarm is a collection of peers that are interested in distributing the same content). The following definition by [Bas+13] is found to be well-suited for classifying P2P systems and is used within this research:

“Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage, and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.”

The characteristics of this definition are elaborated using the work of Rodrigues and Druschel [RD10] when they described the properties of a P2P system. **self-organize into network topologies**: Once a node is introduced into the system (typically by providing it with the IP address of a participating node and any necessary key material), little or no manual configuration is needed to maintain the system.
sharing of resources: Popular P2P systems have an abundance of resources that few organizations would be able to afford individually. The resources tend to be diverse in terms of their hardware and software architecture, network attachment, power supply, geographic location and jurisdiction.

capable of adapting to failures: P2P systems tend to be resilient to failures because there are few if any nodes that are critical to the system's operation. To attack or shut down a P2P system, an attacker must target a large proportion of the nodes simultaneously.

accommodating transient populations: The participating nodes are not owned and controlled by a single organization. In general, each node is owned and operated by an independent individual who voluntarily joins the system.

without requiring the intermediation or support of a global centralized server: The peers implement both client and server functionality and most of the system's state and tasks are dynamically allocated among the peers. There are few if any dedicated nodes with centralized state. As a result, the bulk of the computation, bandwidth, and storage needed to operate the system are contributed by participating nodes.

P2P offers many advantages. These include scalability, high resource availability, no need for a centralized authority (eliminating a single point of failure) and robustness [Bas+13]. With a P2P architecture however, the resources or services available is completely dependant on the participating nodes of the P2P system. Quality and usefulness are determined by the nodes themselves. The power of P2P is obvious when considering Metcalfe's Law, which states that the value of a network is proportional to the square of the number of connected users [Bas+13]. The number of possible connections within a P2P network can be exponential in relation to the number of network nodes, \( n \) [Bas+13]. All nodes can potentially connect to all other nodes, giving a theoretical maximum number of connections of \( n(n - 1)/2 \) the same number as in a fully connected mesh network [Bas+13].

P2P applications were primarily designed and used for large-scale file sharing, allowing participating users easy search facilities and the possibility to obtain or contribute content. This differs from the well known client-server model due to the fact that the files are provisioned in a distributed way and are replicated within the network when necessary. Since hosts participating in P2P networks also devote some computing resources, such systems scale with the number of hosts in terms of hardware, bandwidth and disc space.

Besides sharing data files, another interesting area for P2P applications is the sharing of computing resources. An example is grid computing, using the computing resources of a distributed P2P system can become a common way to solve large problems such as brute-forcing a strongly encrypted and encoded message. It should be mentioned that grid computing existed prior to P2P systems, but introducing P2P to grid computing allows additional flexibility and gives better scaling properties in regard to older grid computing techniques.

File-sharing systems are more popular amongst both benign and malicious P2P systems. The most common use of the P2P principle is multimedia file-sharing like photos, movies, music files, applications, including often illegal content. Malicious use of P2P file-sharing would be the distribution of worms, root-kits, viruses, bot-agents and others. P2P technology is also intensely used for providing communication services, like instant mes-
sengers (“presence”), chat, Internet and video conferencing. Popular applications are for instance Skype, WhatsApp, Lync, Google Talk, and many others.

This research is primarily concerned about P2P systems with a file sharing component and will exclude grid computing traffic because grid computers, normally, don’t have to hide. These systems use well known port numbers and don’t disguise traffic by using masqueraded or dynamic port numbers. Currently grid computing can be detected the same way as conventional traffic by inspecting port numbers.

The main characteristic of a P2P system is that it is not built around the server and client concept but on the cooperation of equal peers. This concept allows individual peers to join or leave the network resulting in the adaptive nature of P2P systems.

2.2. Architectures

P2P systems are categorized with respect to their degree of centralization and network structure.

2.2.1. Degree of Centralization

The decentralization makes it possible to utilise unused bandwidth, storage and processing power within a distributed network. It tends to lower the cost of system ownership and maintenance, increasing the scalability along the way.

There are mainly two different classifications for P2P systems regarding their degree of centralization [RD10; ABG13]:

- Centralized P2P
- Decentralized P2P

Centralized

Within a centralized P2P architecture, a number of centralized index servers maintain a database of the services on offer on the network. The clients are logging on to these index servers at anytime. The list with services is updated the moment a node joins or leaves the network, similar to a registration or deregistration process. An overview of this architecture operation is illustrated in Figure 2.1.

For a peer to get a wanted resource the first step is to submit a query for the resource (e.g. a file) to the centralized server(s). After receiving this request, the server(s) consult their services lookup list and respond back with a message about the peers who can serve the file. The peer then goes out to the serving peer directly to download the file. With this centralized structure, which is mostly classified as simple, the fired queries can be processed quickly, hence achieving relatively good performance. A drawback for a centralized approach, is analogous to the earlier mentioned central C&C server; when the main servers are identified, shutdown of these servers could then be achieved quickly.

Decentralized

A fully decentralized and distributed architecture is illustrated by Figure 2.2. Within fully decentralised architectures, all nodes are of equal importance, regardless of their capacities, resources / services offered or their geographic location. Without requiring the inter-mediation or support of a global centralized server or authority, all nodes perform the
same tasks, acting both as server and client. Hosts participating in such networks are called servants.

A fully decentralized architecture is not so popular nowadays because it is generally quite inefficient. Querying for resources works like a “broadcast” system. A node sends a query message to all its connected neighbors. As the response and request messages are relayed from a node to its neighbors, this way of resource discovery generates large traffic volumes. Messages may also have to cross a large number of hops before they reach a node which can provide the requested resource, increasing the total response time along the way. Within fully decentralised networks, it is also a challenge to provide Quality of Service (QoS) of any kind.

**Variants**

Besides the two P2P main models regarding degree of centralization, two other variants are worth mentioning namely:

- Hybrid decentralized P2P
- Partially centralized P2P

**Hybrid decentralized architectures**, a server cluster (2 or more servers) manages the collaboration between peers and can optionally provide an offered services lookup. The main difference with the centralized model is the addition of more servers in this model. It is observed that some P2P systems have servers that are numerous, geographically distributed and interconnected. The eDonkey system is an example of such an architecture.

**Partially centralized systems**, some nodes have more responsibility compared to others. These so called “supernodes”, can present the resources as local for all their connecting peers and providing connectivity with other supernodes. These supernodes are elected dynamically and so are their peers.
2.2. Architectures

Figure 2.2: Decentralized architecture for P2P Systems

Performance is better than the purely decentralized model and may be less than the hybrid decentralized model, but this model has more flexibility regarding fault-resiliency than the hybrid decentralized one.

An overview of the different degrees of centralization regarding P2P systems is illustrated in Figure 2.3

2.2.2. Network Structure

P2P systems build an overlay network, where each node maintains some point-to-point connections with some other nodes. With the regular activity of nodes joining and leaving the network, the topology of this overlay network is highly dynamic. This topology can be ad-hoc, totally unrelated to underlying physical links and host content, or it can be organised to use the network more efficiently or to locate content faster. When there is a form of association between node content and topology, the P2P system is said to be structured. P2P systems can be classified in three groups, with respect to their level of structure.

