ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Daniel Roman Koller

Application Detection and Modeling using Network Traces

Master Thesis MA-2007-14
March 2007 to September 2007

Tutor: Bernhard Tellenbach
Co-Tutor: Daniela Brauckhoff
Supervisor: Prof. Bernhard Plattner

Acknowledgement

This master thesis would not have been possible without the support of many people. The author wishes to express his gratitude to his tutor, Berhard Tellenbach who was abundantly helpful and offered invaluable assistance, support and guidance. Furthermore, special thanks go to the co-tutor Daniela Brauckhoff and the other members of the Communication System Group (CSG), ETH Zurich, especially to Prof. Dr. Berhard Plattner.
Special thanks also to my classmates, Pascal Gamper, Stefan Keller, Roman Suterm Manthos Takidis, Dominique Guth and also Theus Hossmann for premises and countless hours playing foosball. Last but not least, the author wants to thank his girlfriend Claudia Wyrsch for everything.

Abstract

The motivation for this thesis is the broad diversity and the remaining problems of the existing work in Internet traffic analysis. During the last years many approaches, different in their methodologies and their ambitions, have been published. Internet traffic analysis is used in many fields, e.g. traffic engineering, network design, network security, etc.
Therefore, this thesis has the goal to compare as many of the existing approaches as possible and to point out their strengths and weaknesses as well as their potential for improvements. As result a survey paper is planed to be published. For a better comprehension of the field of Internet traffic analysis, an application detection approach for a newly raised application class is designed: The traffic of the Internet Protocol Television (IPTV) application Zattoo. Its traffic is detected by a semi-port-number-based approach and afterward several attributes of this traffic and of the host producing it are investigated. Totally 19 weeks of traffic from the Swiss Education and Research Network SWITCH are processed and therefore also the temporal consistency of the traffic and host properties can be explored.

Contents

1 Introduction **11**
- 1.1 Motivation . . . 11
  - 1.1.1 Traffic Classification . . . 11
  - 1.1.2 Host Classification . . . 12
  - 1.1.3 Application Detection . . . 12
  - 1.1.4 IPTV . . . 13
- 1.2 The Task . . . 14
- 1.3 Overview . . . 14

2 Problems in Application Detection **17**
- 2.1 Traffic Data Collection . . . 17
- 2.2 Passive versus Active . . . 20
- 2.3 User Behavior versus Machine Behavior . . . 20
- 2.4 Grade Of Automatism . . . 22
  - 2.4.1 Traffic Classification . . . 22
  - 2.4.2 Application Detection . . . 22
  - 2.4.3 Host Classification . . . 22

3 Related Work **25**
- 3.1 Active Approaches . . . 25
- 3.2 Passive Approaches . . . 25
  - 3.2.1 Port Number Analysis . . . 25
  - 3.2.2 Payload-based Analysis . . . 26
  - 3.2.3 Transport-layer Heuristics . . . 26
  - 3.2.4 Machine Learning Approaches . . . 26
- 3.3 Presentation of existing Work . . . 26
  - 3.3.1 Analyzing peer-to-peer traffic across large networks . . . 27
  - 3.3.2 File-sharing in the Internet: A characterization of P2P traffic in the backbone 27
  - 3.3.3 Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures . . . 27
  - 3.3.4 Transport Layer Identification of P2P Traffic . . . 28
  - 3.3.5 Application-Level Traffic Monitoring and Analysis on IP Networks . . . 28
  - 3.3.6 Class-of-Service Mapping for QoS: A statistical Signature-based Approach to IP Traffic Classification . . . 29
  - 3.3.7 BLINC: Multilevel Traffic Classification in the Dark . . . 29
  - 3.3.8 ACAS: Automated Construction of Application Signatures . . . 30
  - 3.3.9 Toward the Accurate Identification of Network Applications . . . 30
  - 3.3.10 Characteristic analysis of Internet traffic from the perspective of flow . . . 30
  - 3.3.11 Identifying Known and Unknown Peer-to-Peer Traffic . . . 31
  - 3.3.12 Self-learning IP Traffic Classification based on Statistical Flow Characteristics . . . 31
  - 3.3.13 Flow-based Identification of P2P Heavy-Hitters . . . 31
  - 3.3.14 Internet Traffic Identification using Machine Learning . . . 32
  - 3.3.15 Bayesian Neural Networks For Internet Traffic Classification . . . 32
  - 3.3.16 A measurement study of correlations of Internet flow characteristics . . . 33
  - 3.3.17 Traffic Classification Using Clustering Algorithms . . . 33

3.3.18 Traffic Modeling and Classification Using Packet Train Length and Packet Train Sizes . . . 34
3.3.19 Traffic Classification On The Fly . . . 34
3.3.20 Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing . . . 34
3.3.21 Flow Clustering Using Machine Learning Techniques . . . 34
3.3.22 Offline/Realtime Traffic Classification Using Semi-Supervised Learning . . . 35
3.3.23 Traffic Classification through Simple Statistical Fingerprinting . . . 35
3.3.24 Profiling the End Host . . . 35
3.3.25 Early Application Detection . . . 36
3.3.26 Identifying and Discriminating Between Web and Peer-to-Peer Traffic in the Network Core . . . 36
3.4 Summary . . . 36

4 Zattoo Detection **43**
4.1 Introduction . . . 43
4.2 Requirements . . . 44
4.3 Analysis and Design . . . 45
4.3.1 Manual Analysis of the Zattoo Traffic . . . 45
4.3.2 Design . . . 46
4.3.3 Data Preprocessing . . . 46
4.3.4 Main Processing Loop . . . 46
4.3.5 Accounting and Reseting . . . 47
4.4 Evaluation . . . 49

5 Results **51**
5.1 Summations and averages over the whole period . . . 51
5.1.1 Average Packet Size . . . 51
5.1.2 Average packet inter-arrival time . . . 52
5.1.3 Data Rate . . . 54
5.1.4 Bytes per Flow . . . 55
5.1.5 Packets per Flow . . . 56
5.1.6 Distribution of the Zattoo bytes and flows on the non-standard port numbers 57
5.2 Overall Traffic Statistics . . . 57
5.3 Chronological Sequences of the weeks 10 to 19 . . . 62

6 Conclusion **65**
6.1 Conclusions out of the survey of the existing approaches . . . 65
6.2 Conclusions out of the IPTV detection . . . 65

7 Outlook **67**

A Originalproblem **69**
A.1 Introduction . . . 69
A.1.1 Cluster and NetFlow Data Set . . . 70
A.2 The Task . . . 71
A.2.1 Survey of existing approaches to application identification . . . 71
A.2.2 A study of IPTV traffic in the SWITCH network . . . 71
A.3 Deliverables . . . 72
A.3.1 Documentation structure . . . 72
A.4 General Information . . . 72

B Zattoo detection Algorithm **73**
B.1 Data Structures for the Zattoo Hosts . . . 73
B.1.1 Internal Zattoo Host . . . 73
B.1.2 Externals Zattoo Host . . . 74
B.1.3 Port-based Zattoo Host . . . 75
B.2 Algorithm to calculate the Variance iteratively . . . 76

List of Figures

1.1 Traffic and Host Classification, Application Detection 13

2.1 Traffic Data collection and Processing 18

4.1 Zattoo Bussiness Position in the Internet television market 43
4.2 Zattoo P2P streaming Network 44
4.3 Architecture of the DDoSVax project 45

5.1 Distribution of the Zattoo flows over the average packet size 51
5.2 Cumulated distribution of the Zattoo flows over the average packet size 52
5.3 Zattoo host distribution over the average packet size 52
5.4 Cumulated Zattoo host distribution over the average packet size 52
5.5 Zattoo flow distribution of the average packet inter-arrival time 53
5.6 Cumulated Zattoo flow distribution of the average packet inter-arrival time 53
5.7 Zattoo host distribution of the average packet inter-arrival time 53
5.8 Cumulated Zattoo host distribution of the average packet inter-arrival time 54
5.9 The Zattoo TCP flows distributed over the range of the data rate 54
5.10 The Zattoo UDP flows distributed over the range of the data rate 54
5.11 The Zattoo flows distributed cumulatively over the range of the data rate 55
5.12 The Zattoo hosts distributed over the range of the data rate 55
5.13 The cumulated number Zattoo hosts over the range of the data rate 55
5.14 The Zattoo flows distributed over the flow size 56
5.15 The cumulated number of Zattoo flows distributed over the flow size 56
5.16 The Zattoo flows distributed over the number of packets per flow 56
5.17 The cumulated number of Zattoo flows distributed over the number of packets per flow 57
5.18 TCP flows between Zattoo host on non-standard port numbers 57
5.19 UDP flows between Zattoo host on non-standard port numbers 58
5.20 Bytes between Zattoo host on non-standard port numbers 58
5.21 Active internal Zattoo hosts per hour 62
5.22 Active “portBased_zattoo_hosts” per hour 62
5.23 The temporal development of the Zattoo bytes in comparison to the total bytes . 63
5.24 The temporal development of the Zattoo packets in comparison to the total packets 63
5.25 The temporal development of the Zattoo flows in comparison to the total flows . . 64

7.1 One possible architecture of a model for application detection 67

List of Tables

2.1 The features which are applicable with the different datasets 18

3.1 Overview of some existing approaches 37
3.2 Overview of some traffic features, used by existing Approaches 38
3.3 Overview of some traffic features, used by existing Approaches continued 39
3.4 Overview of collected traffic data, used by existing Approaches continued 40
3.5 Overview of the existing work, attributes of their approaches 41

5.1 Overview of the results from the 19 weeks traffic observation 59
5.2 Overview of the results from the 19 weeks traffic observation (continued) 60
5.3 Overview of the results from the 19 weeks traffic observation (last part) 61

Chapter 1

Introduction

A broad range of operations profits from the possibilities to classify the traffic of a network in real time or off-line. The classification is usually made according to some sets of characteristics which are more or less predefined. Often the traffic is divided into classes of services or applications. Depending on the goal of a traffic analysis, the best case is, when all traffic can be assigned to a certain network application like HTTP, FTP, DNS, etc. Examples of operations which benefit from knowledge retrieved from traffic analysis are:

- network security
- traffic engineering
- network planning and design
- network monitoring services provided by a network
- accounting and billing
- social engineering

An example for the last point is [39], where relationship discovery is made to enable business-driven IT Management. Another goal might be to build profiles of the hosts from a network. These profiles can also be used in security and diagnostic operations, e.g. detection of members of bot-networks [64]. It is clear, that an Internet Service Provider (ISP) wants naturally to know what are the important properties of the traffic he's got within his network. Interesting are not only the origin and destination application of some traffic. Also various distribution of the traffic can bring useful information, e.g. byte vs. time, byte vs. AS (Autonomous System), length of the connections between two hosts, etc.

1.1 Motivation

The field of network traffic analysis can be divided in three sub-fields: **traffic classification**, **host classification** and **application detection**. These three parts differ in their intent. In traffic classification, the focus lies on the traffic. That means, only attributes and distributions of the traffic are studied, knowledge about the traffic producing applications or the source and destination hosts are not of interest. Also in host classification, where e.g. a host history might interest, only statistics of the hosts are made, without respect to the traffic producing applications. On the other hand, in application detection, it is clear, that the goal is to detect the applications or application classes producing a certain part of the traffic, or the distribution of the traffic over the applications.

1.1.1 Traffic Classification

In [38] and [41] traffic is analyzed from the perspective of a connection between two hosts. In these two papers, they have aggregated uni- or bidirectional connections between two hosts,

and the goal is not to assign a connection to a certain application, but to discover the properties of these connection. Also interesting are the distributions of these properties over time or how one property is distributed over an other property, e.g. bytes of a connection vs rate of a connection. Other measurements can include the byte/connection distributions over the different protocols TCP, UDP, ICMP, etc. The correlations between the attributes of a connection are also investigated in this field. Another important point is the variation of the attributes and their various distribution over some (long) time intervals. Daily, weekly or seasonal fluctuations are indicators showing the dependencies of the attributes/distributions witch are e.g. holidays on a campus, working time in a enterprise or a worm outbreak, etc. In this field it might be important, that all operations are made *without* any knowledge of applications, hosts, network properties, etc., because with such information, which is not introduce by the traffic itself, the results might be different, because such information can include strong temporally uncertainties since nowadays properties of applications and network infrastructure are changing rapidly.

- *Traffic Engineering*: The information about the attributes of the traffic and their distributions can be necessary in usage based pricing and accounting. The detection of so called "heavy-hitters" connections (a very small part of the Internet traffic's connection, called heavy hitters, produces the majority of all transmitted bytes [1]) is a main task in scalable differentiated services or in monitoring and modeling applications.
- (*Network*) *Security*: This knowledge is also useful in the aspect of security. Anomaly and attack detection is one of the possible applications using this information, e.g. in [43] the distribution of source and destination IP addresses vs. port numbers is used to detect DoS, Port Scans, Worms, etc.

1.1.2 Host Classification

One step more is aspired in [34] and [35]. Here the goal is to analyze the traffic from the perspective of a host. The so received behavior of a host can state much information, e.g. who tries to communicate with who, who really communicates with a host, the mix of big and small (big meaning size of a connection in bytes) connections a host has with how many other, etc. Also of interest is if a host is more a consumer or more a producer of traffic, which indicates the classical client-sever model or if the ratio of the produced and consumed traffic during a time interval is below a predefined threshold, where the host might be a member of a Peer-to-Peer (P2P) system. The main difference between traffic classification is clearly, that with the focus on the host, there opens the possibility to get information about the traffic during multiple uni- or bidirectional connections. This is of course an advantage, because this information can be used to build more complex profiles or descriptions from a host than with information about only one uni- or bidirectional connection. On the other hand, the calculations based on such multi-connection patterns needs more system power and time.

- *Traffic Engineering*: One possible interesting point is here the question if there are patterns of communication, e.g. what are the profiles of the host communication with some host with profile p. A step further might be that the profile mixture from all communication partners of some hosts with profile p might have an impact to the profile p. Also interesting would be, that if there are groups of hosts with the same or similar profile p, what is the ratio between group internal traffic and inter-group traffic.
- (*Network*) *Security*: Again, the knowledge from this field can also be helpful in (network) security. Profiles or behavior of a host might be used to detect compromised machine, e.g. if the number of simultaneous TCP connections exceeds a predefined threshold[1].

1.1.3 Application Detection

Classifying traffic according to the application or application class that produces it is an essential task for network design and planning or monitoring the trends of the Internet applications. Since the early 2000s, application detection has become a difficult task, because some applications try to hide their traffic from traditional detection mechanism like port based or payload based

[1] http://www.windowsecurity.com/articles/IDS-Part2-Classification-methods-techniques.html

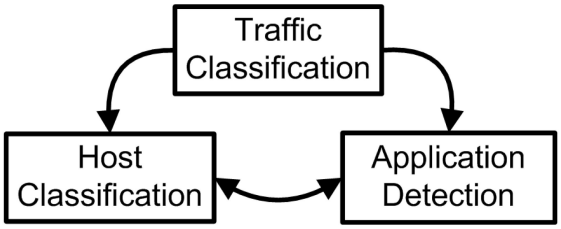

Figure 1.1: Traffic and Host Classification, Application Detection

methodologies.
One important goal of application detection is to improve the quality of service by treating some special types of applications different than others. Examples are e.g. the growing demand for interconnection network bandwidth to support the sharing of large datasets in grid-computing (so called grid bulk transfers) [53]. Other examples are database traffic, VoIP or IPTV.
On the other hand, there are many scenarios, where network administrators do not want a class of applications to run within their network. The most popular of this undesirable applications are the various P2P applications like BitTorrent [10], ares [2], eDonkey [16], KaZaA [36], Gnuttela [25], MP2P [48], Soulseek [62], WinMx [74] or Napster [49].
Many of them were treated in [21], [20], [42], [4], [6], [17], [45], [68], [11], [13], [46], [52], [58], [37], [32], [63], [31], [60]. In the last years, P2P has overtaken the place as the major contributer to the Internet traffic from traditional Web applications. The latest estimates assign up to 70% of the total Internet traffic to P2P [9]. P2P application can cause network congestion and performance degradiation of other services. Here the network owning organisation wants to shape or block the traffic, because of the enomrous bandwidth consumption of these applications or because of some policies.

- *Traffic Engineering*: If a network administrator would exactly know which applications produces the traffic in his network, he would have much less problems. With this knowledge he could exactly decide, which traffic is illegal, dangerous, unwanted, normal, special-priced, preferred, prioritized, etc. Also the detection of new applications is an important task for future provisioning of network resources.
- (*Network*) *Security*: Application detection methods are crucial methods to identify unwanted traffic and malicious applications (bot-software, spyware, worms, DDoS, etc.). Depending on policies and legal obligations, blocking some applications is also a security point which becomes more and more important for universities and enterprises.