- Unstructured
- Structured
- Loosely structured

Unstructured

In unstructured networks, there is no relation whatsoever with the placement of data and the overlay topology. Not knowing where a given resource is, makes that searches are conducted at random, asking a number of servants if they have any resource that match the request. These servants may ask their own neighbours about the resources, which can lead to a request that accesses the entire P2P system (at some point every node is sent the request query). Although there are different possibilities for the construction of the overlay network
(a) Centralized model.

(b) Fully decentralised model.

(c) Partially centralised model.

(d) Hybrid Decentralised model.

Figure 2.3: Different degrees of centralization.
and for the query mechanism, unstructured networks generally result in poor lookup performance, scalability problems and inefficient network usage. However, this scheme is the most widely used since it accommodates easily a transient population and is well adapted to file-sharing. Users of such systems want some specific files and don’t want to store other files for the sake of system efficiency; they don’t want to be concerned with such issues as lookup performance (even if they prefer it fast) or redundancy (even if they want good availability). To solve performance and scalability issues in unstructured networks, the partially centralised or hybrid decentralised model can be used. Searches are still conducted at random but only at the supernode/server level. End users only send queries to their local supernode/server. This two-level structure improves performance and scalability, making the unstructured networks viable.

**Structured**

In structured systems, topology is closely related to hosts content or host content is related to topology. Resources (or index to resources) are stored at specific locations in the P2P system and a mechanism is provided to map a file identifier to its location (or the location of its pointer). Using a distributed routing table, generally a Distributed Hash Table (DHT), queries can be forwarded to an adequate host much more efficiently than in the unstructured case. The disadvantages of structured networks are the difficulty of maintaining the routing table with frequent arrivals and departures of peers and mapping a keyword query to a unique file identifier.

**Loosely structured**

Loosely structured networks are hybrid solutions between structured and unstructured networks. In such systems, a mapping exists between resource and topology, but it is not completely specified and may result in search failure (the search is then conducted as if the network was unstructured). Due to reduced implementation outside academic world, loosely structured networks are not elaborated further.
This chapter covers the basics on traffic classification, which is important to understand the methodologies proposed in this thesis.

A section is included to describe the most common metrics for the evaluation of the performance of a classification mechanism.

3.1. INTRODUCTION

Network traffic consists of IP packets which is examined for further analysis. In most network traffic analysis, the packet flows are uniquely identified by the 5-tuple: source IP address, source port, destination IP address, destination port, and transport layer protocol.

The traffic classification problem can be formalized as follows: A pattern \( p \) represents the object under analysis. Each pattern is described by a set of \( n \) features that have been derived from the analyzed traffic [Kor12]. Thus, it can be interpreted by the \( n \)-dimensional random variable \( X \) that corresponds to an accurate set of features: \( p \rightarrow x = (x_1, x_2, x_3, \ldots, x_n) \) [Kor12].

In the application classification problem, where \( p \) could be represented by flows, it is attempted to assign each of these flows to one of the given application classes \( c \) defined by a random variable \( Y : y = y_1, y_2, \ldots, y_c, y_{c+1} \). \( Y = y_{c+1} \) means that the analyzed flow is not recognized as any of the given classes, i.e., it is unknown [Kor12].

In the malware detection problem \( p \) could be represented by the aggregated traffic directed to the specific IP destination address [Kor12]. Thus, malware detection refers to a binary classification problem. The attempt is to verify if the traffic to analyze corresponds to malicious behavior. Random variable \( Y \) takes values in the set \( y_0, y_1 \), where \( Y = y_0 \) means that the traffic conforms to legitimate behavior, whereas \( Y = y_1 \) indicates malicious activity [Kor12].

In this thesis, the traffic classification problem corresponds to defining pattern \( p_0 \) for P2P and \( p_1 \) for non-P2P.

3.2. APPROACHES

As communication protocols evolve, the selection of an appropriate approach for traffic classification changes [Zha+09]. The variety of new Internet applications including services such as streaming, online gaming, p2p file sharing, or video/voice conferencing have
intensified research efforts to discriminate against such applications. These, in turn, have inspired sophisticated obfuscation mechanisms. Figure 3.1 gives the first view of trends in application development over time with respect to the four main classification approaches [Kor12].

![Figure 3.1: Trends in application development and classification approaches [Kor12]](image)

In the rest of this chapter, the four traffic classification approaches for protocol detection are described. The two content based approaches as well as the host behaviour approach are briefly described as they are not used within this research and the basics of these three approaches were touched upon in Chapter 1 (See Figure 1.2 for an overview).

### 3.2.1. PORT

More than a decade ago, network traffic was accurately classified using UDP and TCP port numbers [Aut11; Zha+09].

The classification of network traffic based on the TCP [Pos03] or UDP [Pos80] port numbers is a simple approach built upon the assumption that each application protocol always uses the same specific transport-layer port [Gom+13].

Nowadays, new internet applications are moving towards the use of dynamic ports to evade detection (Fig 3.1). An example of such an application is Skype, it puts big efforts into the establishment of connections amongst it peers, hereby trying to bypass restrictive firewalls, by randomly selecting ports and even trying to utilize port 80 or 443 when connection on dynamic ports do not succeed.

Thereby, port numbers as a classification mechanism are considered obsolete [Kar+04; MP05; MW06].

As a result, simple inspection of port numbers is no longer a reliable classification mechanism [MP05].
3.2.2. Payload

The second content-based approach involves inspecting the packet payload and for years, it was considered as the most accurate method. Deep Packet Inspection (DPI), extends the examination beyond the packet header only as is the case with port based methods. DPI inspects the complete packet payload. As soon as a unique payload-based signature is identified, this technique can produce reliable classification results [MP05; SSW04]. It was not uncommon that payload-based classifiers were often used to establish ground truth for other methods. DPI methods rely on a database of previously known signatures that are associated to application protocols, and search each packet for strings that match any of the signatures [Gom+13]. This approach is used not only in the classification of network traffic, but also in the identification of threats, malicious data, and other anomalies. Because of their effectiveness, classification systems based on DPI are especially significant for accounting solutions, charging mechanisms, or other purposes for which the accuracy is crucial.

However, deeply inspecting each packet can be a demanding task in terms of computation power and may be unfeasible in high-speed networks. Therefore, some mechanisms search only a part of each packet or only a few packets of each flow, as a compromise between efficiency and accuracy. Besides of the performance issues, the inspection of contents of the packet may also raise legal issues related with privacy protection [SOG07]. Nevertheless, the main drawback of DPI techniques is their inability to be used when the traffic is encrypted [Gom+13]. Since, in these cases, the contents of the packets are inaccessible (encrypted), DPI-based mechanisms are restricted to specific packets of the connection (e.g., when the session is established) or to the cases when UDP and TCP connections are used concurrently and only the TCP sessions are encrypted [Gom+13]. DPI methods are also sensitive to modifications in the protocol or to evolution of the application version: any changes in the signatures known by the classifier will most certainly prevent it from identifying the application [Gom+13]. Moreover, DPI methods that rely on signatures for specific applications can only identify traffic generated by those applications [Gom+13].

3.2.3. Host Behavior

Host behavior-based approaches can potentially address some limitations of content-based methods [KPF05]. The approach allows observing even encrypted payloads as the analysis is based on behavior of the hosts. More specifically, the communicating hosts are represented by Traffic Dispersion Graphs (TDGs) which visualize the behaviour. The classification consists of matching previously observed graphs with graphs resulting from the behavior of a host under examination [KPF05]. Karagiannis et al. for example, proposed an interesting method based on observing and recognizing models of host behavior and then classifying its flows according to the models [KPF05]. The following levels were analyzed:

**social level** The inspection of interactions with other hosts

**functional level** Checking whether a host acts as a client or server (or both) for serving resources

**application level** Recording the transport layer ports to identify the origin of the application.
Although promising, the host behavior classification in [KPF05] was still found to be limiting for the research, because of:

- The reliance on port statistics (not port numbers though). Applications hiding behind port masquerading schemes will slip through.

- The assumption at the functional level that hosts that use a single port for the majority of their interactions with other hosts are likely to be providers of the service offered on that port. As with the previous item, port masquerading could mislead the classifier.

### 3.2.4. Flow Feature

The second approach within the group of the flow analysis classification approaches uses flow features such as average packet sizes, packet inter-arrival times, or flow durations. Features are computed over multiple packets grouped in flows and further used in the training process that associates sets of features with known traffic classes [Kor12]. The classification consists of a statistical comparison of unknown traffic with previously learned rules [Kor12]. Flow feature-based approaches mainly include data mining techniques and machine learning algorithms. Machine learning and machine learning algorithms will be elaborated in the next section.

Moore and Zuev [MZ05] proposed a statistical approach to classify traffic into different types of services based on a combination of flow features such as flow length, time between consecutive flows, or inter-arrival times. The classification process using a bayesian classifier combined with a kernel density estimation method led to an accuracy of up to 95%.

Table 3.1 provides an easy overview of the main characteristics of each classification approach.

### 3.3. Machine Learning

Although there are two methods for the classification approaches as described in the previous section and briefly mentioned in Chapter 1, namely: Pattern Matching and Machine Learning, this thesis will focus primarily on the Machine Learning method of classification. This is done because the approach to predict the class (P2P or Non-P2P) of new unseen traffic with a labelled dataset as a starting point, closely matches the supervised machine learning category.

In general, machine learning algorithms are categorized into supervised learning and unsupervised learning. Supervised learning uses training data, from which classification rules are extracted to classify unseen examples. Unsupervised learning does not rely on training data and groups instances that have similar characteristics into clusters.

#### 3.3.1. Algorithms

The choice of Machine Learning Algorithm (MLA) is a critical step in building a statistics-based classifier. Narang et al. [NRH13] found out that the three most relevant algorithms for P2P detection are: J48, Naïve Bayes and REPTree. Early experiments with Naïve Bayes showed overall performance less than 60% of correct predictions, while the other two scored significantly higher. Gomes et al. [Gom+13] also state that within the P2P traffic classification domain, the most common used supervised learning approach are the tree structures
### Table 3.1: Side-by-Side Comparison of the Approaches for Traffic Classification

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Characteristics</th>
<th>Advantages</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port based</td>
<td>Associates port numbers with applications</td>
<td>- Low computational requirements</td>
<td>Lack of classification performance due to random port numbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Easy to implement</td>
<td></td>
</tr>
<tr>
<td>DPI based</td>
<td>Relies on payload data</td>
<td>High classification performance</td>
<td>- May not work for encrypted traffic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Requires high processing resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Can only be used for known applications</td>
</tr>
<tr>
<td>host behavior</td>
<td>Uses only packet header and searches for previously found host behavior patterns</td>
<td>- Usually lighter than DPI</td>
<td>- Usually has lower classification performance when compared to DPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Applicable for encrypted traffic</td>
<td></td>
</tr>
<tr>
<td>flow feature</td>
<td>uses only packet header and flow-level information</td>
<td>- Applicable for encrypted traffic</td>
<td>- Requires machine learning theory which could increase complexity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Can identify unknown applications from target classes</td>
<td></td>
</tr>
</tbody>
</table>
like J48 and REPTree. These findings led to the exclusion of Naïve Bayes in further experiments and limited the algorithm set to: J48 and REPTree.

In this research two different supervised learning approaches are used: J48 and REPTree. Both J48 and REPTree fall under Decision Tree (DT) classifiers. DT classifiers create a tree whereby each node is composed of a decision that can split the data into smaller sets using the labels from the supplied training set. Each node on the tree can be visualized as an if-then-else decision. The construction of a decision tree can be expressed recursively. First, select an attribute to place at the root node, and make one branch for each possible value [HWF11]. This splits up the example set into subsets, one for every value of the attribute [HWF11]. Now the process can be repeated recursively for each branch, using only those instances that actually reach the branch [HWF11]. If at any time all instances at a node have the same classification, stop developing that part of the tree [HWF11]. The only thing left is how to determine which attribute to split on, given a set of examples with different classes.

For the purpose of this research, Weka\(^1\) was used as the tool for handling machine learning algorithms. This tool provides several different machine learning algorithms and has cross-platform operability such as: Mac OS, Windows and Linux variants. Weka has support for J48 and Reduced Error Pruned Tree (REPTree) DTs which are used as classification tasks in the supervised learning setting.

**C4.5**

DT algorithms in Weka are implementations of the C4.5 algorithm. This algorithm creates binary trees in such a way that at each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

If the target attribute can take on \(c\) different values, then the entropy of \(S\) relative to this \(c\)-wise classification is defined as [Mit97]:

\[
Entropy(S) = \sum_{i=1}^{c} - f_i \log_2 f_i
\]

where \(f_i\) is the proportion of \(S\) belonging to class \(i\).

Given entropy as a measure of the impurity in a collection of training examples, the effectiveness of classifying an attribute in the training data can be measured [Mit97]. This measure is called the information gain, and is simply the expected reduction in entropy caused by partitioning the examples according to the attribute [Mit97]. More precisely, the information gain, \(Gain(S, A)\) of an attribute \(A\), relative to a collection of examples \(S\), is defined as:

\[
Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)
\]

where \(Values(A)\) is the set of all possible values of attribute \(A\), and \(S_v\) is the subset of \(S\) for which attribute \(A\) has value \(v\) (i.e., \(S_v = s \in S | A(s) = v\)) [Mit97].

\(^1\)http://www.cs.waikato.ac.nz/ml/weka/
3. Traffic Classification

J48
This algorithm uses a tree structure and is divided into several phases [PN11]. During the training process, every leaf can estimate the error ratio of the number of wrong classified incidents and the total incidents assigned to each leaf from the supervised training data sets [PN11]. The upper node can also calculate the weighted sum of error estimates for all its leaves [PN11]. If the weighted sum at the upper node is less than the error ratio combined from its leaves, all leaves under the node are pruned [PN11].

REPTree
The REPTree [PTK06] uses a fast pruning algorithm to increase the accurate detection rate with respect to noisy training data [PN11]. Pruning is used to find the best sub-tree of the initially grown tree with the minimum error for the test set [PN11]. However, the number of sub-trees grows exponentially with the size of the initial tree. Thus it is computationally impractical to search all sub-trees. REPTree yields a suboptimal tree under the restriction that a sub-tree can only be pruned if it does not contain a sub-tree with a lower classification error than itself.