Figure 1.1 implicates the three different levels of traffic analysis. It is clear, that host classification and application detection benefit from the results of traffic classification. All findings from analyzing e.g. byte and packet distributions can be used to improve application detection, e.g. the importance of byte accuracy [19]. Many traffic features investigated in the aspect of traffic classification are used to detect various application, the simplest example is the port number distribution. Also host classification uses all cognizance from traffic classification, e.g. the bytes produced or consumed by a host [33] or again port-number distribution from the perspective of a host [35]. In [33] the knowledge about the running applications is used to describe the behavior of a host. **On the other hand, the knowledge respecting a host could be used to classify unknown traffic between this host and other, e.g. a host is a known P2P host and has much unclassified traffic with another host known as P2P, the probability, that this unknown traffic is also P2P is increasing.**

1.1.4 IPTV

One of the newly emerged network application is IPTV. There are many different types of communication models for IPTV:

1. The classic one over HTTP, like e.g. Youtube (`http://www.youtube.com`) or Yahoo! Video (`http://video.yahoo.com`), also called Internet Television
2. Streaming services from a server, like e.g. Joost (`http//www.joost.com`)

3. (On Demand) P2P streaming, like Zattoo (http//www.zattoo.com), TVU (http//www.tvunetworks.com), PPLive (http//www.pplive.com) or SopCast (http//www.sopcast.org)

There are of course many other IPTV applications, the ones from above are, to the best of our knowledge, the most popular which can be watched without a charge. It is clear, that the classic HTTP IPTV can be classified as WEB traffic easy. But the other services are difficult to detect, because there are designed to evade Firewalls and make therefore use of almost randomly selected port numbers, even port number 80. Also it is helpful to consider, that for the services of streaming services from servers and P2P streaming exist other differences which influence the traffic patterns of this applications in a way which can not be neglected. E.g. some applications like Joost offer an On Demand video services, so does TVU, but with a P2P system. On the other hand, Zattoo provides an application where TV Channels can be selected as in the traditional Cable or Satellite Television. They have made contracts with many TV stations in Europe, so they can broadcast almost all station from e.g. Switzerland, Germany, etc. This combination of different application layer protocols, different type of service and the fact, that there are enormous potential in this field[2], make IPTV an ideal candidate to test some application detection. Additionally, there is the interesting possibility for some applications, that the users can also publish some content, which make IPTV part of the WEB 2.0 movement and does not leave it for the big companies.
We selected to construct an detection algorithm for the popular IPTV from Zattoo. This, because this start-up streams 40 TV stations with an P2P system. The P2P architecture makes it easy to provide an bandwidth which can satisfy the users and also lead the traffic through firewalls by using almost randomly chosen port numbers if necessary.

1.2 The Task

This thesis has two major tasks which can benefit from each other.

- **A Survey of existing approaches to application identification**, this is the more theoretical sub-task. First as much as possibly of the existing work has to be searched and studied. The amount of all these approaches should be reduced by considering the approaches that:
  - Bring in new algorithms, models, technique, etc.
  - Cover a new field in traffic analysis/application detection.
  - Found new significant properties of an application or application class.
  - Applied different methodologies to evaluate their work

  After the studies of these approaches, a set of characteristics should be derived to categorize the different approaches. Finally, a survey paper has to be compiled, where the most relevant results of the analysis of the existing work are presented.
- **A study of IPTV traffic in the SWITCH [65] network**, this is the more practical part. A detection algorithm should be designed and by of the experience during this work the main tasks and problems in application detection should be better understood.

1.3 Overview

Chapter 2 describes the problems in traffic analysis. E.g. the first is already at the beginning of the process of traffic analysis, which starts normally with gathering some data from a network devise. Either one get enough information from its collected data and got not enough disk space for a long term analysis or with the selected gathering method, one is able to store data from a long time interval, but with this gathering method, there is not enough information in the collected data (because of the high compression) to detect something. Chapter 3 presents

[2] http://www.heise.de/newsticker/meldung/78585

existing approaches and categorizes them according to properties introduced in chapter 2. In chapter 4, the way to the detection method for the Zattoo IPTV is presented. Chapter 5 shows the results for the detection of Zattoo traffic within the SWITCH network [65]. Finally in chapter 6 and 7 we present our conclusions and the outlook for future work.

Chapter 2

Problems in Application Detection

In the early days of the Internet, it was easy to detect, which application has produced which traffic. First, there was only a smaller amount of traffic because the available bandwidth was much smaller and the Internet was not as popular as nowadays [51]. Second, there were only a few applications producing the majority of the Internet traffic. There were HTTP, EMAIL, FTP, etc. Additionally it was clear how to identify this applications: Each application had their port number(s) and their traffic therefore could easily be captured by these numbers.
Since then, the worldwide Internet and its traffic volume and the available and necessary bandwidth has grown rapidly. Since 1995 a series of technological improvements lead to an explosion of the traffic volume: In 1995 the MPEG Audio Layer-3 (i.e. the popular MP3) was published which offer a huge compression gain, the release of the first free available MP3 players around 1997 (i.e. winamp), the video data formats as e.g. DviX and the increase of bandwidth available making broadband technology inexpensive for more and more people not only in the first world and finally the first popular file sharing network Napster [49], which revolutionized file sharing in 1999. Technically, Napster was a hybrid P2P system rather than a pure P2P system, because it made use of severs indexing the content provided by its users. Over the years the several P2P systems replaced WEB as the main contributer of the Internet traffic. As already mentioned in section 1.1.2 the latest estimates assign up to 70% of the total Internet traffic to P2P systems [9].
Of course, the ISPs began to shape the traffic from the various P2P systems and therefore the use of no-privileged port numbers was introduced in newer versions of the existing P2P systems or was standard in the newer ones.
With this history in the background being the reason for the evolution of application detection, some crucial problems of such a traffic analysis are shown in this chapter. The here presented aspects are selected, because they have a big impact on the results or performance of application detection or because, to the best of our knowledge, they got to less attention in existing work.

2.1 Traffic Data Collection

Normally, network traffic analysis starts with some traffic data collection, then there might be a preprocessing. The preprocessing could sort the collected traffic data or eventually already apply a filter, which e.g. remove all packets with invalid IP addresses and AS (Autonomous System) numbers or sorts out all uni-directional connections. The results of a preprocessing can also have an impact on the future collection of traffic data, or on the next step, the real processing. Also the result of this action might influence itself in a more or less recursive way and also the preprocessing or even the traffic data collection which could also already provide a filter. In Figure 2.1, one possible setup for a traffic analysis is shown. Of course, there are other setups, which might be more effective in some scenarios, e.g. in [24], [37] or [38]. In Figure 2.1 we try to present a general traffic application setup. Traffic can be collection from multiple network devises, e.g. more than one border gateway of an ISP. Also the advantages of a real-time traffic analysis or of an active measurement (active in the meaning, that the measurement itself also produces traffic) can be helpful for the “classic” off-line traffic analysis,

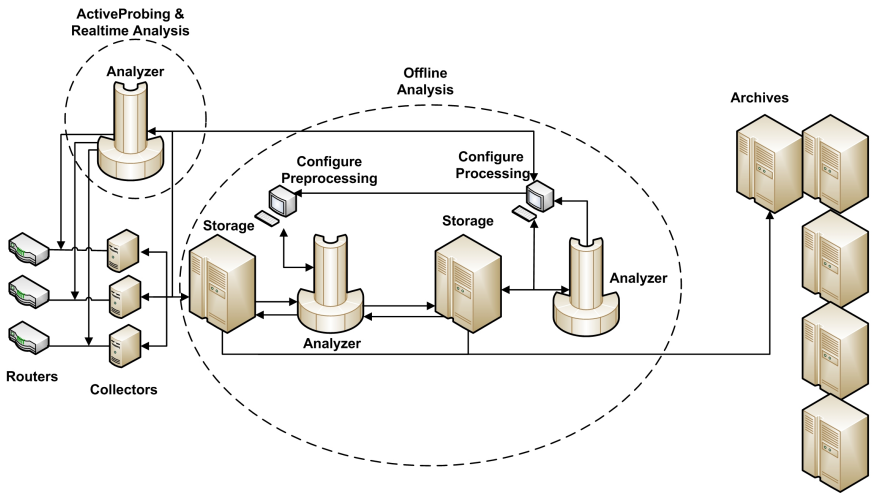

Figure 2.1: Traffic Data collection and Processing

where the collected traffic data is first stored and one has more time for the analysis of this data, e.g. to assembly the data from the different gateway routers.

The processes of traffic or host classification and application detection start usually with the collection of some data from one or more link to the Internet. The simplest way is to install a packet catcher application, e.g. *tcpdump* [66] or *ethereal* [22], on a connection to the Internet and catch all traffic running over a network device. This method is still often used to verify some measurements or the get some information about some specific not-open application [59]. This kind of collection gives all data, transport layer *packet header* and *payload*, and therefore most information possible. Even with expensive products made for enterprises there are three drawbacks: Scalability, encryption and privacy concerns. For scalability reasons it is clear, that if a link from which the whole packets are caught is very fast, enormous storage resources are needed. For examples, the traffic over the SWITCH network [65], a medium backbone from Switzerland, was 1'238'179 GBytes in June 2007. The problem caused by encryption is clear: If the payload of a packet is encrypted, it carries no information and there is only a waste of disk capacity.

	features				
data source	packet level	flow level	connection level	intra flow	multi flow
packet trace	yes	yes	yes	yes	yes
sampled packets	yes	biased	no	biased	biased
flow data	some	yes	no	no	yes
SNMP	no	no	no	no	no
server logs	some	some	some	no	some

Table 2.1: The features which are applicable with the different datasets

To reduce these drawbacks, some catch only the first few, e.g. five packets (header and payload) from a connection. But here the drawbacks of scalability (only for links with much traffic), encryption and privacy are still relevant. Another way is to randomly sample the packets, but this technique has also some drawbacks [8].
To confront these drawbacks, one possibility is to store only the packet header of each packet. Here, much of disk storage is saved and without payload the privacy concerns are smaller, but still exists because of the IP addresses. However, the challenge of encrypted payload does not exist anymore. Naturally, much information from the (unencrypted) payload is lost. But with dropping the packet payload, a traffic gathering system works also for fast links. For links with even more traffic, the principle of a *flow* was introduced.
A *flow* is defined as a uni- or bidirectional sequence of packets that have some attributes in

common. Typically the five-tuple source IP, destination IP, source port-number, destination port-number,IP protocol type. The information from the packet header is compressed to flow information as bytes/packets per flow, flow duration, etc. Depending on the implementation of the application which collects the packet and builds a *flow* from each connection, a *flow* is either unidirectional or bidirectional, therefore the feature of a *flow* can be very different. So it is important to know the properties of such a flow collecting application. E.g. when is a flow terminated? Normally after no packet was send between two host for 60s. Or when does this application terminate a flow if it is too long? Depending on the interval when the application does export its collected flows, this is e.g between 13min and 19min for the popular Cisco *NetFlow* V5 [50]. An other popular format is *CFlow* from Juniper Networks. The Internet Protocol Flow Information eXport (IPFIX) [28] is a IETF working group which was created to form a common, universal standard of export for IP flow information. They chose the Cisco NetFlow Version 9 as basis for IPFIX. Their definition [57] of a flow is:
A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. *One or more packet header field (e.g., destination IP address), transport header field (e.g., destination port-number), or application header field (e.g., RTP header fields [RFC3550])*
2. *One or more characteristics of the packet itself (e.g., number of MPLS (Multiprotocol Label Switching) labels, etc.)*
3. *One or more of fields derived from packet treatment (e.g., next hop IP address, the output interface, etc.)*

In this context it is interesting that also for UDP and ICMP, which are not connection oriented, a flow represents a kind of connection.
A step more are SNMP (Simple Network Management Protocol) and server logs from e.g. a web server or an application server. The information gathered with such instruments depends very much on the manufacturer of the router, server, etc.

Depending on the information brought in by the datasets, there are different levels of candidate attributes to be used in traffic classification or application detection [58].

- **Simple packet level:** Here are features like mean, variance or RMS of the packet size, etc. Other attributes are time series, e.g. correlation over time. The attributes on this level have the advantage that they are independent of some higher level aggregation application like flow-collectors.
- **Flow level:** Depending on the implementation of the *flow* principle, there are informations like flow-duration, bytes per flow, number of packets per flow, etc. A drawback is, that sometimes packets from more than one application layer protocol are aggregated into a flow, which could distort the flow level attributes.
- **Connection level:** Additional the the flow level information, the handshakes of a connection oriented transport-layer protocol, e.g. TCP, can bring in some new knowledge. Also the symmetry of a connection, advertised window size or throughput distributions are not contained at the flow level.
- **Intra flow/connection level:** Attributes on this level require packet level information, but in the context of a flow. Examples are packet inter-arrival statistics, loss rates or latencies.
- **Multi-flow level:** Some characteristics show up only if more than one flow are considered. For example in a P2P system there are normally data traffic and signaling traffic. This combination carries informations which can not be found if one flow is examined. Other examples are VoIP or IPTV applications which have a login or update procedure over more than one flow.

Table 2.1 form [58] shows the cohesions between the different datasets and the levels of candidate attributes. The more information is gathered, the more system power is required in respect to CPU, storage or memory. Therefore for fast links it is always a trade-off between the

amount of information contained in a dataset and the cost of infrastructure to collect, store and process these datasets.
Another problem is evaluation of traffic analysis: To evaluate the results of such a classification out of the standard dataset (e.g. NetFlow) contains often not enough information, so packet header traces are needed, for which the collection and storage is much more expensive.

The literature read for this survey uses very different datasets. There are several traces from universities, e.g. [20] or [38], and some from enterprises, e.g. [21]. Some of the earlier work have access to dataset from regional or in the best case large backbones, which contains traffic from universities, enterprises and residential, which indicates the best traffic mix. Before some traffic analysis is performed it is important to know, what is the composition of the investigated datasets. Are there any special events like a worm outbreak or especially for the universities, which often provided some services which contribute a major part to the traffic, what are these services. Also it is important to have datasets which cover different weekdays on many daytimes, because a part of the traffic is always time-dependent on work-time or holidays.

2.2 Passive versus Active

Beside on analysis based on gathered data only, the other way is the use a methodology which also produces traffic, an *active* approach. It is clear, that for traffic classification the only way to use an active approach is to produce traffic and to classify this traffic later. This technique is also used to verify classification results in application detection or host classification.
For application detection or even for understanding an application, active probing brings in new possibilities. Some newer examples are [5] and [59], where Skype™[61] is investigated in a active way. An other example is [42], where they try to circumvent BitTorrent's robustness against selfish peers, which are trying to download more than than they fairly share with other peers. A hybrid approach is [68], where first a passive methodology is used to catch P2P hosts, and to validate the results of this heuristic, an active polling is adopted.
From our point of view, there is much unused potential in mixing passive approaches with active ones. If this is as in [68] just to verify the results from the passive analysis or to enhance the passive approach to get more specific information about e.g. webservers or super-nodes in P2P systems.
Also for host classification, the possibilities from combinations of active and passive analysis are rather unused until now. It is clear, that it would be difficult to handle an active traffic analysis system which can perform active probing for e.g. many applications in a big network, because such a system should not produce to much traffic.

2.3 User Behavior versus Machine Behavior

To detect something, one have to know what this something is. If you look for a certain bird in a forest, you have to know how a bird does look like. It would also be better if you know, that a bird can fly and does not life under the floor. So, you should know the behavior of this bird, e.g. how he wheezes. The same is it with application detection. One should know as much as possible about the applications to be able to detect them. Often such knowledge is more or less available, because the application is open, or because somebody has already written some work about a certain application, e.g. for Skype™in [5] or in [59].
Such a behavior of a specific application from the view point of the network layer is normally difficult to describe. Often it is easier to define a behavior for an application group, e.g. DATABASE, or MAIL applications. It is clear, that a description of an application should consist of attributes which can be measured. If you look for a certain bird, you also can only use attributes of the bird which you are able to detect. For a bird this might be color of his plumes or the dimensions of his body, etc. But if you only know the number of plumes or the average hemoglobin value of his blood, you probable will not find the bird. It is similar for application detection.
But what is application behavior? In the existing work are many examples what can be counted as application behavior. The simplest behavior is of course the port number which is used by an

application, this attribute of an application can easy be measured and in the past, it was enough information to clearly identify the traffic producing application. But **here** lies the main problem of application detection: You have the collected traffic data, which gives you many attributes which maybe can be used to identify an application (see [47]), but many from these attributes do not help, because their range is to big or to small and their distribution carries to less information.
On the other hand, you have the applications you want to be detected. One should clearly assign to each application some attributes, which made it possible to distinctly identify this application. This is much more difficult, than it seems to be. Since there are many not open applications for which exhaustive studies have to be done to get such attributes. Also, the Internet traffic composition is changing very fast. Every day new applications or newer versions of existing ones emerge which might be targets for application detection. A huge effort would be needed to have an updated list of attributes for every application one wants to detect. A good example of one of the most “wanted” applications are the popular P2P applications. Attributes for this class of applications are the facts, that a host participating a P2P system usually is connected to many other hosts or he tries it at least, or that such a host often produces approximately as much traffic as he consumes, or that many P2P systems transmit the files split into blocks with a fix size. There are many other attributes which might be helpful for a detection of P2P systems, but they are usually not equally adequate for every P2P system and because a main task to the designers of new P2P systems is to hide their traffic, such attributes change frequently.
Application behavior is not only dependent of the design of an application. The human using an application plays of course also his role. The impact of the human behavior is not to underestimate. Therefore, we divide application behavior into two parts: *user behavior* and *machine behavior*.
We propose two definitions of *user behavior* and *machine behavior*: First we define user behavior as actions which are performed by a human being and not by an automatism. Of course it is difficult to draw the boundaries between these two behaviors: An automatism is normally started and configured by some human and therefore also an action performed by a human. But there we draw the borderline. The action of setting up an automatism or configuring an automatism is clearly user behavior. But afterward, when the automatism is running, this is machine behavior. Also starting, stopping, reconfiguring, restarting, etc. are user behavior. Of course, there are automatisms controlled by other automatisms, which could also be implied, controlled, etc. by other automatisms and so on. But at some point there is normally a human which is responsible for all these automatisms and would therefore control them.
Second, we define user behavior as all behavior which are affected from actions performed by some human being. From the view of network traffic, this means that all possible differences, depending on a configuration by a human, between traffic produced by the same applications are user behavior and all other attributes, which are not affected by humans are machine behavior.
A simple example is POP3 traffic. With the first definition, only the set up and afterward the starting and closing time of the email-applications are user behavior. After starting, the email-application automatically checks with a fix interval for new emails. Also the sending of an email is user behavior if this happens when the send button was pushed. But if the email is sent within a regular send-receive action after the predefined interval, this is machine behavior. Of course the byte distribution over the periodical send/receive actions is defined by user behavior, since the user defines when he writes an email which will be sent with the next periodical send/receive action. With the second definitions, more attributes are affected by user behavior: The port numbers, the length of the intervals between two send/receive action, the encryption, etc.
To our best knowledge no previous work about traffic/host classification or application detection considers the differences between traffic generated by user behavior and machine behavior according two one of our two definitions.
Beside this two simple definitions, there are other aspects of user behaviors, which could be used mainly in host classification. Humans often visit the same or similar web pages, which introduce certain long- or middle-range dependencies. Examples could be user profiles for search engines or for BitTorrent the combination between website were the *.torrent* files were downloaded and the connectivity within the users of BitTorrent or simply with the amount of traffic consumed and produces by these hosts.

2.4 Grade Of Automatism

Whenever a new mechanism was introduced by science, there was normally still much work to do until a functional, practical, easy learnable, handable and affordable version of this mechanism was widely used by the people or enterprises. All steps of network traffic analysis should be automated until a product can be released which enterprises and Internet service providers can easy deploy and use. Of course the collection and the storage of traffic data is automated long ago. Also the aggregation of the gathered traffic in e.g. flow data is automated since the first implementation of a flow collector. Other points to consider can be e.g. a pre- flow-sorting procedure, because the flows are not sorted by start-time since they can not be exported until they finished. Also a kind of pre-filtering of the gathered data might be necessary, e.g. neglect all flows below a certain size etc.

2.4.1 Traffic Classification

In this field, the measurements are easily automated and also the evaluation of the results can be made without human interaction. The summary of the measurements can be written hourly, daily, weekly or monthly into some file and some plots can be generated With predefined thresholds, alerts can be released to inform the person responsible for the network. Here it is not so complicated to automate to whole process, because no additional information is processed, accept the information contained by the collected traffic. So, only the set up of of all parameter for this process has to be done by hand. Depending of the traffic mix and the dynamic of the traffic attributes, these parameters have to be adjusted from time to time. An automated solution is to define the thresholds for these parameters depending on the distribution of the parameter, e.g. three times the mean plus the variance of the size of a flow, etc.

2.4.2 Application Detection

It is easy if the classical port number based approach is used to almost fully automate the traffic classification. The only problem beside the accuracy is that with new applications or newer versions from existing applications, the list linking the port numbers with applications has to be updated. But this update should easily sold as periodical service by the software producing company.
It is more complicated if payload inspection is used: More system resources are needed and just the collection and storage of such a system would be more complex. Also the reference data which is compared with the collected traffic has to be up to date and also region dependent, because the traffic from Japan or China is different to traffic from the U.S. or Europe. Depending on the aspired accuracy, there are every week new applications or versions of existing applications where the payload has to be extracted and delivered to the administrator of such a network analysis infrastructure. But here also, this can be solve with periodical updates. Generally, it can be said, that for every approach, there is the problem to encounter any new or changed application.
Even if the problem seems to be solved with unsupervised or semi-supervised machine learning, the traffic from detected previous unknown applications appears in new cluster and must still be manually to assign to the application. An other problem when some approach with machine learning is applied, is the selection of an adequate feature set. Depending on the traffic composition, such a set has to be change from time to time. Here, some feature selection algorithms like the backward greedy feature selection promise some help. But the selection of the input of such an feature selection algorithm stays again unsolved. Another point is the detection of such a necessity to change something like a payload signature or a feature set for machine learning. Even more complicated are the approaches which use some transport layer heuristics, which can be very sensitive to newer versions of an application.

2.4.3 Host Classification

For Host Classification, there are even more tasks to automate. Depending on the classification procedure, if an approach related to applications running on a host or not, there are the same points as above. Additionally, there is the problem to update the classification results for the

observed host periodically under certain circumstances. Depending on the traffic dynamic and the behavior of the watched hosts, the results are only accurate for a few minutes. Furthermore, if one got a profile of a host, it has to be exported and be processed. E.g. for billing purposes or some security operations have to be performed. Also if the behavior of a host suddenly completely changes, some reaction should be predefined, because probably an other, perhaps hostile, user has got control over this host.

Chapter 3

Related Work

All work done in the field of traffic analysis can be separated within multiple dimensions. In this chapter some classification of the existing approaches is presented and evaluated. Some traffic analysis usually starts with gathering data from a network device. This is also the point where the various approaches firstly differ in.
Generally traffic classification and application detection can be split up into passive and active attempts. Active approaches usually try to get some information about one certain application or one certain group of akin applications. The goal of passive approaches is normally to get finally a better knowledge about all application.

3.1 Active Approaches

When a new obscure application becomes (very) popular, sooner or later there is some research which tries to expose this nescience. Often, this could only be done if the scientists play some active role in the context of this applications. Some examples are the studies of Skype™ in [5] and [59]. [5] describes the key components and functions of Skype™. They interact with other Skype-nodes and additionally they check the payload. In [59] some different measurements for bandwidth consumption, node and user behavior are presented.
In [52] the pure Peer-to-Peer (P2P) communication application Winny, which was the most popular P2P application in 2005 in Japan, is observed using two methods. First, they tried set up a decoy peer which collected all pairs of IP addresses and the service ports of all possibles peers. Second, they collect the same information by letting the service ports of the Winny peers playing the same role as the decoy in method one. In [42] the manner of BitTorrent against selfish peers is examined and ways to elude thees are presented. There was much work done in similar to these mentioned papers.
The drawback of such active approaches is, that they normally treat only one specific application or protocol. At most, they deal with a group of application with are similar to each other. Neither they are able to react to some changes without human intervention nor they can cover some new application in the same group.

3.2 Passive Approaches

There has been much work in this appendage of traffic classification. Overall, there are four different techniques to get some information about the traffic and the application running over some link.

3.2.1 Port Number Analysis

This is the classical method to classify Internet traffic. Because many traditional network applications run on well-known ports, assigned by IANA [27]. For example, [60] analyzed the behavior of three popular P2P applications after identifying them by port number. They were interested in topology characterization, traffic characterization and dynamic behavior characterization. Also

in [24] the well-known port numbers are used to assign the applications to the traffic they examined. In [32] this approach has shown to be ineffective, because some newer applications, for example the current generation of P2P applications, try to hide their traffic using dynamic port numbers or hiding behind well-known applications. Unfortunately these are the applications which generate huge traffic volume.

3.2.2 Payload-based Analysis

To enhance the flaccidities of the approach above, payload analysis was used. This method is well researched i.e. in [32], [63], [31], [55], [46] and [63]. Here are the disadvantages in the fact, that first, unknown traffic or unknown applications can't be detected. Second, there are some new applications as BitTorrent (also newer version of Kazaa) which encrypt their traffic. Third, payload analysis brings in some privacy and security concerns. Finally, these approaches normally need increased storage and processing resources [18].

3.2.3 Transport-layer Heuristics

Transport-layer heuristic information is used to confront the drawbacks of the approaches from the sections 3.2.2 and 3.2.1. In [3] the flow characteristics are analyzed using i.e. arrivals, sizes, durations. Not interested in applications, they just wanted to classify the traffic with these flow attributes. For P2P applications there are some new methods in [32], [11] and [4]. In [32] the unique behaviors of some P2P applications are used when they set up their connections or transfer the data. This technique is shown to perform better than port number based classification and equal to the payload based. In [11] the network diameter and the specific host behavior are used to identify known and unknown P2P traffic. In [4] the make use of the inherent behaviors of the hosts to the P2P traffic. [12] uses the size of the IP Packets, their inter-arrival time and the order of the packets. In [55] they present a method for automated constructions of application signatures.
For all these approaches there is a need for much information about the specific applications, therefor it would be difficult to expand some of these methods to detect more different applications as Internet-games or chat applications. In addition, there is a different approach [33] ([34]), called BLINC, where social, functional and application behaviors are used to identify all types of traffic in dependency to the hosts.

3.2.4 Machine Learning Approaches

Normally machine learning approaches consists of two parts: model building and classification: Some model is built from the training data and then this model is putted into some classifier which finally classifies the data.
Machine learning techniques can be divided into the supervised and unsupervised techniques. The interarrival-times and size attributes of a flow are used in [44] to group the flows with the EM-algorithm in a unsupervised approach. In [76] they used the AutoClass algorithm. In [17] k-means and DBSCAN and in [15] vector quantilisation and Bayes using Gaussian Mixtures are used for unsupervised learning. To be fast, they apply the K-Means algorithm on the first five packets in [6]. Some supervised techniques also make use of some connection-level statistics. [58] uses nearest neighbor and linear discriminate analysis. In [45] Naïve Bayes is used as classifier. [20] uses a new semi-supervised learning algorithm.

3.3 Presentation of existing Work

Here, some examples of the existing work are presented and evaluated within the mentioned dimensions. Of course not all existing work in traffic analysis is covered, but to the best of our knowledge we selected the ones which brought in some new methodologies or which stand out for other reasons.

3.3.1 Analyzing peer-to-peer traffic across large networks

In [60], the focus lies on topology and traffic characterization and also on the dynamic behavior of the three popular P2P systems Fasttrack [36], Gnutella [25] and DirectConnect [14]. Using *netflow* [50], the three P2P systems are identified by their port numbers. This identification method is adequate because the datasets are from 2001, where the usage of not-well known port number is not common yet. For each P2P system, host distribution, traffic volume, host connectivity, traffic pattern over time, connection duration and on-time and mean bandwidth usage are observed. Traffic Data is collected from September to December 2001, for each month one week. For the host distribution, the results are as respected: For each P2P system there are more host from month to month. Fasttrack has about 0.5 million to 2 million IP addresses each day, this is about 5$\sim$7 times that of Gnutella and 150$\sim$300 times that of DirectConnect. For the traffic distribution, they observe that the top 1% of the IPs, prefixes and ASes transmit 73%, 64%, 73% of the total traffic, respectively. They also inspect how many host are in one of the P2P system during different time intervals and how long these hosts are active hosts in one of the three P2P systems. Here, the result is that 65% of the hosts of FastTrack join the P2P system only once and that the average uptime of 60% of the host is only 10 minutes each time they join FastTrack. Also up- and downstream bandwidth are observed, with the result, that the bandwidths are limited by the network infrastructure because many hosts have a asymetric bandwidth structure, where the downstream bandwidth takes almost 100% of the available bandwidth. Finially, the correlations between the three metrics traffic distribution, host conectivity and mean bandwidth are calculated and compared.

3.3.2 File-sharing in the Internet: A characterization of P2P traffic in the backbone

To detect P2P traffic on nonstandard ports a set of heuristics is developed in [31]. They empirically derive their methodology studying different P2P protocols. Their approach is payload-based, and therefore P2P traffic on all ports can be detected. But it has its limitation with encrypted or byte-stuffed traffic. Two way are found to identify different P2P traffic. *Packet size*, where the control packets of each P2P system can be identified by investigating of the first few bytes. And *block size*, where most P2P systems use fixed block sizes for data transfer. The following P2P protocols are observed: eDonkey2000 [16], FastTrack [36], Gnutella [25], Napster [49], BitTorrent [10], DirectConnect [14]. Additionally, empirically identified rules for each P2P protocol are used to reduce the false positives. The results were: eDonkey 2000, FastTrack and Napster are the most popular P2P systems. In Europe, eDonkey2000 is more popular than in Asia. The newer P2P protocols BitTorrent and DirectConnect gain an increasing part of the overall P2P traffic. For the dataset examined in this work, the total P2P traffic increase 2.5% for packets and 1.5% for bytes from August 2002 to May 2003. In 2002, with exeption of eDonkey 2000, there is no traffic nonstandard port numbers. In 2003, with exeption of Napster and BitTorrent, all protocol use arbitrary port numbers. Port 80 usually appears as destination port to to by-pass firewalls which do not filter web traffic normally.

3.3.3 Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures

In [63] a real-time approach classification system is designed based on the first few packets of a P2P connection. This approach works in the middle of a network and is robust against asymmetric routing and packet losses. They focus their work on TCP data traffic of the five most popular P2P systems of 2003 (*Kazaa, Gnutella, eDonkey, DirectConnect,* and *BitTorrent*). By manual inspections, for each P2P systems an application signature is developed. This signatures consists of some bytes, words or string which are used in a fixed pattern match at a fixed offset for the different P2P systems. This operation is relatively trivial, so the complexity in the worst case is the length of the pattern. Because this approach needs payload data, it would not work for P2P systems using encryption or if byte-stuffing is used in a newer version of a client from one of these P2P systems. With their trace, they have no false positives and compared to a port based methodology, they have a false negative rate of 5% to 10%. For Gnutella, Kazaa

and DirectConnect 34.5%, 72.6% and 38.8%, respectivly, of the traffic detected are on non-standard ports. For eDonkey only 4.1% of the detected traffic are on non-standart ports. But for BitTorrent, the signature-based approach detects less traffic than the port based methodology. And only 0.95% of this detected traffic are on non-standart ports. This, because the use of non-standart ports is not widespread at this time for the BitTorrent P2P system. To evaluate their approach, the results are compared with results from a port-analysis, althogh the drawbacks of this technique is the motivation for their work.

3.3.4 Transport Layer Identification of P2P Traffic

To circumvent the disadvantages of a payload-based approach, [32] presents a methodology based only on transport-layer heuristics. Additionally a payload-based detection is implemented to compare the results. They identify 99% of all P2P flows and more than 95% of all P2P bytes, compared to the payload analysis. Their work detects P2P traffic which can not be found with payload analysis, because of encryption and unkown P2P protocols. They also get three new P2P systems previously unkown to them. This payload analysis is similar to [63] but instead of five, they inspect nine P2P systems: eDonkey, Fasttrack, BitTorrent, WinMx [74], Gnutella, MP2P [48], Soulseek [62], Ares [2] and DirectConnect. And here, also UDP packets are inspected. The transport-layer heuristics are only based on observing source and destination IP addresses and port numbers. The first heuristic is that if a IP pair uses both UDP and TCP, it is concerned as P2P. Six out of the nine P2P protocols use both, TCP and UDP. To reduce the rate of false positives, flows from other applications using UDP and TCP are removed. Examples are DNS, mail servers, gaming, mailware, IRC, etc. The second heuristic consists of monitoring connection patterns of {IP, port} pairs. If a host communicates with one source port to distinct peers, it recieves data from these distinct peers on distinct ports, it uses a different port number for each peer. So, if the number of distinct peers is equal to the number of distinct ports, the {IP, port} pair is considered as P2P. If the difference between IPs and ports is more than e.g. ten, it is considered as non P2P. With these heuristics, after reducing the false positives, 20% to 100% more P2P traffic was detected than with the known port numbers analysis.