Boosting
Boosting and especially AdaBoost is designed specifically for classification [HWF11]. It can be applied to any classification learning algorithm. To simplify matters the assumption is that the learning algorithm can handle weighted instances, where the weight of an instance is a positive number [HWF11]. The presence of instance weights changes the way in which a classifier’s error is calculated: It is the sum of the weights of the misclassified instances divided by the total weight of all instances, instead of the fraction of instances that are misclassified [HWF11]. By weighting instances, the learning algorithm can be forced to concentrate on a particular set of instances, namely those with high weight [HWF11]. Such instances become particularly important because there is a greater incentive to classify them correctly [HWF11]. The J48 and REPTree algorithms, are examples of learning methods that can accommodate weighted instances [HWF11].

3.4. Performance Criteria
Conceptually, in building a classifier three different sets can be identified: the training set, the validation set and the test set [Luz14]. The training set is used to create the initial classifier and train it [Luz14]. The validation set is used to experiment with different parameters of the used MLA for the classifier [Luz14]. Finally, the test set is used to measure the classification accuracy [Luz14].

This research does not split the dataset into 2 different sets, but uses an approach which is called cross validation.

Cross Validation: K-fold cross validation avoids the need of a validation set, by dividing the training set into K parts or folds [Luz14]. K − 1 folds are used to train the classifier and the remaining one is used as validation set [Luz14]. This process is repeated for each of the k folds [Luz14]. Besides these k folds, Weka uses the full training set as the last step and the result is the average values of all these calculations. A common value for k is 10, which is the value used in the experiments for this research as well.
3.4. PERFORMANCE CRITERIA

To evaluate any classification method, criteria for classification performance need to be defined. In this section the metrics to quantify the performance of the P2P classifier are discussed, beginning with False Positive Rate (FPR) [HWF11; Rah+14; DD11] and True Positive Rate (TPR) [HWF11; Rah+14; DD11; Luz14].

They are defined as follows:

\[
FPR = \frac{FP}{FP + TN} \tag{3.1}
\]

\[
TPR = \text{Recall} = \frac{TP}{TP + FN} \tag{3.2}
\]

The metrics are built upon the concept of True Positives (TPs) or Hits, True Negatives (TNs) or correct rejections, False Positives (FPs) or false alarms and False Negatives (FNs) or misses. These notions are often used in anomaly detection and traffic classification where each object is placed into one of several classes.

TPR or Recall is a metric about completeness. What % of positive flows did the classifier label as positive?

To get acquainted with the above metrics, a simple scenario is described. We want to classify traffic into P2P and non-P2P from a dataset. Suppose we have a set of 100 network flows, where 75 of these are P2P and the other 25 represent the non-P2P flows.

From this dataset, the classifier finds:

- 80 flows to be P2P, actually 67 of these are P2P, representing the True Positives, and 13 are non-P2P, which stand for the number of False Positives.
- 20 flows to be non-P2P. Of these, 12 are indeed non-P2P, but 8 flows are P2P.

The focus is on the P2P flows. True Positive Rate (or Recall) is the number of flows correctly categorized as P2P divided by the total number of flows that are actually P2P. Thus, in this scenario, the 

\[ TPR = \text{Recall} = \frac{67}{67 + 8} = 0.893. \]  

The False Positive Rate is the number of falsely classified flows as P2P to the total number of non-P2P flows,

\[ FPR = \frac{13}{13 + 8} = 0.619. \]  

A complementary measure to Recall is Precision [HWF11; Luz14] which is defined as follows:

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}
\]

Precision is the number of correctly classified P2P flows to all flows classified as the P2P type, thus, 

\[ \text{Precision} = \frac{67}{67 + 13} = 0.838. \]

The overall accuracy [HWF11] is defined as the following equation illustrates:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}
\]

Accuracy is the ratio of the sum of the TPs and TNs to the sum of TPs, TNs, FPs and FNs. Complementary to the FPR is the True Negative Rate (TNR) or specificity [HWF11]. Specificity measures the proportion of negatives which are correctly identified as such, that is in the above scenario the ratio of P2P flows which are correctly identified as not being P2P

\[
TNR = \text{Specificity} = \frac{TN}{FP + TN} \tag{3.5}
\]

To assess the performance of the proposed classification methods True Positive and False Positive Rates as classification metrics are used, as well as Precision, and F-Measure...
[HWF11; Luz14] as classification metrics. *F-Measure*, combines Precision and Recall, and is defined as:

$$F - \text{Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$  \hspace{1cm} (3.6)$$

The F-measure is included because it considers both the precision and the recall to compute a score. This score can be interpreted as a weighted average of the precision and recall, where a F-measure reaches its best value at 1 and worst score at 0.

The **Matthews Correlation Coefficient (MCC)** [HWF11; Luz14] provides a way to measure the quality of the classifier using the predicted values and the real values of each sample. It takes into consideration both false positives and false negatives, which makes it suitable for tests even when the categories are not balanced with respect to the number of samples. The MCC ranges between -1 and 1, where 1 represents a perfect prediction, -1 implies that all predictions were wrong and 0 suggests that the classifier is as good as a random prediction. The MCC is defined as:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}$$  \hspace{1cm} (3.7)$$
This chapter elaborates the proposed framework for P2P traffic classification which can separate P2P traffic from Non-P2P traffic. The framework is composed of a filter module, a dialogue generation module, an aggregation module and classifier module. The framework is inspired by the work of Narang et al. [NRH13] for the research on feature selection of P2P botnet traffic, Rahbarinia et al. [Rah+14] with their P2P traffic categorization and the work of Narang et al. [Nar+14] regarding P2P botnet detection.

The framework does not rely on Deep Packet Inspection (DPI) or signature-based mechanisms (which are considered useless when encryption is applied). This framework focuses on observing the different dialogues which takes place amongst the nodes, essentially the idea of who is talking to whom.

The host pairs are extracted from IP headers as well as a set of features which is found relevant to distinguish P2P traffic from non-P2P traffic, such as flow duration, volume size, minimum packet size, maximum packet size, etc (see Chapter 1).

The framework uses supervised machine learning algorithms and network traces of P2P applications & botnets to build models which can correctly categorize different P2P applications.

The framework is illustrated in Figure 4.1

4.1. BACKGROUND

As explained in previous chapters, instead of the standard 5-tuple approach, where source address, source port, destination address, destination port and protocol are used, this work is based on the idea that a classifier needs to be found which does not rely on port numbers (as more applications are using dynamic port numbers or masquerade over well-known ports). With numerous applications also encrypting their payload, the classifier should and does not rely on payload data neither.

This leads to adopting an approach which looks at the node endpoints only, the 2-tuple approach, consisting of the two nodes participating in a dialogue.

In short, the proposed framework is:

- A classifier which is protocol agnostic, port agnostic and payload agnostic. The only reliance is on information from the IP header.
4.2. System Design

This work relies on understanding the differentiating behavior of P2P applications (optionally including P2P botnets) from Non-P2P traffic.