3.3.5 Application-Level Traffic Monitoring and Analysis on IP Networks

In [37] a new approach is presented. First the categorize the Internet applications into classes according to traffic patterns. Second they develop a flow grouping method to assign the traffic flows to an application. The category of an application depends on how many sessions a application uses between to hosts, how the port numbers are selected and between how many hosts the communication occurs. There are five different categories:

- Type S-F-2 (S for single session, F for fixed port numbers, 2 because the communication of an application from this category takes place between two hosts)
- Type M-F-2 (M for multiple session)
- Type M-D-2 (D for dynamic port numbers)
- Type M-F-3 (3 because the communication of an application from this category takes place between three or more hosts)
- Type M-D-3

For applications belonging to the categories with fixed port numbers, the traffic classification is easy. This is more difficult for applications of the categories with dynamic port numbers. Here, this task is solved with the three mechanisms application port table (APT), important port selection (IPS) and the flow relation map (FRM). In a first step the APT is constructed. For more than 100 of the most popular applications the default port numbers (TCP and UDP) and other informations were collected and the application was categorized in the way S/M-F/D-2/3. Also the most frequently used port numbers (TCP and UDP) in addition to the default port numbers were stored in the ATP. The IPS selects for each flow the important port, either the source or destination port. The FRM consists of two consecutive unities: The property dependency grouping (PDG) and the location dependency grouping (LDG). The PDG groups flows with

dependencies over port numbers, protocol numbers and source or destination addresses into PDG groups. These PDG groups are again grouped by the LDG. The association between all flows of two PDG groups is weighted and than sum of these weights builds the weight of the association of the two groups. If the maximum of this weight over the number of flows from one of the two PDG groups is bigger than some threshold than these tho PDG groups form a LDG group. After grouping al PDG groups to LDG groups, the final LDG groups can be associated with an application due the APT. In this way, with 500 LDG groups, 99% of the flows, 95% of the packets and 92% of the bytes were classified.

3.3.6 Class-of-Service Mapping for QoS: A statistical Signature-based Approach to IP Traffic Classification

[58] puts the popular applications into four classes: *Interactive* (Telnet), *Bulk data* (FTP-data, Kazaa), *Streaming* (RealMedia streaming), *Transactional* (DNS, HTTPS). After a discussion about candidate features for the classification, average packet size and flow duration were selected and tested with *Nearest Neighbor* (NN), *Linear Discriminant Analysis* (LDA) and *Quadratic Discriminant Analysis* (QDA). For the training data, six applications were used: DNS, FTP-data, HTTPS, Kazaa, Realmedia and Telnet. For cross validation, each dataset available was divided into ten pieces and nine of them were used as training data to classify afterward the last piece. The errors were always below 10%.

3.3.7 BLINC: Multilevel Traffic Classification in the Dark

In [34] three levels of traffic description are investigated: the social, the functional and the application level. This approach works in the dark, with no access to packet payload, port numbers and without any other information other than a current (2005) flow collector can provided (**BLIN**d **C**lassification). The goal is not to assign the flows to application, but to classify the hosts. At the *social level* the popularity of a host is captured, that means with how many hosts communicates and what is the function of these hosts, are there communities of hosts, etc. For this level, the informations about only source and destination IP addresses are used. At the *functional* level the functional role of a host in the network is analyzed. Is a host a server, does a host provided some service on certain port, is he only a consumer or is he a part of a P2P network, etc. At this level, the characteristics of source and destination IP addresses and the source port numbers are needed. At the *application level*, the transport layer interactions are analyzed. First only the 5 tuple (srcIP, dstIP, srcPort, dstPort, protocol) is used to classify the flow. Second, other heuristics, like bytes and packets transferred, are processed to refine the result. For each application a pattern is empirically constructed, which is represented by a *graphlet*. With a collection of these *graphlets* the behavior of a host is described. At the social level, a community of hosts will appear as *bipartite clique* [34] of IP addresses. If such a clique is “perfect”, this is a hint for malicious flows. There are also “perfect” cliques for gaming communities, but most “perfect” cliques are from worms or DDoS. If there is only a partial overlap, this case correspond to collaborative communities like P2P systems or gaming groups. But also to hosts connecting to the same services at the same time (browsing on the same web server or streaming). If there is only a partial overlap, but within the same domain, this leads to a service “farm”. Only the last few bits of the IP addresses are different, so a service is provided by a multi-machine environment. Examples are mail, web or streaming servers. For the functional level, they made three groups: typical client-server behavior, typical collaborative behavior and Obscure client-server behavior. For the first group, the classical example is web traffic, for the second group clearly P2P and for the third group mail traffic and FTP. In the application level, the knowledge of the two other level is combined. Here each empirically defined *graphlet* represents an application or a class of applications. Additionally the relative cardinality of dstPorts versus dstIPs, the average packet size per flow and other heuristics were used to distinguish more exactly between the different applications. Overall, BLINC classifies 80%-90% of the flows with accuracies form 90%-95%, depending on the datasets.

3.3.8 ACAS: Automated Construction of Application Signatures

In this payload based approach [55], the following applications are detected: FTP control, SMTP, POP3, IMAP, HTTPS, HTTP, SSH. The first *n* bytes of a flow are used to construct a *feature vector*. The feature vector has $256 * n$ elements, all initialized with 0. Then, for each byte *i* of the *n* bytes, the the component $i * 256 + c[i]$ is set to 1, where $c[i]$ is the value from this byte. So, each feature vector has *n* non zero elements. These feature vectors are used for three machine learning algorithms: Naïve Bayes, AdaBoost and Maximum Entropy (or Maxent). These algorithms are trained with 1, 4 and 8h datasets to compare the dependencies between training length and error rate. Also for *n* different values were used: 64 bytes and 256 bytes. To evaluate their experiments, the default port numbers of the applications are used. The results are better for all applications if the training sets become longer. But, the results are not better for every application if *n* is bigger. For SMTP the error rate is bigger if the first 256 bytes are taken to build the feature vector. The error rate for the other application decreases as expected. Overall, they reach a flow accuracy above 99% for all applications.

3.3.9 Toward the Accurate Identification of Network Applications

In [46] a full payload packet trace based approach is presented. This method combines nine different layers of classification, where each layer is more complex and has a better accuracy. Before they start the classification, all *simplex flows* are removed from their data sets. A simplex flow is a connection between two host where only one host is sending and the other host never replies. The first layer consists the classical port number based classification method. The second layer accesses to entire packet header of both traffic directions. Layer three to eight check if a flow has a well-known packet signature or if a flow follows well-known protocol semantics. The third layer looks if there is a known signature compared with one packet. The 4. layer looks if there is already visible hint for a protocol in one packet. The 5. layer does the same as the third, but for 1 KByte, also does the 6. for the 4. layer. If a flow is not classified correctly by a layer, this flow is given to the next layer, until all layers has not correctly classified, or one of them calculated the correct result. The correctness of a classification is given by the 9. layer, which consists of a host/port history, which represents the knowledge of all classifications in the past. If no layer is able the calculate a matching result with the last layer, this flow is saved for manual inspection. The traffic is categorized in the following groups: BULK, DATABASE, INTERACTIVE, MAIL, SERVICES, WWW, P2P, MALICIOUS, GAMES, and MULTIMEDIA. With the classical port based approach 28% of the packets and 30% of the bytes stay unknown. With the methodolodgy presented here only less than 0.01% of the packets and the bytes stay unknown.

3.3.10 Characteristic analysis of Internet traffic from the perspective of flow

The main focus of [38] is on *flash flows*. A flash flow is a flow whose duration is less or equal to one second, the number of packets of such a flow is no greater than three and the amount of bytes is less or equal to 500. First a high level analysis on the flows is performed. Traffic distribution over time, distribution of the flow duration and byte/packet distribution within the flows are investigated. 99% of the traffic was either TCP or UDP. On average, there were 642 bytes per packet, 678 bytes per TCP packet and 239 bytes per UDP packet. There were 2.7 times more UDP flows than TCP flows, but 90% of all packets and bytes are running over TCP. Therefore, UDP is more important from the perspectives of flow. The average flow duration is 57.32s for TCP and 10.72s for UDP. The standart deviation of the flow duration for TCP and UDP was 540.84s and 76.62s, respectively. 60% of all flows are shorter than 10s, 20% of the TCP flows and 80% of the UDP flows. Also have 53% of the UDP flows only 64b, which means that half of the UDP flows consists only of one packet. Within the three dimensions of flow duration, bytes per flow and packet per flow 59.3% of all flows were flash flows. Of the TCP flows, 7.7% and of the UDP flows 76.8% were flash flows. Most of the flash flows are P2P or Dos/DDos traffic. For TCP, the thirty most used port numbers contribute 90.6% of the TCP traffic and for UDP the thrity most used port numbers contribute only 65.85% of the total UDP traffic. From

these thrity TCP ports, only port 80 was from the standart port from IANA [27]. From the UDP ports, only port 53 (DNS) was registered on IANA.

3.3.11 Identifying Known and Unknown Peer-to-Peer Traffic

[11] needs only application independent transport layer information which is specific for P2P behavior of a host. First they inspect the diameter of a network. All hosts are initiated with level zero if they first occur in a dataset. If there is a connection from a host which has already a level to a new host, than the new host gets the level from the existing host plus one. If a unknown host initiates a connection to a host already being in a network, the new host receives the level from the older host minus one. If the diameter of such a network is greater than the diameter (three), than this network (and its traffic) are marked as P2P. Second, the behavior of a host is monitored. If the consumed and produced amount of traffic is more or less equal, a host acts as server and client, which is typical for a P2P hosts. The results were evaluated with a classic port based classification system, because no payload traces were available. Some port numbers were excluded from the port based classification scheme, because they carry both P2P and other traffic. About 10% of the P2P traffic detected with the port based classifier was not found with the two heuristics presented in this work. And on average, 20% of the detected P2P were false positives.

3.3.12 Self-learning IP Traffic Classification based on Statistical Flow Characteristics

In this short work [76], they use the packet inter-arrival time, the packet length mean and variance, the number of bytes per flow and the flow duration to classify the traffic with machine learning. A flow is here bidirectional with a timeout of 60 seconds. As machine learning algorithm AutoClass is used, a implementation of the Expectation Maximization (EM) algorithm. This unsupervised Bayesian classifier builds the ‘natural’ classes inherent, which can afterward be used to classify the flows. Because the learning process was slow, they only used 1000 randomly sampled flows from eight know destination ports (FTP data, Telnet, SMTP, DNS, HTTP, AOL messenger, Napster and Half-Life). *Sequential forward selection* (SFS) was used to select the best attribute set. This algorithm starts with each attribute and first selects the best attribute, than this attribute is tested with each of the remaining attributes, and the best pair is selected. And so on, until no further improvement is achieved. For validation, they assumed, that the port number defines the origin application of a flow. As stated at the begin of this work, this is not always correct, but they assumed also, that this is accurate enough (!). With this methodology they get an average accuracy of 87% (94% for Half-Life, but only 74% for HTTP).

3.3.13 Flow-based Identification of P2P Heavy-Hitters

[68] demonstrates an algorithm called *PeerTracker* that identifies P2P host based on Cisco NetFlow informations [50]. A prototypical implementation of the *PeerTracker* currently runs at the central network services of the ETH Zurich. All measurements were done with data from SWITCH [65], a medium sized Internet backbone in Switzerland. The IP address range of SWITCH contains about 2.2 million addresses. In 2004 on average 60 millions NetFlow records were captured per hour. The “default” port numbers of a P2P system are used to detect if a host is a P2P host or not. Even the newer P2P systems use dynamic port numbers, from time to time there rests still some traffic on the default port numbers, e.g. for communication, or connection set up. When a network connection is detected between two hosts, each host becomes a *candidate peer*. A *candidate peer* which has more P2P traffic becomes a *active peer* otherwise it becomes a *non-peer* if it has no P2P traffic for the probation period (900 seconds) and will be deleted. Every *active peer* is monitored for further P2P traffic. After a maximum time (600 seconds) without P2P traffic, an *active peer* becomes a *dead peer*. Each *dead peer* is still monitored for P2P activity, but not reported as active anymore. After another maximum time (one hour) without P2P traffic, a *dead peer* is deleted from the internal state of the *PeerTracker*. During every measurement interval (900 seconds) the last 100 used local and remote ports were stored together with the respective amount of data for every host. With a threshold (different

for host within the SWITCH network and host outside) value for the data, it can be determined to which P2P network a host belongs. In a thirty day measurement, 2982 P2P host could be identified within the SWITCH network. The usage of the default port numbers was investigated and also the distribution of bytes transfered within the different P2P networks. Additionally, to reduce the number of false positive an active polling approach was used.

3.3.14 Internet Traffic Identification using Machine Learning

In [18] different machine learning algorithms are compared. The supervised Naïve Bayes and the unsupervised EM clustering method called AutoClass. The data analyzed consists of on 72 hour and one 24 hour traces from the University of Auckland. For each flow seven features, e.g. total number of packets, mean packet size (in each direction and combined), flow duration etc, are used to classify the traffic into nine classes like SMTP, HTTP, DNS etc. Because the two traces are from 2001, classification based on port numbers is used as base truth. This is thought as accurate, because the emergence of dynamic port numbers in P2P systems is later. Three metrics are used the measure the precision of the two approaches: *precision*, *recall* and *overall accuracy*. Precision is the ratio of True Positives to True and False Positives. Recall is the ratio of True Positives to the True Positives and False Negatives. And overall accuracy is the sum of all True Positives to the sum of all True Positives and False Positives for all traffic classes. The AutoClass algorithm outperforms the Naïve Bayes with over 90% against circa 82% in overall accuracy. For the Naïve Bayes, the recall and precision are over 80% for six out of the nine classes. The precision and recall for the AutoClass is placed around 90%. They conclude, that the unsupervised approach is at least equal in performance to the supervised algorithm and because of the advantage that the AutoClass is able to discover unknown applications, they would consentrate on this kind of machine learning algorithms in the future.

3.3.15 Bayesian Neural Networks For Internet Traffic Classification

[45] compares two supervised machine learning algorithms. These are the Naïve Bayes and a Bayes Neural Network. Two 24 hour data sets from an academic campus with circa thousand users are available. The first containing 337 thousand flows and the second, eight months later with 175 thousand flows. Both sets are from work-week days. The aim is to classify these flows into ten classes, e.g. BULK, DATABASE, P2P, MAIL, ATTACK, etc. To get a base truth, the flows are manually and semi automated classified into one of these ten classes based on the payload. The get the features, they make extensively use of the tcptrace tool [67]. The complete list of all the features is presented in [47]. The computation of all the 246 features for all the 553,188 flows takes over 40 hours. After some investigation, the FFT features were removed and then the computation of remaining 226 features needs 40 seconds with a standard deviation of 22. Because some of the values of the features are integers and others are floating point numbers, every feature is mapped to a value between 0 and 1. So unknown values are 0.5. For the Naïve Bayes, the accuracy suffers from features with irrelevant or redundant information, therefore a Fast Correlation-Based Filter (FCBF) is used the reduce the number of features to a set of the ten most valuable features. The Naïve Bayes and the Neural Network are compared with three experiments. 50% of the data from each day are used as training data and the other 50% are used to test. The Neural Network reaches over 99% in metrics of flow, packet and byte accuracies. Where the results of the Naïve Bayes are located between 74% and 95%. In the third experiment, again 50% of the data from the first day are used as training data, but the test is done with the complete trace of the second day. Again, the Neural Network outperforms the Naïve Bayes. Getting about 95% accuracies the results of the Neural Network are the headline result of this paper, because in this near realistic situation are the training and testing data sets eight months apart. Furthermore, the length of the training data is varied and the number of features for the Neural Network is reduced down to twenty with a feature-interdependent ranking, where these are not the same features as with the FCBF, only one feature is in both sets. This field is also left for future work.

3.3.16 A measurement study of correlations of Internet flow characteristics

[41] studies the heavy-hitter flows in four different dimensions and the correlations between these dimensions. Before it existed many different classification schemes for such flows. By **size** as elephant or mice flows, by **duration** as tortoise and dragonfly, by **rate** as cheetah or snail and by **burstiness** as alpha or beta traffic. These definitions are not clear and the relation between them are never investigated. First, the different characteristics are defined. An *elephant* flow is a flow with a size grater than x KByte and a *mice* flow is a flow with size smaller than x KByte. A *tortoise* flow is defined as a flow with a duration longer than y min and *dragonfly* as a flow with duration less than y min. A *cheetah* flow is a flow with a rate greater than z KByte/s and a flow is *snail* if it's rate is less than z KByte/s. x, y and z are defined as the mean plus three standard deviations of all flows in the respective dimension. For burstiness, they deliver three different definitions: The first is based on the variation of traffic at a timescale T. A flow is divided into bins b_i of duration T. s_i is the number of bytes sent in b_i, then *variance burstiness* of that flow is the standard deviation of all s_i. The disadvantage of this definitions is that the variance is undefined for flows shorter than T and that boundary effects add error for flows shorter than 3-5 T. Second there is the *RTT burstiness* which is the mean of the bytes sent in every RTT multiplied with the average RTT. This definition avoids the drawback of definitions with a fixed timescale. Their third definition is *train burstiness*. Here a burst is a train of packets with an inter-arrival time less than t. Burst size is the number of bytes sent during each burst. Burst duration is the time between the first and the last packet of a burst and burst rate is the is the burst size divided by burst duration. Inter-burst is the time between two bursts. Finally, *train burstiness* is the mean of the burst rates of a flow multiplied with the mean of the inter-bursts. So *porcupines* are flows with a burstiness greater than m KByte and *stingrays* as flows with a burstiness less than m KByte. m is also the mean plus three standard deviations of the burstiness of all flows. They show, that size and burstiness are strongly correlated while size and duration are rather independent of each other. It is also shown, that if somebody is paying attention to the bursty flows, one could catch most of high-rate and large-size traffic. Burstiness can also be used as an other metric to identify elephant flows instead of size and duration. But using the duration as indicator of the volume of the traffic can sometimes be misleading. Flow size and duration might need to be handled as different and independent dimensions. For future work they left the classification of the protocol timeout mechanisms due application/user behavior or network congestion. These timeouts elongate the flows and might have a big impact in performance in case of a retransmission of some packets. It would be interesting to know which part of the timeouts are caused by e.g. network congestion, software flaws, malicious attacks.

3.3.17 Traffic Classification Using Clustering Algorithms

[17] is using two unsupervised clustering algorithm: K-Means and DBSCAN, that has previously not been used for traffic classification. They compare the results also with the AutoClass algorithm. The advantage of unsupervised machine learning over supervised machine learning is, that also previously unknown classes of traffic can be detected and classified. These algorithms are selected, because they base on different clustering principles. The K-Means algorithm is partition-based, the DBSCAN (Density Based Spatial Clustering of Applications with Noise) is density based and the AutoClass algorithm is a probabilistic model-based algorithm. The K-Mean and DBSCAN algorithms are also selected, because they are much faster than the AutoClass algorithm. Also the number of clusters has an impact in the speed. If the algorithm uses only a few “good” clusters, the performance can be enhanced. The AutoClass and K-Means algorithms have both an accuracy of 85%, where the DBSCAN only achieved 75%. Although DBSCAN has a lower overall accuracy, it produces clusters most accurate because it can lable connections as noise. The following flow characteristics were used with Euclidian distance as metric: total number of packets, mean packet size, mean payload size excluding headers, number of bytes transferred (in each direction and the sum), and the mean packet iner-arrival time. They found, that because of the heavy-tail distribution of many of there characteristics and because of the Euclidian as metric, the logarithms of the characteristics give much better results for all the clustering algorithm.

3.3.18 Traffic Modeling and Classification Using Packet Train Length and Packet Train Sizes

In [15] the main focus lies on the use of the unsupervised learning algorithms Vector Quantization and Bayesian Classification using Gaussian Mixtures Models. The attributes for these algorithms are the packet train length and the packet train size. Where a packet train is essentially the flow between two hosts. They look only for the five traffic classes HTTP, SMTP, DNS, SSH and POP3. Data from 90 days are collected and data sets of 60 minutes, 30 minutes and 15 minutes are used for training. The results are a little bit better for the longer training, but only a few percents. More impact has the introduction of a threshold for the distortion of each traffic type which yields to improvements up to 20%. Overall the accuracy of Gaussian Mixture Models is 98.6% compared to 94.9% achieved with Vector Quantization based classification.

3.3.19 Traffic Classification On The Fly

The focus of [6] lies on early detection of the traffic application. The goal is to know the application after five packets. They find that the size of the first five TCP data packets is enough information to detect the application because the negotiation phases of the application are so different during the first packets. An unsupervised machine learning with K-Means is applied and they observe different numbers of packets with the conclusion that five packets brings the best separation of application among the clusters. They also test different numbers of clusters and get 50 clusters as the best combination with the five packets. So the flows are lying in a five-dimensional space, where each packet is associated with one dimension and the Euclidean distance between the spatial representation of the flows can be used as metric to evaluate them. They compare their results with payload measurements and get more then 80% of correctly identified flows. The only problem is POP3, where 86.8% of the flows are labled as NNTP and 12.6% as SMTP. They correct this weakness with the knowlegde of the destination port and also get an accuracy over 90%. They note several limitations and challenges whith their approach, e.g. out of order packets, packet sampling measurement environments, applications with similar or unknown behavior, short flows with less than five data packets, etc. The main problem is, that these technique could easly be evaded. The insertion of a few packets at the begining of a flow is enough and this approach would not recognize the flows anymore.

3.3.20 Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing

[4] identifies behaviors that are *inherent* to peer-to-peer (P2P) traffic. BitTorrent and Gnutella hosts are detected with these behaviors. From packet header traces of two days they achieve accuracy from 83% in detection of BitTorrent Host with a false positive rate of 2% and for Gnutella host a true positive rate of 75% with 4% false positives. They investigate the amount of data exchanged between the hosts, the rate of failed connections, the ratio of incomming-to-outgoing connenctions and the ratio of privileged-to-non-privileged port numbers. Because peers in a P2P system are usually end-user machines, it is normal, that they come and go frequently. Therefore, the automatism connecting to other peers is not working perfect and there are more faild attemps to contact other peers than with non-P2P applications, where the server receiving the request from the client is normally well known and persistent. An other differnece to some classic client-server applications is the fact, that there is a balance between incomming and outgoing connections. Both in number of packets and number of bytes. And because, P2P applications are typically user-level processes operating on a variety of platforms and environments using privileged port numbers. Meaning port numbers above 1024. Of course other applications, espacially gaming also uses these port numbers. With conbinations of these four inherent behaviors, they reach quiet a good detection performance, 75% of the P2P hosts in less than 10 ten minutes and 20% within a minute.

3.3.21 Flow Clustering Using Machine Learning Techniques

[44] uses packet header traces to divide the traffic into classes like bulk traffic, single and multiple transactions or interactive traffic. Unsupervised learning using Expectation-Maximization

(EM) constructs k "soft" [1] clusters according to a set of features. Some of them are packet size statistics, inter-arrival time. They also make use of attributes which were rarely used in other papers. These are the time spend idle, in bulk transfer mode and in transaction mode, or the number of transitions between transaction and bulk transfer mode. The also observed the distribution of the port numbers over their clusters and used this distribution as indicator for the applications within these clusters. It is concluded, that this kind of classification is exact enough for traffic class detection, but does not bring in the information needed for application detection.

3.3.22 Offline/Realtime Traffic Classification Using Semi-Supervised Learning

The main contribution of [20] is the introduction of *semi-supervised machine learning*. This has the advantage, that only a few sample flows are needed for the training sequence of the algorithm (K-Means). Compared to a supervised learning algorithm, the semi-supervised learning approach can detect new or changed applications. They have only a few one-hour traces but a good base truth gained with payload classification, transport-layer heuristics and HTTPS (port 443) inspection. Their classifier can be developed by iteratively adding new unlabeled flows to enhance the classifier's performance. They employ the backward greedy feature selection method and received following eleven features: total number of packets, average packet size, total bytes, total header (transport and network layer) bytes, number of caller to callee packets [2], total caller to callee bytes, total caller to callee payload bytes, total caller to callee header bytes, number of callee to caller packets, total callee to caller payload bytes and total callee to caller header bytes. They find, that features with a time component, e.g. duration or inter-arrival time, are not useful by the feature selection algorithm and that they should be avoided because of invariance within different networks. They achieve a flow and byte accuracy depending on the application between 60% and 90% both with the offline and realtime classification.

3.3.23 Traffic Classification through Simple Statistical Fingerprinting

Using the order, the direction, the size and the inter-arrival time of the packets [12] introduces the principle of *statistical fingerprinting*. The key difference from other work are: The use of *normalized thresholds* in the classification algorithm, the *smoothing filter* to reduce the noisy factors and the use of the order in which the packets arrive. Because the work is in a early stage, only three applications are tested, which are probably the most easy ones. With the four values from above, a Probability Density Function vector $P\vec{D}F$ is constructed. This vector describes a certain protocol, e.g. POP3. After applying a Gaussian Filter (reducing the "noise") to each component, they introduced the concept of a *protocol mask* $\vec{M}$. The next step is to calculate a *anomaly score S* which describes "how statistically far" a unknown flow is from a given protocol $P\vec{D}F$. With these two metrics the *anomaly score threshold T* is constructed. Finally, the protocol p 's fingerprint is defined as the union of the protocol p 's mask vector $\vec{M}^p$ and the protocol p 's threshold vector $\vec{T}^p$: $\Phi^p = \{\vec{M}^p, \vec{T}^p\}$. After the fingerprints of some protocols have been trained, the unknown flows can be assigned to one of the previous trained protocol or be labeled as unknown. The problem is, that this approach is side dependent. The fingerprints can only be used in the network, were the training has been made. For the three tested protocol they reached accuracies between 90% and 95% compared to manually inspection.

3.3.24 Profiling the End Host

[35] is the continuation of BLINC [34], [33]. But instead of to assign the traffic to some applications or groups of applications, the focus lies here in designing general profiles of end-host activity. Consecrating on dominant and persistent behaviors they try to neglect ephemeral actions because many new application make e.g. use ephemeral ports. They extend the concepts of the *graphlets* by introducing *activity graphlets* and *profile graphlets* which are a compressed version of the activity graphlets. Here a graphlet is a graph arranged in six (five in BLINC) columns corresponding to: srcIP, protocol, dstIP, srcPort, dstPort and dstIP. With the reduction

[1] Soft if the clusters are compared with some "hard" clusters from e.g. K-Means

[2] The caller is the host which initiates the flow and the callee is the host which reacts to the initial request

from the activity graphlets to the profile graphlets the needed negligence. Only the requested dominant and persistent behaviors stay in the graphlets. The classification of the hosts is not really evaluated and a lot is left for of future work.

3.3.25 Early Application Detection

[7] is the continuation of [6]. Here the intent is to classify the traffic as fast as possible and therefore only the first few packets of a TCP connection are observed (typically the first four or five). The TCP control packets (SYN, Keep-Alive or ACK with no data) are not considered, because they don't carry any application data. Also only the TCP connections which have at least four data packets are accounted. Because, they don't have much information, they use the size and the direction of the packets. With this features three well known clustering algorithms are evaluated. K-Means, Gaussian mixture model (GMM) and spectral clustering on Hidden Markov Models (HMM). For the base truth a commercial classification tool called Traffic Designer [3] is used. Using GMM together with TCP port numbers leads to an accuracy of 98%. One disadvantage of this approach is, that the TCP packets have to be in the correct order. An "attack" against this approach could be to pretending to "be" an other application during the first few packets or padding the payload of the the (first) packets.

3.3.26 Identifying and Discriminating Between Web and Peer-to-Peer Traffic in the Network Core

In [21] they present an approach to classify *unidirectional* flows from the network core. An unsupervised algorithm called *clustering* is used to classify the flows. With the Euclidean distance the K-Means algorithm with 400 clusters is applied. The selected features are *total number of packets*, *mean packet size*, *mean payload size excluding header*, *number of bytes transferred*, *flow duration* and *mean packet inter-arrival time*.
Because they have limited disk space on their network monitor, they collect only eight one-hour traces. About 85% of the flows and about 90% of the bytes are send over TCP and therefore they consentrate only on TCP traffic and left UDP for future work. Payload classification, transport-layer heuristics and HTTPS identification are used to find the base truth. About 29.1% of the flows and 4.2% of the bytes can not be classified using this methodology. This traffic is labled as UNKOWN (no payload, port 443 and others). They reach on average over the different traces 95% and 79% in terms of flow and byte accuracy, respectively.

3.4 Summary

The tables 3.1, 3.2, 3.3 and 3.5 (y $\widehat{=}$ yes, p $\widehat{=}$ partly) present an overview of the existing approaches. Which features is used by which work, which algorithm are implemented, etc. In table 3.4 is shown which work has what kind of collected traffic data and how much of it.

[3] Qosmos: http://www.qosmos.com

Hint to the approach and his reference	Active	Passive	Payload	Packet Header	bidirectional	unidirectional	Application Detec.	Host Classification	Traffic Classification	supervised Learning	unsupervised Learning	K-Means	GMM	EM (AutoClass)	HMM	Vector Quantization	DBSCAN	Naïve Bayes	Bayes Neural Network	AdaBoost	Maxent	Nearest Neighbor	LDA	TCP	UDP	specific packet format	P2P blocksize	RMS packet size	inter-arrival time
Large networks [60]		X					X	X	X															X					
File-sharing [31]		X	X				X																	X	X	X	X		
Accurate [63]		X																						X					
Transport-Layer [32]		X	X				X																	X	X				
Application Level [37]		X			?	?	X																						
Signature-Based [58]		X				X	X				X											X	X	X	X				
BLINC [34]		X		X	X			X										X		X				X	X				
ACAS [55]		X	X															X		X	X								
Accurate Identif. [46]		X	X			X	X																						
Flow perspective [38]		X				X			X															X	X				
Known & Unknown [11]		X																						X	X				
Self-learning [76]		X		X			X				X				X									X	X				X
Heavy-Hitters [68]	X	X			X	X		X				X						X						X	X				
Naïve Bayes [18]		X		X	X		X			X	X				X									X	X				X
Neural Networks [45]		X		X	X		X			X								X	X					X	X				X
Flow Correlations [41]		X		X		X			X			X												X	X				
K-Means & DBSCAN [17]		X		X		X					X	X			X		X							X					
Vector Quant. [15]		X		X		X							X			X								X	X				
On the Fly [6]		X		X	X	X					X	X												X					
Inherent [4]		X		X	X		X	X																X	(X)				
Soft EM [44]		X		X		X					X				X									X		X			X
Semi-supervised [20]		X		X			X			X	X	X												X					
Fingerprinting [12]		X		X																				X					X
Profiling [35]		X				X		X																X	X				
Early [7]		X		X			X			X		X	X	X										X					
Web vs P2P [21]		X				X	X				X	X												X					X

Table 3.1: Overview of some existing approaches

Hint to the approach and his reference	# Of Packets/Flow	Mean Payload	Packet Size Distr.	Flow Duration	Flow Size	Packet Size	Traffic Direction	Variance Packet Size	Jitter of First 4 packets	Source IP	Destination IP	Next Layer Protocol	Port Numbers	Order of the Packets	Header Bytes	caller to callee packets	caller to callee bytes	caller to callee payload bytes	caller to callee header bytes	producer and consumer	Network diameter	callee to caller pkts	callee to caller bytes	callee to caller payload	callee to caller header	minimum packet size	maximum packet size	quartiles of packet size
Large networks [60]													X															
File-sharing [31]																												
2003b													X															
Accurate [63]																												
Transport-Layer [32]																												
Application Level [37]																												
Signature-Based [58]	X		X	X	X					X	X	X	X															
BLINC [34]			X							X	X	X	X	X						X			X					
ACAS [55]																												
Accurate Identif. [46]																												
Flow perspective [38]	X			X	X		X						X															
2005h			X													X				X								
Known & Unknown [11]														X						X								
Self-learning [76]			X	X	X			X									X						X					
Heavy-Hitters [68]													X															
Naïve Bayes [18]	X	X		X												X				X		X						
Neural Networks [45]	X	X	X	X	X	X	X	X						X	X	X	X	X	X			X		X	X		X	
Flow Correlations [41]				X	X																							
K-Means & DBSCAN [17]	X	X	X		X	X										X				X			X					
Vector Quant. [15]	X				X																							
On the Fly [6]						X																						
Inherent [4]																X						X						
Soft EM [44]				X	X									X	X										X	X	X	
Semi-supervised [20]	X	X												X	X				X			X	X	X				
Fingerprinting [12]						X	X			X	X																	
Profiling [35]																												
Early [7]						X	X																					
Web vs P2P [21]	X	X	X	X	X	X		X	X																			