4.2.1. The Dialogue

Nodes connected to their neighbors in the P2P overlay network maintain regular communication amongst themselves to check for updates, to exchange commands and/or to check if the peer is alive or not. Since certain benign P2P applications (and certainly botnets) are known to use dynamic port numbers, the regular 5-tuple flow definition will not be able to give a clear picture of the activity a host is engaged in [NRH13]. This traditional 5-tuple definition will create multiple flows out of what is actually a single conversation happening between two nodes (although happening on different ports), and thus give a false view of the communications happening in the network [NRH13]. A helicopter view of the dialogues between the P2P nodes is the approach used here to distinguish P2P from Non-P2P.

4.2.2. P2P Traffic

All P2P applications whether malicious or benign operate with specific control messages. These P2P application specific messages are used by nodes to connect to the P2P network, send out request for resources, send out response messages, joining or leaving the network etc. Since each application has its own specific control messages, the patterns hidden in these control messages is where the classification focuses to categorize different P2P applications by considering the median value of the inter arrival time (IAT) of packets for each different P2P application.

With respect to P2P Botnets, P2P bot communications will tend to be low in volume, but the bots keep contacting each other at certain intervals of time, and thus the duration of their dialogues will be large. [Rah+14]. Although bots remain connected to each other...
and thus exhibit long dialogue durations, a benign P2P node’s conversation with another is not expected to be long [Rah+14; NRH13]. In summary, four features are extracted from the datasets and these are used to differentiate P2P traffic from Non-P2P traffic. The four features used in this work are [Nar+14; NRH13]:

1. The median value of the IAT of packets.
2. The duration.
3. The number of packets.
4. The volume of data.

4.2.3. Details

Filter: This module reads raw packet data from offline network trace files. The module reads each packet and discards packets without a valid IPv4 header. From this set, only those packets which contain payload and have a valid TCP or UDP header are kept. This discards packets with zero payload or other protocols besides TCP and UDP (ICMP, ARP etc.) Source IP, Destination IP, Payload length and Timestamp are extracted and then stored for future use.

This module algorithm is found in Algorithm 1.

**Algorithm 1** Filter Module

```java
function FILTER(packets)
    ArrayList<Pkt> filteredPkts;
    for Packet p in packets do
        epoch ← p.getepoch();
        header ← p.getheader();
        if header and header.getType() in [TCP, UDP] then
            SourceIP ← header.getSourceIP();
            DestinationIP ← header.getDestinationIP();
            pSize ← header.getPayloadSize();
            if payloadSize not null or zero then
                nextPkt ← Pkt(SourceIP, DestinationIP, pSize, epoch);
                filteredPkts.add(nextPkt);
            end if
        end if
    end for
    return filteredPkts;
end function
```

Dialogue generation Module: The output of the Filter module is sent to the this module as input. This module creates a list of the dialogues by aggregating the retrieved packets (send by the filter). Each dialogue is identified by the combination of <IP1,IP2> and an initial INTERVAL value of 2 seconds. The dialogues are created with the idea that a uni-directional flow can be converted to a bi-directional flow if source (A) and destination (B) ip pairs match and they contacted each other within the INTERVAL time sinds the first
packet of either A→B or B→A. While iterating through the packets, if a packet is found which belongs to the IP pair the dialogue and the time-stamp lies within INTERVAL time from the beginning or end of the dialogue, the packet is added to this dialogue and the attributes of the dialogue are modified accordingly [Nar+14].

This is illustrated in Algorithm 2 [Nar+14]

**Algorithm 2 Dialogue Module**

```java
function GENDLGS(filteredPkts)
    ArrayList<Dialogue> initDlgList;
    ArrayList<PacketGroup> pgList;
    pgList ← filteredPkts.groupPktsByIPpair();
    for PacketGroup pg in pgList do
        sort packets in pg by timestamp;
        nextDlg ← Dialogue(NULL);
        for Packet p in pg do
            if p.timestamp between (nextDlg.start - INTERVAL) && (nextDlg.end + INTERVAL) then
                nextDlg.addPacket(p);
            else
                nextDlg ← Dialogue(p);
                initDlgList.add(nextDlg);
            end if
        end for
    end for
    return initDlgList;
end function
```

**Aggregation Module**: The dialogues created in the previous module are aggregated for a 1 hour interval. This means that several dialogues between the same IP pair combination are aggregated to a single dialogue. This 1 hour interval can be adjusted to higher values as needed, providing the flexibility to look at the data for any desired aggregated time-period (e.g. 3 hours, a day, etc.) Such flexibility is especially valuable for bots which are extremely stealthy in their communication patterns and exchange as low as a few packets every few hours. From the datasets obtained, it is found that the zeus botnet exposes this behavior [Rah+14; Nar+14]. Besides the fact that this behavior is dramatically different from the others, the number of dialogues found for zeus within the 1 hour frame was significantly lower than the rest. The number of dialogs per interval is illustrated in Figure 4.2

Zeus requires special attention and evaluation, resulting in the zeus dataset being excluded from the experiments. The outcome of the aggregation module is then used to train the classification model. The attributes of each dialogue which are analyzed are: number of packets, volume, duration and the Median value of IAT [NRH13; Nar+14].

The aggregation module’s algorithm is seen in Algorithm 3 [Nar+14].

**Classification Module**: The Classification module uses Weka’s supervised machine learning algorithms for training the model and classifying the test data. Models of the classification were created using two Decision Trees(DTs) algorithms, namely J48 and REPTRee with and without Boosting.
Algorithm 3 Aggregation Module

```plaintext
function AGGREGATION(initDlgList, INTERVAL))
    ArrayList < Dialogue > finalDlgList;
    ArrayList < DlgGroup > dgList;
    dgList ← initDlgList.groupDlgByIPpair();
    for DlgGroup dg in dgList do
        sort dialogues in dg by timestamp;
        nextDlg ← Dialogue(NULL);
        for Dialogue d in dg do
            if d.timestamp between (nextDlg.start - INTERVAL) && (nextDlg.end + INTERVAL) then
                nextDlg.addDlg(d);
            else
                nextDlg ← Dialogue(d);
                finalDlgList.add(nextDlg);
            end if
        end for
    end for
    return finalConvList;
end function
```
5.1. DATA COLLECTION

The datasets used in this work are from:

• A dataset from Shiravi et al. [Shi+12].
  Dataset¹, consists of labelled network traces, including full packet payloads that are publicly available to researchers. The main dataset of interest for this research is the dataset of general Non-P2P internet usage.

• A dataset from Rahbarinia et al. [Rah+14].
  This consists of two main datasets: a dataset of P2P traffic generated by a variety of P2P applications and a dataset of traffic from three modern P2P botnets.

There are three main datasets for the training of the P2P classifier:

• a dataset of P2P traffic generated by a variety of P2P applications.

• a dataset of traffic from three modern P2P botnets.

• a dataset of non-P2P traffic.