Table 3.2: Overview of some traffic features, used by existing Approaches

Hint to the approach and his reference	minPacketSize / max	the first five modes	# transitions betw. Trans. and bulk mode	time spent idle	time in bulkMode	time in Trans. Mode	rate of failed con.	in. to outgoing con.	portNumb. <> 1024	rate (kB/s)	Burstiness	246 features!	Wavelet Analysis	first 64/256 bytes	social behavior	functional level	hand made graphlets	#dstPort / #dstIP	recursive host detection	#noPayload flows	application port table	important port selection	Flow relation Map	TCP/UDP IP Pairs	IP, port pairs	byte, word, string match
Large networks [60]																										
File-sharing [31]																										
2003b																										
Accurate [63]																										X
Transport-Layer [32]																								X		
Application Level [37]																					X	X	X			
Signature-Based [58]																										
BLINC [34]															X	X	X	X	X	X						
ACAS [55]														X												
Accurate Identif. [46]																										
Flow perspective [38]																										
2005h													X													
Known & Unknown [11]																										
Self-learning [76]																										
Heavy-Hitters [68]																										
Naïve Bayes [18]																										
Neural Networks [45]												!!														
Flow Correlations [41]										X	X															
K-Means & DBSCAN [17]																										
Vector Quant. [15]																										
On the Fly [6]																										
Inherent [4]							X	X	X																	
Soft EM [44]	X	X	X	X	X	X																				
Semi-supervised [20]																										
Fingerprinting [12]																										
Profiling [35]																										
Early [7]																										
Web vs P2P [21]																										

Table 3.3: Overview of some traffic features, used by existing Approaches continued

Hint to the approach and his reference	collected traffic data
Large networks [60]	800M Cisco Netflows
File-sharing [31]	68h Tier 1 US ISP ca. 650M flows
Accurate [63]	36h of a backbone link: 4.58M TCP Connections and VPN T3 link: 6d/2.8G packets
Transport-Layer [32]	Tier 1 US ISP: 10h: ca. 300M flows
Application Level [37]	POSTECH: 866M flows
Signature-Based [58]	7d from Waikato, commercial streaming server logs, and 2 T3 lines
BLINC [34]	Genome campus 44h and residential university, 1.1M flows
ACAS [55]	100GB from 500 residential users
Accurate Identif. [46]	Genome campus 24h
Flow perspective [38]	3 weeks from their campus (6000 computers) 2.6G flows
Known & Unknown [11]	2*24h traces from a regional ISP, commercial and academic institutions
Self-learning [76]	flows from Auckland-VI, NZIX-II, Leipzig-II traces from NLANR
Heavy-Hitters [68]	30d, 60M flows per hour from SWITCH
Naïve Bayes [18]	72h from Univ. Auckland and 24h also from Univ, Auckland
Neural Networks [45]	10*24min non-overlapping payload samples from a 24h interval from the Genome Campus
Flow Correlations [41]	Los Nettos (2.8M flows) and NLANR (100000 flows)
K-Means & DBSCAN [17]	72h TCP/IP header trace from Univ. Aukland + 1 hour payload trace from Univ. Calgary
Vector Quant. [15]	90d from TeNeT and ISP's 1Mbps link
On the Fly [6]	6 moths trace
Inherent [4]	2*24h traces from a regional ISP, commercial and academic institutions
Soft EM [44]	traces from the Universities of Auckland and Waikato, ca. 20000 flows
Semi-supervised [20]	48*1h traces, Campus, Residential, WLAN. Spanned over 6 months
Fingerprinting [12]	faculty campus edge, 40GB (with payload)
Profiling [35]	month 10.2005 and 2 weeks from 11.2005, enterprise
Early [7]	8 different network edge, university, MIT, college. 20M flows
Web vs P2P [21]	8*1h traces, 9-10am and 9-10 pm, Thursday, Friday, Sat, Son. Campus. 5M flows

Table 3.4: Overview of collected traffic data, used by existing Approaches continued

	Large networks [60]	File-sharing [31]	Accurate [63]	Transport-Layer [32]	Application Level [37]	Signature-Based [58]	BLINC [34]	ACAS [55]	Accurate Identif. [46]	Flow perspective [38]	Known & Unknown [11]	Self-learning [76]	Heavy-Hitters [68]	Naïve Bayes [18]	Neural Networks [45]	Flow Correlations [41]	K-Means & DBSCAN [17]	Vector Quant. [15]	On the Fly [6]	Inherent [4]	Soft EM [44]	Semi-supervised [20]	Fingerprinting [12]	Profiling [35]	Early [7]	Web vs P2P [21]
Packet Features	y	y	y	y	y		y		y			y		y	y				y	y	y	y	y		y	y
Flow Features						y	y	y	y	y			y	y	y		y			y	y	y	y			y
Connection Features		y							y			y		y	y	y		y			y					
Intra Flow Features									y			y			y	y						y	y			
Multi-flow Features				y	y		y		y		y		y							y	y	y		y		
Automated Feature Selection					y							y			y							y				
Auto. New Application Detection				p		y					y		p				p		y		p	y				
Auto Adjust for the Features												p			y	y	y				y	y	y	y	y	y
User/Machine Behavior		y					p													p			y			
Active & Passive													y													
Traffic Classification		p	p							y																
Host Classification		p					y						y							p			y			
Application Detection		y	y	y	y	y	y	y	y	p	y	y	y	y	y			y	y	p	y	y	y		y	y
Byte Accuracy		y		y	y										y				y			y			y	y
Flow Accuracy			y		y	y	y	y	y		y	y		y	y		y	y			y	y	y		y	y

Table 3.5: Overview of the existing work, attributes of their approaches

Chapter 4

Zattoo Detection

4.1 Introduction

In June 2006 a small company named Zattoo Inc. [29] started a P2PTV (peer-to-peer Internet Protocol Television) system. Zattoo was first tried out with Swiss free-to-air channels coinciding with the FIFA World Cup 2006 in Germany. This service was only available in Switzerland, because they thought that the small Swiss market might be a good test before starting with bigger countries like Germany, France or UK.
The technical background is from the Project *End-Host Multicast* of the group “Warriors of the

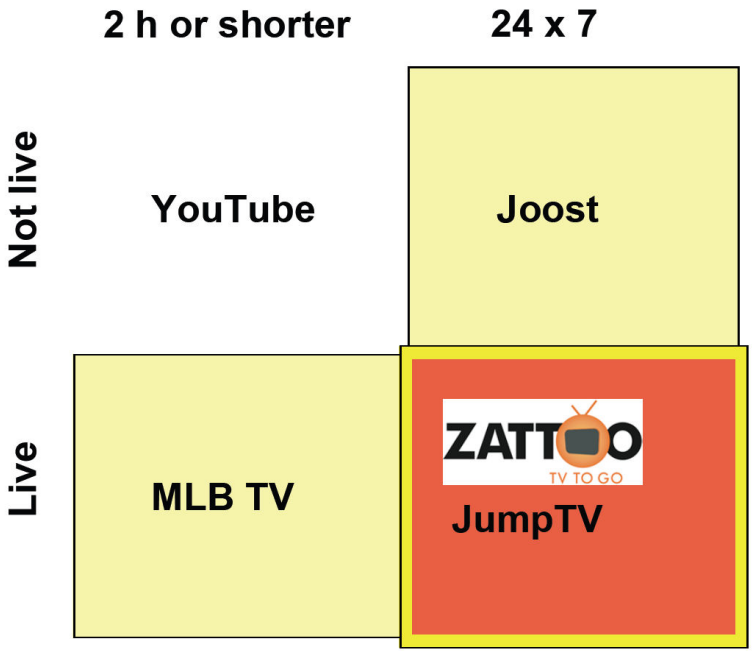

Figure 4.1: Zattoo Bussiness Position in the Internet television market

Net” led by Sugih Jamin[1] from the University of Michigan. Sugih Jamin is also the Chairman, CTO and Co-Founder from Zattoo Inc. With the software, available for Windows, Mac and Linux and a connection to the Internet everybody can watch TV without a TV set. Since June 2006 they successfully conclude contracts with several other TV stations in Switzerland, Germany, France, Italy, Spain, UK, etc. Some technical aspects are also published in some papers:

- How to reach hosts behind NATs (**N**etwork **A**ddress **T**ranslation) and Firewalls: [69], [73], [72]
- How to optimize a P2P network for streaming TV: [78], [77], [26], [71], [79], [70]

[1] http://irl.eecs.imich.edu/jamin

In Figure 4.1 (source: Zattoo Inc.: [56]) the position of Zattoo in the Internet television market is shown. Nothing has to be paid by the user to watch the several stations, since everything is paid by advertising, which is shown during the switches from one station to another and during the login-process. Figure 4.2 (source: Zattoo Inc.: [56]) shows a simplified view of the Zattoo P2P streaming network. The future goals are to enter to global market, the EU15 and North America

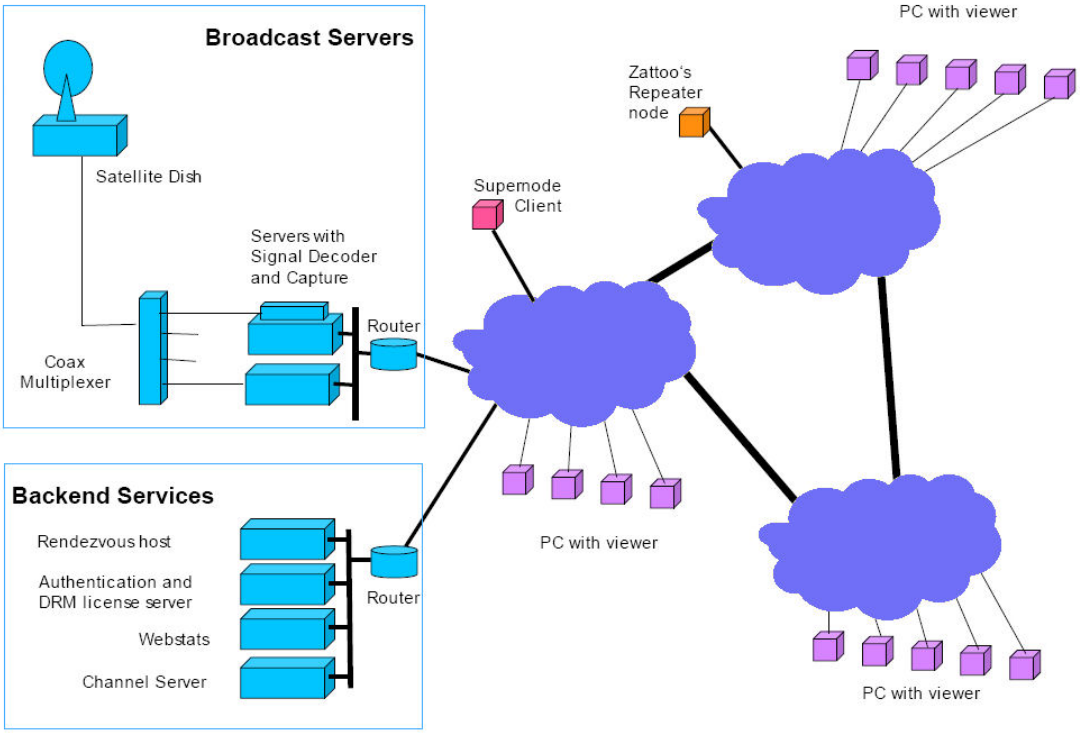

Figure 4.2: Zattoo P2P streaming Network

until end of 2007, Japan, Korea and Australia until mid of 2008. The biggest challenge are legal concerns, especially in North America.

4.2 Requirements

The goal is to construct a methodology for the detection of traffic form the Zattoo P2P system within SWITCH [65]. Such a methodology should be able to identify both, traffic produced by the Zattoo P2P system and the hosts which are members of this network. This methodology has to meet following needs:

- Detect all Zattoo traffic, both signaling and streaming data.
- Discover hosts which are members of the Zattoo P2P system.
- Examine the traffic detected as Zattoo traffic in comparison to “all” traffic from and to the SWITCH network.
- Compare the results with the “classical” port based approach.
- Such a scalability, that one is able to examine traffic from a long time period, e.g. 2-3 months.
- The detection methodology should be able to process the traffic data from near real-time.
- The detection should be side-independent (Also hosts which are behind a firewall or a NAT should be detected).

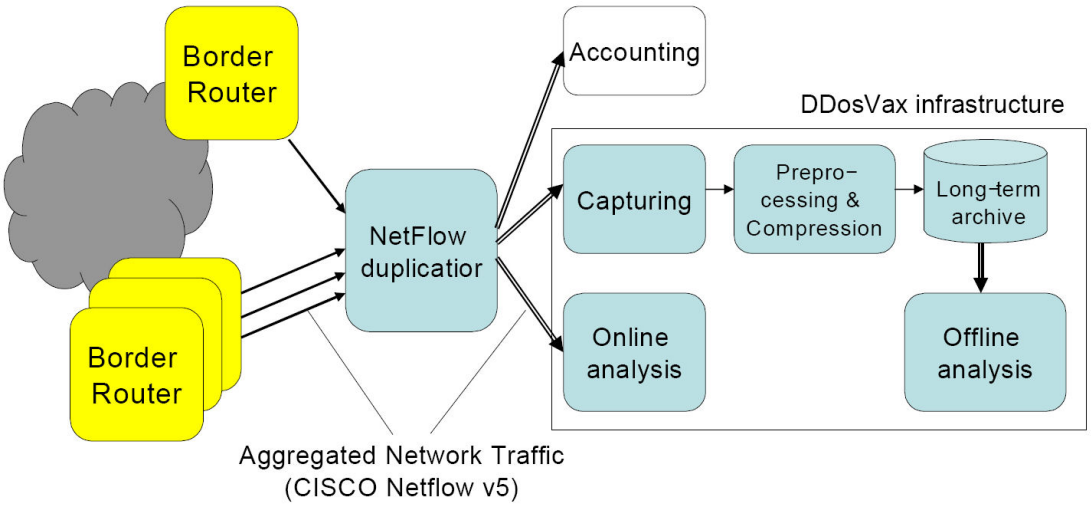

Figure 4.3: Architecture of the DDoSVax project

The methodology has to accept NetFlow v5 format, which can be get form the DDoSVax project [75], which was started in 2000 the detect distributed denial of service attacks. In Figure 4.3 the architecture of the DDoSVax project is shown. It maintains a large archive of NetFlow v5 data which is provided by the four border gateway routers of the SWITCH network since the beginning of 2003.

4.3 Analysis and Design

4.3.1 Manual Analysis of the Zattoo Traffic

First, the software from Zattoo was installed on a TIK[2] PC and on a student Laptop, both running under Windows XP. The traffic from and to this hosts was inspected locally with Ethereal. When the Zattoo client is started, first, the users ID and password are checked against Zattoo's database. Then a geo-filtering is made to decide from which country the user is, because the available TV station vary form country to country. After the user has selected a channel, the streaming starts form many sources over the port numbers 5002 for TCP and 5003 for UDP.
The funny thing is, that if after the installation of the Zattoo client, the Windows XP asks if one would aloud Zattoo get through the Firewall and one denies this question, Zattoo randomly chooses other port numbers and one can watch TV just the same.
The login with the user ID and password is to the best of our knowledge always performed with the same destination IP (207.210.107.116, GNAL-2), under which also `http://www.zattoo.com` is located. Also at the startup of the Zattoo client there is a connection with IP address of an advertising server (64.22.80.15, GNAL-2 and 91.123.96.14, Zattoo Inc. Networks). If the RIPE[3] database is search, one get that the address range 91.123.96.00 - 91.123.111.255 is assigned to Zattoo Inc.
From the manual inspection of the Zattoo traffic, following points are the background of the future detection approach:

- It was clear, that the standard port numbers for are 5002 for TCP and 5003 for UDP, but if the connection can not be build with these port number due to whatever reason (NATs, Firewalls, etc.) other port numbers are used. Also the fact, that many other Internet applications (e.g. Radio Free Ethernet, FileMakerPro Server, many trojans, etc.) us these two ports is not helpful.Therefore a detection of Zattoo traffic and host can not only be founded on port numbers.

[2] http://www.tik.ee.ethz.ch
[3] http://www.ripe.net

- The initial login and authentication over TCP with port number 80 can be used to mark hosts, which may watch television over Zattoo in the future.

We have several structural limitations which exclude a multitude of possible approaches which might be successful for this task.

- No packet payload is available, which might have been used to identify the Zattoo signaling flows or for evaluation (Although most of the zattoo traffic is encrypted [56]).
- We can not use any packet header information, which is not aggregated into the Netflow v5 format.
- We had no additional knowledge about the Zattoo P2P system at the time we started with the design of the detection methodology, which could be helpful. But on the other hand, it was the aim the build a detection approach for any exemplary Internet application. And it can not always be assumed, that such an application is open and its architecture is well known.

4.3.2 Design

Our Zattoo detection methodology is divided into four parts: Data Preprocessing, main processing loop, Accounting and Resetting and final processing.

4.3.3 Data Preprocessing

First, we get the data in *.bz2* compressed files which contain one hour from the border gateways from SWITCH. Because of the architecture of the DDoSVax project, there are always two different hour files. As shown in Figure 4.3 one file contains all traffic from three gateways and the other from the last one. The traffic is assembled into Cisco NetFlow v5 and sorted according to the export-time when the flows were exported from gateways. To be able to process later the whole traffic of all four border gateways in chronological order, the flows have to be sorted, i.e. for each hour the two files have to be merged. This whole preprocessing is done by the *data mole framework* of Dominik Schatzmann (Special Thanks!).

4.3.4 Main Processing Loop

After the preprocessing, there comes the main loop over every flow. Firstly we check, if the flow is:

- An *outgoing flow*, i.e. the source IP address matches with the SWITCH IP address space and the destination IP address is not in the SWITCH IP address space.
- An *incoming flow*, i.e. the source IP address matches not with the SWITCH IP address space and the destination IP address is in the SWITCH IP address space.
- An *internal flow*, i.e. the source IP address matches with the SWITCH IP address space and the destination IP address is in the SWITCH IP address space.
- An *transit flow*, i.e. the source IP address matches not with the SWITCH IP address space and the destination IP address is not in the SWITCH IP address space.

Depending on this classification, the flow is processed differently, because we treat hosts with are in the SWITCH network (*internal hosts*) distinctly than host from other ISPs (*external hosts*). The reason for this dissimilar attendance is simply that we expect for the internal hosts to be able to catch percentually more of their traffic than for external hosts. Although we are not able to collect the traffic from or to an internal host, which is not flowing over one of the border gateways, i.e. traffic from internal hosts to internal hosts, we think that this assumption holds.
We only account only to incoming and the outgoing flows, because this makes the whole accounting easier since so there is only outgoing and incoming traffic and no transit traffic. Another reason is, that there are only a few tens out of a million flows which are not incoming or outgoing flows. (Special thanks to Bernhard Tellenbach, who implemented the IP matcher!)

Internal Zattoo hosts

If a outgoing flow has the IP address of 207.210.107.116, 64.22.80.15, 91.123.96.14, which as already mentioned belongs to Zattoo Inc. and is called every time a Zattoo client is started, as destination IP address, then the internal host is marked as potential Zattoo host. For every potential Zattoo host, a `struct` of the type *zattoo_host* (see Appendix B.1.1) is allocated. This data structure contains all necessary primitives to account much information of the traffic from and to this host. For each flow, which is marked as *zattoo_flow*(see below), from or to this host, several counters are incremented. E.g. the number of bytes from or to this host for TCP and UDP, the number packet, the number of flows the duration of these flows, etc.
Each internal Zattoo host data structure carries also an mean metric and some variance metrics to calculate these moments iteratively (see B.2) for this host. Additionally there is a hash table where all (external) Zattoo communication partners are stored with there IP address.
All the internal Zattoo host data structures are stored in a hashtable. So it can be checked, if a IP address is already marked as (possible) internal zattoo or not.

External Zattoo hosts

If we detect traffic from an internal Zattoo host to an IP address which is not part of the SWITCH IP address space, and we conclude that this traffic is from the Zattoo application (how we conlcude this is shown in the next section), this IP adresse is mark as *external Zattoo host*. As for the internal Zattoo hosts, a `struct` of the type *externalZattooHost* (see Appendix B.1.2) is allocated. This data structure is also stored in a hash table, so one can always check if there is already an external Zattoo host `struct` for a certain IP address.

Zattoo Data Traffic

As already mentioned, the standard port numbers for Zattoo are 5002 for TCP and 5003 for UDP. If we detect traffic on one of these standard ports with a internal Zattoo host ad source or destination (because we can not assume, that the routing is symmetric) with an IP address not in the SWITCH IP address space, we declare this flow as Zattoo traffic. If the flow is an outgoing flow, we build a new external Zattoo host with the destination address of this flow, if no external Zattoo host with this IP address is already in the hashtable of the external Zattoo hosts. If the flow is an incoming flow, we build an external Zattoo host under the same conditions.
If we detect traffic on the standard Zattoo port numbers from an external Zattoo host to an IP address within the SWITCH IP address space, which is not already in the hashtable for the internal host, then we mark this flow as Zattoo traffic and create a new internal Zattoo host. This, because we do not assume, that we are able to detect all flows between SWITCH hosts and Zattoo Inc. and let therefore open this second possibility to detect internal Zattoo hosts.
If a flow is detected from an already caught internal Zattoo host to an already detected external Zattoo host on other ports than the standard ports, then this flow is marked as *possible Zattoo flow*.
To have the possibility to compare the results from our detection approach with another methodology at all, we also implement a port based detection, where only the port numbers 5002 for TCP and 5003 for UDP are used to decide, if a flow is part of the Zattoo traffic or not.

4.3.5 Accounting and Reseting

For each parsed flow, we account some properties on flow level, host level and ISP level (of course we have only “our” ISP SWITCH). It was difficult to decide which measurements should be done. Since we have only information from the flow level and no deeper information, e.g. packet header, we try to catch as much information as possible. It is obvious that, e.g. there are always more possible combinations of flow level attributes, which can build host level properties.

Per Flow Accounting

There are many attributes of a flow which as to measured in the context of application detection. Since we have our traffic data from border gateways, we calculate for each flow in which

category it falls from the point of view of the ISP, this it already mentioned in section 4.3.4. Also the counters for bytes, packets and flows, received and sent, and for TCP and UDP, are updated with each flow. If we have a Zattoo flow, we update all counters, again for bytes, packets and flows (TCP and UDP). So, we can compare the amount of the overall traffic with the Zattoo traffic.
Also counters for the connections from and to the IP addresses of Zattoo Inc. are logged, since the are indicators of changes in the application behavior of Zattoo. If a flow is marked as possible Zattoo flow, the byte, packet and flow counter for the unsure Zattoo traffic are incremented. And the distribution of these possible Zattoo flows and bytes over the port number range [1-65535] for both, TCP and UDP are updated.
For each Zattoo flow, the average data rate, the average packet size and the average packet inter-arrival time are calculated and with them, the distributions of the flows over the particular scope is updated, again for both protocols, TCP and UDP. Last but not least, the two metrics number of bytes and packets per flow are computed and their distributions over a selected range (1-10ˆ6 in steps of 10 for the bytes and 1-10ˆ5 for the packets) are updated.

Per Host Accounting

In our detection approach, the host plays the most important role. Every host is part of our detection methodology and because we want to explore, how our multi-flow or host based detection performs, we calculate a lot of statistics for every Zattoo host.
We start with the counters to Zattoo Inc., where it is interesting how many such connections a host has while “he” is watch TV on Zattoo. Then, for each Zattoo flow a internal Zattoo host sends or receives, we update the following counters:

- Number of sent and received flows (TCP and UDP)
- Number of sent and received packets (TCP and UDP)
- Number of sent and received bytes (TCP and UDP)
- Number of sent and received flows (TCP and UDP) on other ports than 5002 for TCP and 5003 for UDP
- Number of sent and received packets (TCP and UDP) on other ports than 5002 for TCP and 5003 for UDP
- Number of sent and received bytes (TCP and UDP) on other ports than 5002 for TCP and 5003 for UDP
- The data rate for sent and received traffic (TCP and UDP)
- The start-time of a host’s first Zattoo flow and the end-time of a host’s last Zattoo flow
- The total duration of all sent and received Zattoo flows (TCP and UDP)
- The number of external Zattoo host, if there was no connection with the actual external zattoo host of this flow up to now

For flow metrics average data rate, average packet size and average packet inter-arrival time also the per-host values (mean and variance) are updated.
For the external Zattoo host, the statistics are less extensive. The reasons are the memory consumptions, since we have much more external hosts, than internal hosts (see table 5.1 in section 5.2) and again the fact, that we can catch more information for the internal host than for the external, because we are able to collect all cross-ISP-traffic from these hosts what we can not do for the external hosts. For each flow, the following attributes of an external Zattoo host are refresh:

- Number of sent and received flows (TCP and UDP)
- Number of sent and received packets (TCP and UDP)
- Number of sent and received bytes (TCP and UDP)

- The end-time of a host's last Zattoo flow
- The number of internal Zattoo host, if there was no connection with the actual internal zattoo host of this flow up to now

The last category of accounted hosts are the hosts, which are detected by the port based reference approach. Here, hosts with IP addresses are observed, we call this kind of host "portBased_zattoo_host" (see Appendix B.1.3) we do not refer to external hosts, since the detection does not make use of them and again because of memory consumptions.
As every other category of host, we count the sent and received bytes, packets and flows for each host. As the internal Zattoo hosts, also the "portBased_zattoo_host" has a time property, which indicates the end-time of the last sent or received flow (whatever it was a TCP or UDP flow).

Per Interval Accounting

To have an overview about the trends for the Zattoo traffic, some interval-based are done. The interval length is a free choice, but in our measurements, we set it to one hour. After each interval, some statistics and counters are stored and reset. The most obvious statistic is the number of IP addresses with Zattoo traffic during the last interval. If an internal Zattoo host's end-time of its last Zattoo flow is bigger (in ms) than the last interval's start, than this host is counted as active host. To compare the results with the port based methodology, we do the same with the "portBased_zattoo_hosts". Also the byte, packet and flow counters are stored and reset after each interval. To get the longterm behavior of the features average packet size, average packet inter-arrival time and average data rate, the distribution of the flows and hosts over the range of this attributes are also stored and reset. Additionally, these distributions are also cumulated and normed to one, to have traffic-amount-independent distributions over the time. In this context, cumulated means that e.g. the cumulated number of flows with an average packet size of n bytes is the cumulated number of flows with an average packet size of $n-1$ bytes plus the number of flows with an average packet site of n bytes.
If a Zattoo host (internal, external and port based) has no traffic for an so called "Aging-interval" it is removed from his hashtable. We call this action "Aging" This action makes the whole detection approach more scalable in memory consumption, which is with this deletion only dependent on the number of active Zattoo hosts and not on the total number of Zattoo hosts. It has also the advantage, that an off-line detection process, as we performed it, can be clustered. If the jobs overlap each other, the result is the same as if there is no "Aging".

4.4 Evaluation

First, it is very clear, that our detection methodology lacks of evaluation and verification. Since we have only Netflow data, we are not able to build reference results which are accurate enough to be used to get *false positives* or *false negatives* produced by our approach.
The only possibility to compute results which can be compared with our approach is to run a port number based methodology. It is clear, that specially this kind of detection is not really adequate for an application which uses arbitrary port numbers as Zattoo does. Although it is better to have results from two different methods than from one only. Having two different sets of results, it is possible to check if our results have the differences to the port based results, as they should have (see Chapter 5). Another way is to ask Zattoo Inc, if the have accurate numbers for their traffic within SWITCH. To compare the measurements for the flow and host distribution over the metrics average packet size, average packet inter-arrival time and average data rate it would also be nice the have some values from other applications or the average over the total traffic from SWITCH. E.g. to compare Zattoo with BitTorrent with respect to them. Because of the performance, it is not an option to compare these attributes of the Zattoo traffic with the results for the total traffic, since the computation of these values would have to be done for each flow what would increase the time needed to process one flow dramatically.
To improve the performance of our algorithm, some advancements can be made:

- Before processing a flow, it could be check if the flow has more than one packet or consists of more bytes than a certain threshold.

- Only metrics which are relevant for detection are computed, all other which serve only accounting purposes are omitted.
- Also flows, which can be clearly assigned to other applications or services are not process (E.g. ICMP).
- Flows with invalid attributes are also left out. E.g. invalid source or destination addresses, wrong timestamps, etc.

To enhance our methodology, additional metrics like the number of different communication partners per port number might be helpful. Also the ratio sent and received flows or bytes, the ratio of TCP to UDP traffic or the number of failed connections per port number can be used to make our approach more accurate.

Chapter 5

Results

In this chapter, the result from our longterm run of our Zattoo detection methodology are presented. The algorithm has been running with data from 19 weeks. Start was the March 30, 2007 at 06:00:00, we call this week one. An overview is given in the tables 5.1, 5.2 and 5.3. Week 9, started at May, 25, had a corrupted file and was not processed.
First, we present the results which are computed with all detected Zattoo flows and hosts from the whole 18 weeks from which we have the results. Second, the time dependent metrics from the weeks 10 to 19 are shown.

5.1 Summations and averages over the whole period

In this section, all accounted statistics which more or less represent attributes of hosts and flows that have to be build from summations over time are described. This includes distributions of the Zattoo hosts and flows over the range of some general flow attributes like e.g. average packet inter-arrival time.

5.1.1 Average Packet Size

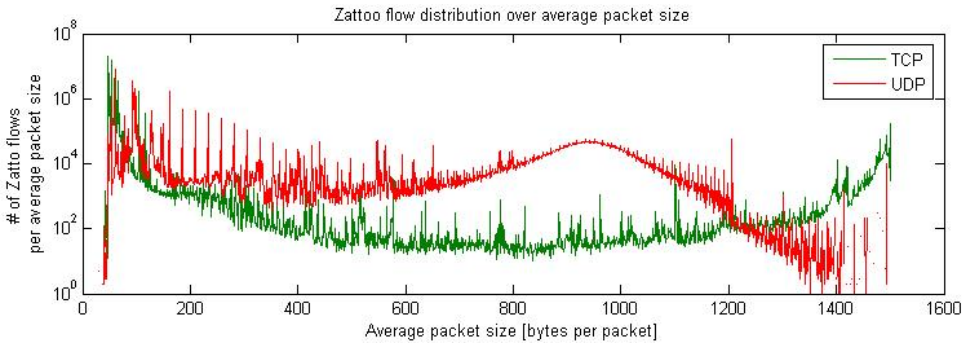

Figure 5.1: Distribution of the Zattoo flows over the average packet size

In Figure 5.1 the distribution of the flows over the average packed size range (we set this range from one to 1500 bytes) is shown. For a P2P application it can be expected, that for the signaling traffic smaller packet size is used and that for the data transmission the Maximum Transmission Unit (MTU) is occupied. In our measurements, the results do not match with our expectation. For TCP, we most of the flows are smaller than 200 bytes and for UDP the majority of the flows is smaller 1000 bytes. The difference from our expectations and our result is because our expectations were based on P2P file-sharing applications.
The distributions of the flows over the packet size becomes more clear, if we cumulated the number of each value from the average packet size as shown in figure 5.2. This means that we add always the number of Zattoo flows with the average packet size $n-1$ to the number of Zattoo flows with the average packet size n and set this value for the cumulated number of

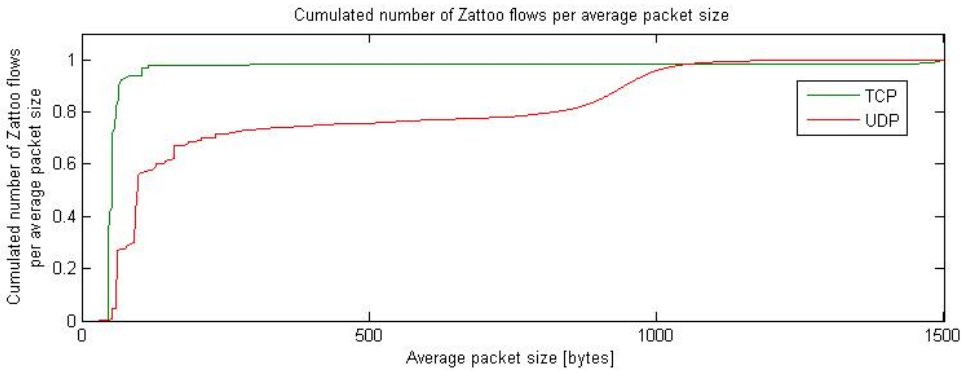

Figure 5.2: Cumulated distribution of the Zattoo flows over the average packet size

Zattoo flows with average packet size of n. In figure 5.2 it is clearly visible, that over 90% of the TCP Zattoo flows have an average packet size below 200 bytes, that about 70% UDP Zattoo flows are also below the 200 bytes and that for both protocols, almost 100% of the flows have an average packet size smaller than 1000 bytes. These facts might be helpful to exclude some flows if one is not sure if some traffic is from the Zattoo application or not.