(D1) Ordinary P2P Traffic: The P2P Traffic was obtained from Rahbarinia et al. This set is created between mar. 26 2011 - apr. 11 2011. The following describes how it was created:
To collect the P2P traffic dataset, an experimental lab network consisting of 11 distinct hosts was build, which was used to run 5 different popular P2P applications for several weeks. Specifically, 9 hosts were dedicated to running Skype, and the two remaining hosts to run, at different times, eMule, µTorrent, Frostwire, and Vuze. This choice of P2P applications provided diversity in both P2P protocols and networks (see Table 5.1). The 9 hosts dedicated to Skype were divided into two groups: two hosts were configured with high-end hardware, public IP addresses, and no firewall filtering. This was done so that these hosts had a chance to be elected as Skype super-nodes. The remaining 7 hosts were configured using filtered IP addresses, and resided in distinct sub-networks. Using both filtered and unfiltered hosts allowed sample collection of Skype traffic that may be witnessed in different real-world scenarios. For each host, a separate Skype account was created and some

¹http://www.iscx.ca/datasets
of these accounts were made “friends” with each other and with Skype instances running on machines external to the lab. In addition, using AutoIt [Ben09], a number of scripts were created to simulate user activities on the host, including periodic chat messages and phone calls to friends located both inside and outside of the lab network. Overall, 83 days of a variety of Skype traffic was collected, as shown in Table 5.1.

The other two distinct unfiltered hosts were used to run each of the remaining legitimate P2P applications. For instance, initially these two hosts were used to run two instances of eMule for about 9 consecutive days. During this period, a variety of file searches and downloads\(^2\) were initiated. Whenever possible, AutoIt [Ben09] was used to automate user interactions with the client applications. This process was replicated to collect approximately the same amount of traffic from FrostWire, µTorrent, and Vuze.

(D2) P2P Botnet Traffic: In addition to popular P2P applications, several days of traffic from three different P2P-botnets: Storm, Waledac, and Zeus were obtained. It is worth noting that the Waledac traces were collected before the botnet takedown enacted by Microsoft, while the Zeus traces are from a version which is likely still active and relies entirely on P2P-based command-and-control (C&C) communications. Table 5.1 indicates the number of hosts and days of traffic we obtained, along with information about the underlying transport protocol used to carry P2P management traffic.

(D3) Non-P2P Traffic: The Non-P2P Traffic was obtained from Shiravi et al. This dataset contained only Non-P2P traffic.

<table>
<thead>
<tr>
<th>Application</th>
<th>Protocol</th>
<th>Hosts</th>
<th>Capture Days</th>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skype</td>
<td>Skype</td>
<td>9</td>
<td>83</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>eMule</td>
<td>eDonkey</td>
<td>2</td>
<td>9</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>FrostWire</td>
<td>Gnutella</td>
<td>2</td>
<td>9</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>µTorrent</td>
<td>BitTorrent</td>
<td>2</td>
<td>9</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Vuze</td>
<td>BitTorrent</td>
<td>2</td>
<td>9</td>
<td>TCP/UDP</td>
</tr>
</tbody>
</table>

Table 5.1: P2P traffic dataset summary

### 5.2. Attribute Importance

In order to measure which attributes provide perform better for the data prediction, the attribute evaluator is used with the Information Gain Ranking Filter.

Figure 5.1 shows the attribute ranking on the training dataset. The training dataset is composed of:

- 20,000 samples from eMule

\(^2\)To avoid potential copyright issues, assurance was made to never store the downloads permanently.
5. Data Analysis

Figure 5.1: Attribute importance on training dataset

- 20,000 samples from uTorrent
- 20,000 samples from FrostWire
- 20,000 samples from Vuze
- 20,000 samples from Waledac
- 20,000 samples from Storm
- 30,000 samples from the Non-P2P set.

From these figures, it is observed that the attributes IAT and nrOfBytes are of more importance than nrOfPackets and Duration.

5.3. Classifier Metrics

This work was evaluated using network trace datasets obtained from Shiravi et al. and Rahbarinia et al. Within these datasets, data from four P2P applications (eMule, uTorrent, FrostWire and Vuze) and two P2P botnet applications (Waledac and Storm) were used. According to Chapter 4, offline packet captures were parsed to create and further aggregate into dialogues. The data obtained in this fashion was labeled to form a so called “labeled training set” for each application. It is important to note that in the traces of Storm and Waledac, the number of known ‘malicious hosts’ are 13 and 3 respectively. It is not known whether the other IP addresses seen in the network traces are benign or malicious. For the experiments, a dialogue is treated as ‘malicious’ as either of the IPs (either source or destination) is known to be ‘malicious’.

In order to not over-estimate the accuracy of the results, the Decision Tree (DT) Machine Learning Algorithms (MLAs) were used with ten-fold cross-validation.
DTs are simple to train and fast algorithms. However, they tend to create complex tree structures and over-fit the data. Although this gives high accuracy on the training data and even over the test data such detection models may not generalize to a real-world scenario. Besides using J48 and REPTrees, boosting was used with the trees to increase the accuracy obtained from a single classifier. The AdaBoost meta-classifier of Weka was used, with 10 trees.

Figure 5.2 gives the results for ten-fold cross validation performed with the J48, REPTree Boosted J48 and boosted REPTree.

![Classifier Performance](image.png)

Figure 5.2: Weka's Algorithm accuracy

The test datasets used were:

- **Full test set.** This set contained:
  - 15,000 samples from eMule
  - 15,000 samples from uTorrent
  - 15,000 samples from FrostWire
  - 15,000 samples from Vuze
  - 15,000 samples from Waledac
  - 15,000 samples from Storm
  - 20,000 samples from the Non-P2P set.

- **eMule test set,** containing:
  - 5,000 samples from eMule
  - 1,667 samples from the Non-P2P set.

- **uTorrent test set,** containing:
As Tables 5.2, 5.3, 5.4 and 5.5 show, the classifier could consistently give more than 99.9% accuracy in detection of the two P2P botnet applications, and at least 93.7% accuracy in the detection of the four P2P applications, with very low false positives. As an overall average, the accuracy stands above 95%. The classifier performance on the full test set is shown in Table 5.6.

Figure 5.3 shows the algorithm’s accuracy per individual P2P application.

<table>
<thead>
<tr>
<th>P2P Application</th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMule</td>
<td>0.980</td>
<td>0.098</td>
<td>0.966</td>
<td>0.975</td>
<td>0.962</td>
<td>0.901</td>
</tr>
<tr>
<td>uTorrent</td>
<td>0.941</td>
<td>0.094</td>
<td>0.966</td>
<td>0.953</td>
<td>0.932</td>
<td>0.828</td>
</tr>
<tr>
<td>FrostWire</td>
<td>0.987</td>
<td>0.085</td>
<td>0.971</td>
<td>0.979</td>
<td>0.968</td>
<td>0.917</td>
</tr>
<tr>
<td>Vuze</td>
<td>0.986</td>
<td>0.089</td>
<td>0.969</td>
<td>0.978</td>
<td>0.967</td>
<td>0.913</td>
</tr>
<tr>
<td>Waledac</td>
<td>0.999</td>
<td>0.101</td>
<td>0.966</td>
<td>0.982</td>
<td>0.973</td>
<td>0.929</td>
</tr>
<tr>
<td>Storm</td>
<td>1.000</td>
<td>0.081</td>
<td>0.972</td>
<td>0.986</td>
<td>0.979</td>
<td>0.945</td>
</tr>
<tr>
<td>Weighed avg.</td>
<td>0.982</td>
<td>0.091</td>
<td>0.968</td>
<td>0.976</td>
<td>0.964</td>
<td>0.906</td>
</tr>
</tbody>
</table>

Table 5.2: Boosted REPTree Performance Per P2P Application

Figure 5.4 shows this classification TPR compared with the work of Rahbarinia et al. [Rah+14].
### 5.3. Classifier metrics

#### Boosted J48

<table>
<thead>
<tr>
<th></th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMule</td>
<td>0.986</td>
<td>0.087</td>
<td>0.970</td>
<td>0.978</td>
<td>0.967</td>
<td>0.913</td>
</tr>
<tr>
<td>uTorrent</td>
<td>0.940</td>
<td>0.090</td>
<td>0.967</td>
<td>0.954</td>
<td>0.932</td>
<td>0.830</td>
</tr>
<tr>
<td>FrostWire</td>
<td>0.989</td>
<td>0.078</td>
<td>0.973</td>
<td>0.981</td>
<td>0.972</td>
<td>0.926</td>
</tr>
<tr>
<td>Vuze</td>
<td>0.987</td>
<td>0.082</td>
<td>0.972</td>
<td>0.979</td>
<td>0.969</td>
<td>0.919</td>
</tr>
<tr>
<td>Waledac</td>
<td>1.000</td>
<td>0.094</td>
<td>0.968</td>
<td>0.984</td>
<td>0.975</td>
<td>0.936</td>
</tr>
<tr>
<td>Storm</td>
<td>1.000</td>
<td>0.082</td>
<td>0.972</td>
<td>0.986</td>
<td>0.979</td>
<td>0.945</td>
</tr>
<tr>
<td>Weighed avg.</td>
<td>0.984</td>
<td>0.086</td>
<td>0.970</td>
<td>0.977</td>
<td>0.966</td>
<td>0.912</td>
</tr>
</tbody>
</table>

Table 5.3: Boosted J48 Performance per P2P Application

#### REPTree

<table>
<thead>
<tr>
<th></th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMule</td>
<td>0.982</td>
<td>0.111</td>
<td>0.962</td>
<td>0.972</td>
<td>0.957</td>
<td>0.888</td>
</tr>
<tr>
<td>uTorrent</td>
<td>0.937</td>
<td>0.101</td>
<td>0.964</td>
<td>0.950</td>
<td>0.927</td>
<td>0.817</td>
</tr>
<tr>
<td>FrostWire</td>
<td>0.989</td>
<td>0.098</td>
<td>0.966</td>
<td>0.978</td>
<td>0.967</td>
<td>0.912</td>
</tr>
<tr>
<td>Vuze</td>
<td>0.985</td>
<td>0.096</td>
<td>0.967</td>
<td>0.976</td>
<td>0.964</td>
<td>0.906</td>
</tr>
<tr>
<td>Waledac</td>
<td>0.999</td>
<td>0.098</td>
<td>0.967</td>
<td>0.982</td>
<td>0.974</td>
<td>0.931</td>
</tr>
<tr>
<td>Storm</td>
<td>0.999</td>
<td>0.086</td>
<td>0.970</td>
<td>0.985</td>
<td>0.977</td>
<td>0.940</td>
</tr>
<tr>
<td>Weighed avg.</td>
<td>0.982</td>
<td>0.098</td>
<td>0.966</td>
<td>0.974</td>
<td>0.961</td>
<td>0.899</td>
</tr>
</tbody>
</table>

Table 5.4: REPTree Performance per P2P Application

#### J48

<table>
<thead>
<tr>
<th></th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMule</td>
<td>0.982</td>
<td>0.099</td>
<td>0.966</td>
<td>0.974</td>
<td>0.961</td>
<td>0.897</td>
</tr>
<tr>
<td>uTorrent</td>
<td>0.935</td>
<td>0.090</td>
<td>0.967</td>
<td>0.951</td>
<td>0.929</td>
<td>0.821</td>
</tr>
<tr>
<td>FrostWire</td>
<td>0.990</td>
<td>0.090</td>
<td>0.969</td>
<td>0.979</td>
<td>0.969</td>
<td>0.919</td>
</tr>
<tr>
<td>Vuze</td>
<td>0.985</td>
<td>0.102</td>
<td>0.965</td>
<td>0.975</td>
<td>0.963</td>
<td>0.902</td>
</tr>
<tr>
<td>Waledac</td>
<td>0.999</td>
<td>0.099</td>
<td>0.966</td>
<td>0.982</td>
<td>0.973</td>
<td>0.931</td>
</tr>
<tr>
<td>Storm</td>
<td>1.000</td>
<td>0.084</td>
<td>0.971</td>
<td>0.986</td>
<td>0.978</td>
<td>0.944</td>
</tr>
<tr>
<td>Weighed avg.</td>
<td>0.982</td>
<td>0.094</td>
<td>0.967</td>
<td>0.975</td>
<td>0.962</td>
<td>0.902</td>
</tr>
</tbody>
</table>

Table 5.5: J48 Performance Per P2P Application
Table 5.6: Classifier Performance Per Algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosted REPTree</td>
<td>0.982</td>
<td>0.076</td>
<td>0.983</td>
<td>0.983</td>
<td>0.972</td>
<td>0.904</td>
</tr>
<tr>
<td>Boosted J48</td>
<td>0.983</td>
<td>0.071</td>
<td>0.984</td>
<td>0.984</td>
<td>0.974</td>
<td>0.912</td>
</tr>
<tr>
<td>REPTree</td>
<td>0.982</td>
<td>0.081</td>
<td>0.982</td>
<td>0.982</td>
<td>0.970</td>
<td>0.900</td>
</tr>
<tr>
<td>J48</td>
<td>0.982</td>
<td>0.079</td>
<td>0.983</td>
<td>0.982</td>
<td>0.971</td>
<td>0.902</td>
</tr>
</tbody>
</table>

Figure 5.3: Algorithm accuracy performance per application

Figure 5.4: Comparing this research with the work of Rahbarinia et al. [Rah+14]
CONCLUSION

P2P applications have been widely used recently. They bring us many conveniences, but increasing P2P traffic also brings P2P malware with it. One of the first steps against P2P malware is detection of P2P traffic, an accurate P2P traffic classification becomes more and more significant.

The answers to the research sub questions are formulated as follows:

• RQ1: Which algorithm is suitable for P2P traffic classification?
  Adaboost with J48 is selected as an implementation algorithm of a statistics-based classifier based on the analysis results..

• RQ2: What relevant features are needed for P2P traffic classification?
  There are four features found to be relevant for P2P traffic classification when an algorithm based on Decision Trees(DTs) is used. These are:
  1. The median value of the IAT of packets.
  2. The duration.
  3. The number of packets.
  4. The volume of data.

• RQ3: How do we apply a port agnostic, payload agnostic classification technique into a P2P traffic classification approach?
  By restricting the feature set to not rely on port and payload data we can create a P2P traffic classifier which is both port and payload agnostic.

• RQ4: How effective is this P2P traffic classification approach?
  The classifier in this research is able to achieve high accuracy (>98%) compared with other P2P classifiers.

The main research question can now be answered:

How do we classify network traffic into P2P and non-P2P traffic?

Network traffic classification into P2P and non-P2P traffic is achieved with Machine Learning (ML) as a method for P2P traffic classification, using the algorithms J48, REPTree
and AdaBoost for analysis of statistical flow features, which are both port and payload agnostic.

Compared with other existing detection schemes, this approach also exhibits high accuracy. The accuracies of other schemes [Che+09], [Li+09], [KNC10], [YC13] are 95.67%, 96.03%, 95% and 97.46% respectively.

6.1. LIMITATIONS
The framework is able to correctly detect and categorize P2P applications—whether malicious or benign—with high accuracy. There are some limitations though and this section describes these. The accuracy obtained with the classification of benign P2P applications is lower when compared to the accuracy of detection of P2P botnets. Adding more benign P2P applications to the training dataset or possibly increasing the sample size and experimenting with several other features can help in correctly categorizing P2P traffic such as the control (or management) traffic information of P2P applications.

By design, being port and protocol agnostic makes that many lower-level details (e.g. at the TCP/UDP layer) are neglected.

If node A and node B are engaged in P2P file sharing and both of them are also a part of a botnet, this is seen as a single dialogue. Because of an overlap between the botnet data and application data, dialogue approach is unable to correctly classify the kind of botnet or application running on peers A & B. It is therefore possible that smarter bots which mimic benign-like behavior and/or add noise (or randomness) to their communication patterns, could evade the current classifier.

One noteworthy remark is the fact that all packets having no payload are discarded. This was necessary to remove corrupted packets and sanitize the network traces obtained. However, such an approach has an inherent drawback of dropping all legitimate packets with zero payload, such as TCP connection establishment (SYN) packets. This could be exploited by using zero payload TCP packets (SYN or ACK packets) to exchange simple commands for instance in a botnet.

6.2. RELATED WORK
The work by [Nar+14] and a recent survey by Gomes et al. [Gom+13] are currently the best reference to related work. While P2P traffic detection has been a topic of much research, P2P traffic categorization has received very little attention. In the following, the most relevant work on P2P traffic categorization is described.

Hu et al. [HCL09] use flow statistics to build traffic behavior profiles for P2P applications. However, [HCL09] does not attempt to separate P2P control and data transfer traffic. Because data transfer patterns are highly dependent on user behavior, the approach proposed [HCL09] may not generalize well to P2P traffic generated by different users. Furthermore, [HCL09] is limited to modeling and categorizing only two benign P2P applications (BitTorrent and PPLive), and does not consider malicious P2P applications at all.

In [Haq+10], Haq et al. discuss the importance of detecting and categorizing P2P traffic to improve the accuracy of intrusion detection systems. However, they propose to classify P2P traffic using deep packet inspection, which does not work well in case of encrypted P2P traffic. More recently, a number of studies have addressed the problem of detecting
6.3. **FUTURE WORK**

A recommendation for future work is to increase the accuracy of benign P2P applications. Possibilities are for instance to increase the number of benign P2P applications in the various training and test sets. Incorporating other features (still being port and payload agnostic), whether replacing or adding features, can also be part of the way forward.

6.4. **REFLECTION**

Working on the P2P classification framework was a pleasant, educational but also stressful journey. The received data sets for instance are huge regarding file sizes. Wireshark\(^1\) had difficulties reading the offline trace files. Windows laptop with 4Gb, Mac OSX desktop with 8Gb were not enough. Even a dedicated quad-core Windows Desktop with 32Gb was not able to satisfy the (memory) resource hunger of the Wireshark tools. In the end, self-written tools were applied to chop the big data files into smaller chunks. Early experiments with training data with millions of instances resulted in Weka being sent for calculation journeys of several days, which ended in not enough memory or system crashes.

All those setbacks aside, diving into Machine Learning and Machine Learning Algorithms was really educational as those topics are not in the current curricula.

6.5. **MIT-LICENSE**

All code is distributed using the MIT-License and will be available at: https://github.com/pmolijn

The MIT-License states:

Copyright (c)2014 Patrick Molijn, Open Universiteit Nederland

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
BIBLIOGRAPHY

BOOKS


ACADEMIC ARTICLES


TECHNICAL DOCUMENTATION


Pavel Piskac and J Novotny. "Network Traffic Classification Based on Time Characteristics Analysis". Masaryk University Faculty of Informatics, 2011.


<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network.</td>
</tr>
<tr>
<td>AOC</td>
<td>Area Under Curve.</td>
</tr>
<tr>
<td>C&amp;C</td>
<td>Command and Control.</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of Applications with Noise.</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service.</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Table.</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System.</td>
</tr>
<tr>
<td>DPI</td>
<td>Deep Packet Inspection.</td>
</tr>
<tr>
<td>DT</td>
<td>Decision Tree.</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation-Maximization.</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative.</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive.</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate.</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol.</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model.</td>
</tr>
<tr>
<td>GTVS</td>
<td>Ground Truth Verification System.</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure.</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority.</td>
</tr>
<tr>
<td>KNN</td>
<td>$K$ Nearest Neighbor.</td>
</tr>
<tr>
<td>LC</td>
<td>Linear Classifier.</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminant Analysis.</td>
</tr>
<tr>
<td>MCC</td>
<td>Matthews Correlation Coefficient.</td>
</tr>
<tr>
<td>MLA</td>
<td>Machine Learning Algorithm.</td>
</tr>
<tr>
<td>NB</td>
<td>Naïve Bayes.</td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbor.</td>
</tr>
<tr>
<td>QDA</td>
<td>Quadratic Discriminant Analysis.</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service.</td>
</tr>
<tr>
<td>QT</td>
<td>Quality Threshold.</td>
</tr>
<tr>
<td>REPTree</td>
<td>Reduced Error Pruned Tree.</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine.</td>
</tr>
<tr>
<td>TDG</td>
<td>Traffic Dispersion Graph.</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative.</td>
</tr>
<tr>
<td>TNR</td>
<td>True Negative Rate.</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive.</td>
</tr>
<tr>
<td>TPR</td>
<td>True Positive Rate.</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP.</td>
</tr>
</tbody>
</table>
**Glossary**

**bot** A bot is a compromised computer with malicious software installed.

**botherder** See botmaster.

**botmaster** User who controls a botnet.

**botnet** A botnet is a network of bots and are controlled by a botmaster.

**centroid** A centroid is a data point (imaginary or real) at the center of a cluster.

**malware** malicious software.

**P2P** A Peer-to-Peer (P2P) is a type of decentralized and distributed network architecture in which individual nodes in the network (called "peers") act as both suppliers and consumers of resources, in contrast to the centralized client–server model where client nodes request access to resources provided by central servers.

**servent** A servent is a host within a computer network acting as both a SERVer and a client.

**Supervised learning** Supervised learning algorithms are trained on labelled examples, i.e., input where the desired output is known. The supervised learning algorithm attempts to generalise a function or mapping from inputs to outputs which can then be used speculatively to generate an output for previously unseen inputs.

**swarm** A swarm is a collection of peers that are interested in distributing the same content.

**Unsupervised learning** Unsupervised learning algorithms operate on unlabelled examples, i.e., input where the desired output is unknown. Here the objective is to discover structure in the data (e.g. through a cluster analysis), not to generalise a mapping from inputs to outputs.