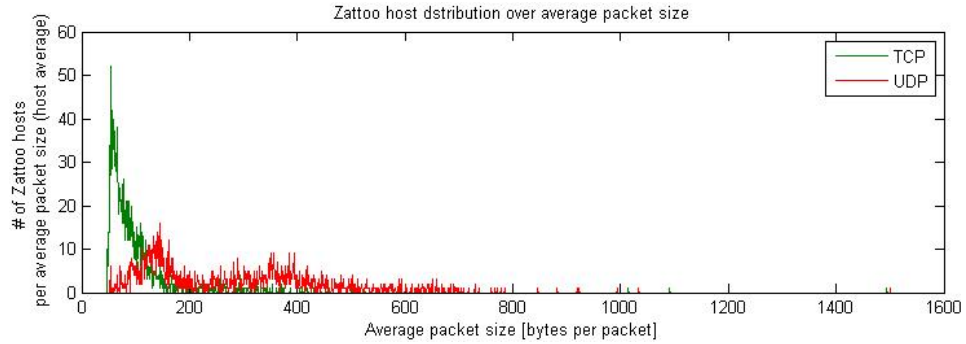

Figure 5.3: Zattoo host distribution over the average packet size

If we look at the distribution of the Zattoo flows and their cumulated distribution over the average packet size in figure 5.3 and 5.4, than it is obvious that they are similarly distributed as their Zattoo flows. We first computed the average packet size for each flow on which a host is participating, and than construct the mean of all these averages for this host.

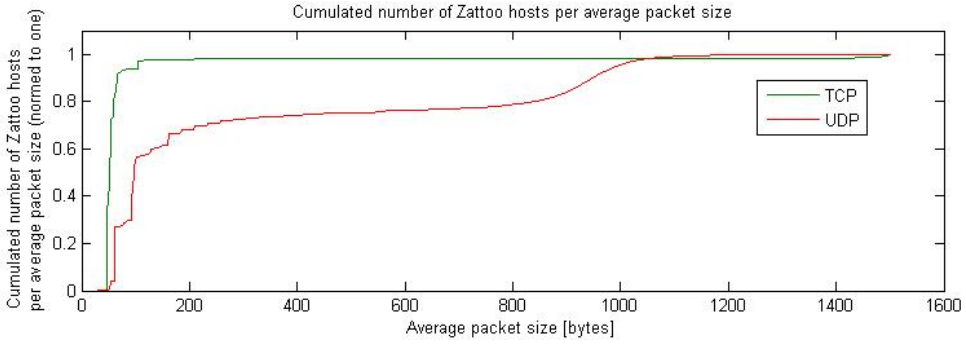

Figure 5.4: Cumulated Zattoo host distribution over the average packet size

5.1.2 Average packet inter-arrival time

In Figure 5.5 the distribution of the flows over the range of the average inter-arrival time of the packets is shown. The average inter-arrival time of the packets is callculated with the number

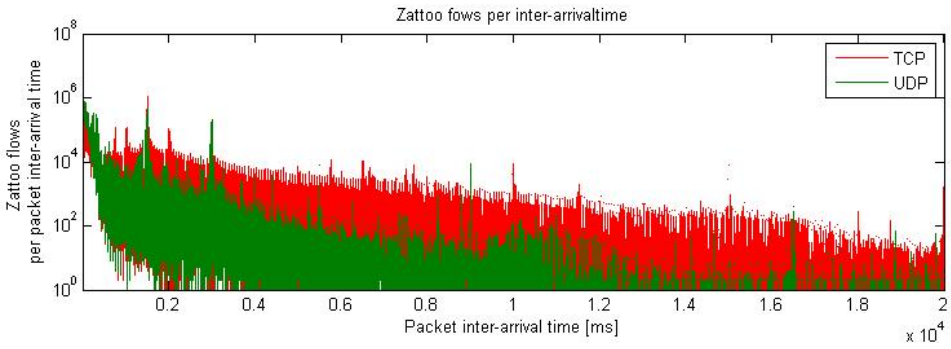

Figure 5.5: Zattoo flow distribution of the average packet inter-arrival time

of packets per flow divided by the flow duration. In figure 5.6 it becomes more visible that approximately 90% of the TCP Zattoo flows have an average packet inter-arrival time smaller than 1000 milliseconds and that the 90% limit for the UDP flows is around 2000 milliseconds.
If the mean of the average packet inter-arrival time is computed for every Zattoo host as in figure 5.7, their distribution is a little bit different in comparison the Zattoo flows. Almost every Zattoo host has a mean TCP average packet inter-arrival time below 400 milliseconds and for UDP the picture becomes clearer with figure 5.8, where it can be seen that approximately 80% of the Zattoo hosts have a mean average packet inter-arrival time below 750 milliseconds.

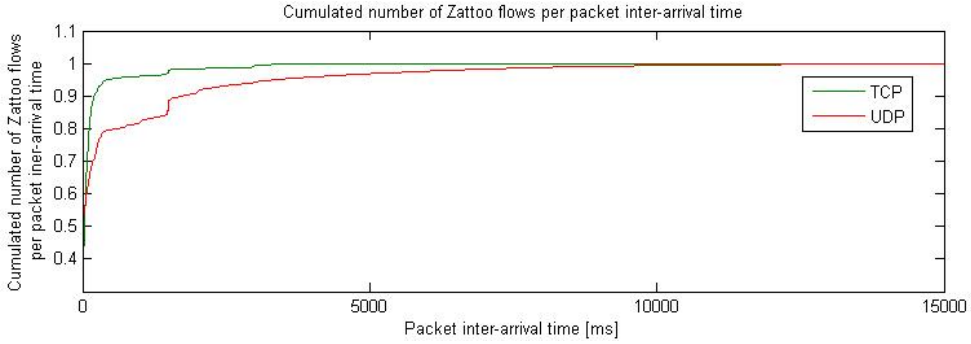

Figure 5.6: Cumulated Zattoo flow distribution of the average packet inter-arrival time

As for the average packet size, also Zattoo flow and host distributions over the average inter-arrival time can be used to determine if some traffic is from the Zattoo P2P system. We assume, that the small packet inter-arrival times are because a constant data arriving is necessary for an IPTV application.

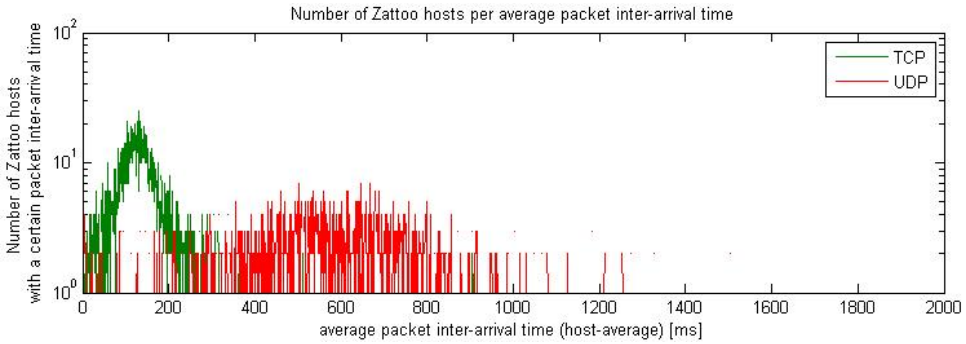

Figure 5.7: Zattoo host distribution of the average packet inter-arrival time

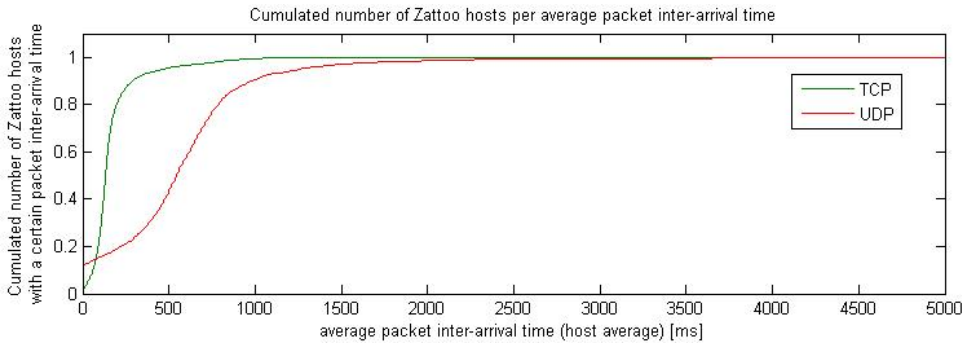

Figure 5.8: Cumulated Zattoo host distribution of the average packet inter-arrival time

5.1.3 Data Rate

We got the impression from the average packet inter-arrival time, that a more or less constant data arrival time is needed for an IPTV P2P network, we expected to got a analog image for the data rate. We calculate the data rate for each Zattoo flow by dividing the size of a flow in bytes with the flow duration.

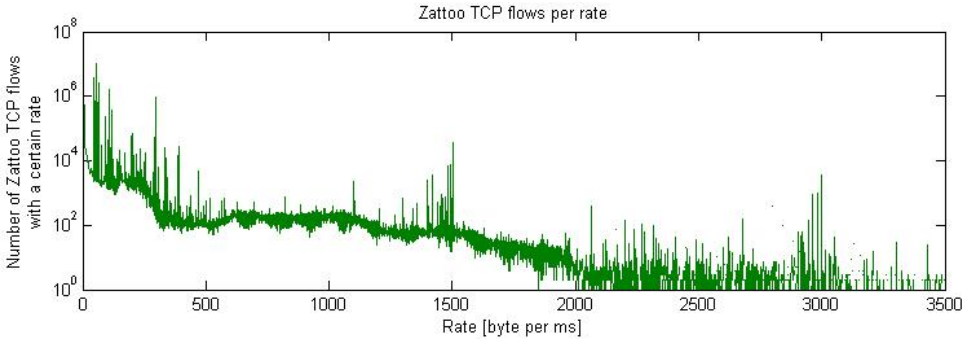

Figure 5.9: The Zattoo TCP flows distributed over the range of the data rate

If a flow had a duration of zero milliseconds, the data rate was set to the number of bytes, as if the flow duration was one millisecond. Because the distributions of the TCP and UDP Zattoo flows overlap each other, we had to make the two figures 5.9 and 5.10. Because we assumed that most hosts in the SWITCH network have very fast Internet access, we thought that the data rate would be rather big. We found, that most of the Zattoo flows are not as fast as wee expected.

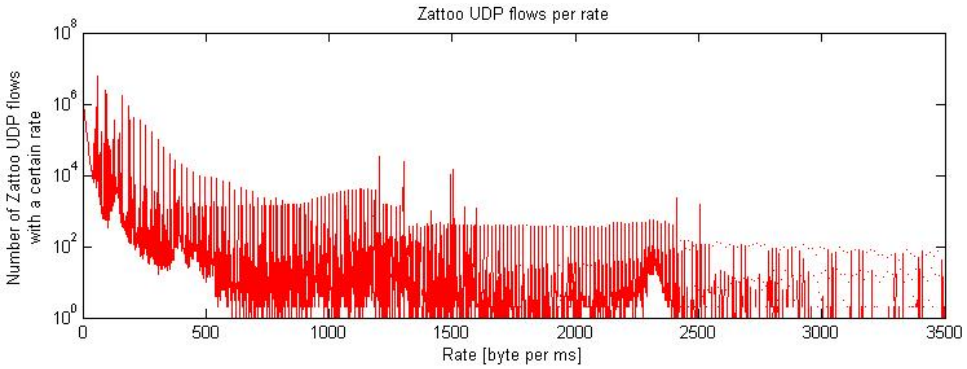

Figure 5.10: The Zattoo UDP flows distributed over the range of the data rate

Again this becomes more clear if the cumulated distribution of the Zattoo flows in figure 5.11 is considered. Nearly 100% of the Zattoo flows have data rate not bigger than 250 bytes per milliseconds. This might be that small, because our assumption was again based on consider-

ations over P2P file-sharing systems, where the traffic is transmitted with the highest data rate as possible [60].

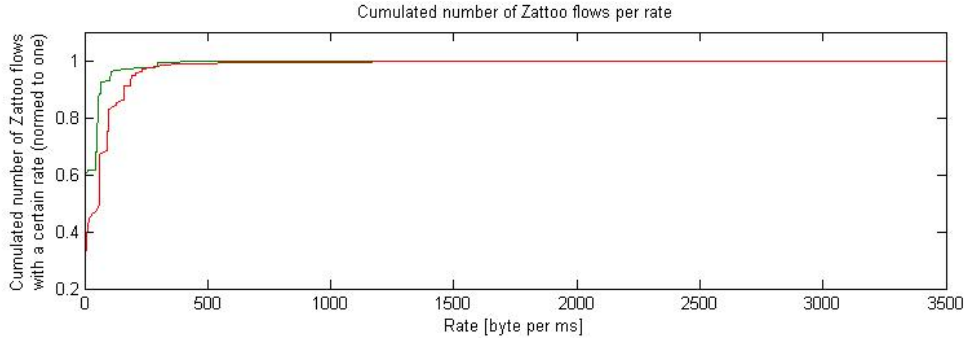

Figure 5.11: The Zattoo flows distributed cumulatively over the range of the data rate

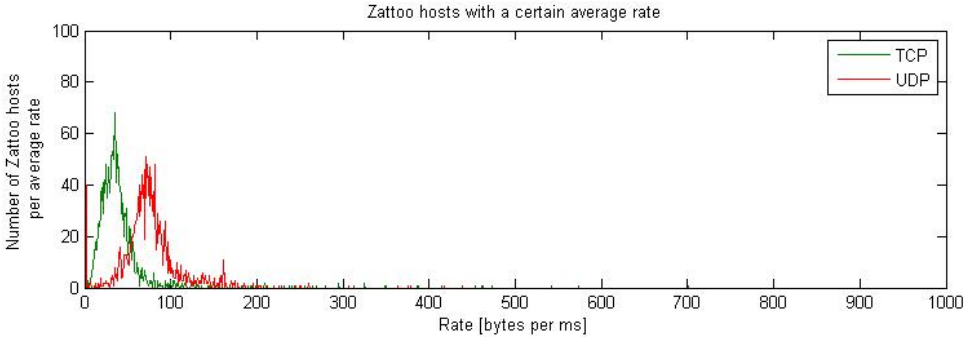

Figure 5.12: The Zattoo hosts distributed over the range of the data rate

Again, we have different distributions for the mean data rate per Zattoo host (see figure 5.12). Most of the hosts have a similar mean data rate and it is smaller than the the distributions of their flows let assume. About 90% of the hosts have a data rate smaller than 100 bytes per millisecond. In figure 5.13 it is also visible that the average per host for TCP is about half of the value for UDP.

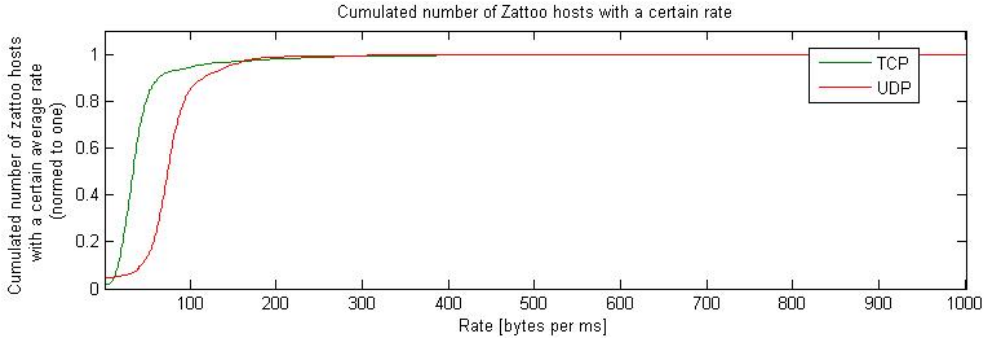

Figure 5.13: The cumulated number Zattoo hosts over the range of the data rate

5.1.4 Bytes per Flow

In figure 5.14 it is viewable that the distributions of the Zattoo flows are similar to the distribution of all the flows from the SWITCH network. We set the range of this metric to [0-5E6] bytes since we thought that most of the flows would fall into this range (not most of the bytes, since the majority of the bytes is carried by a few *elephant flows*). If additionally to figure 5.14 also figure

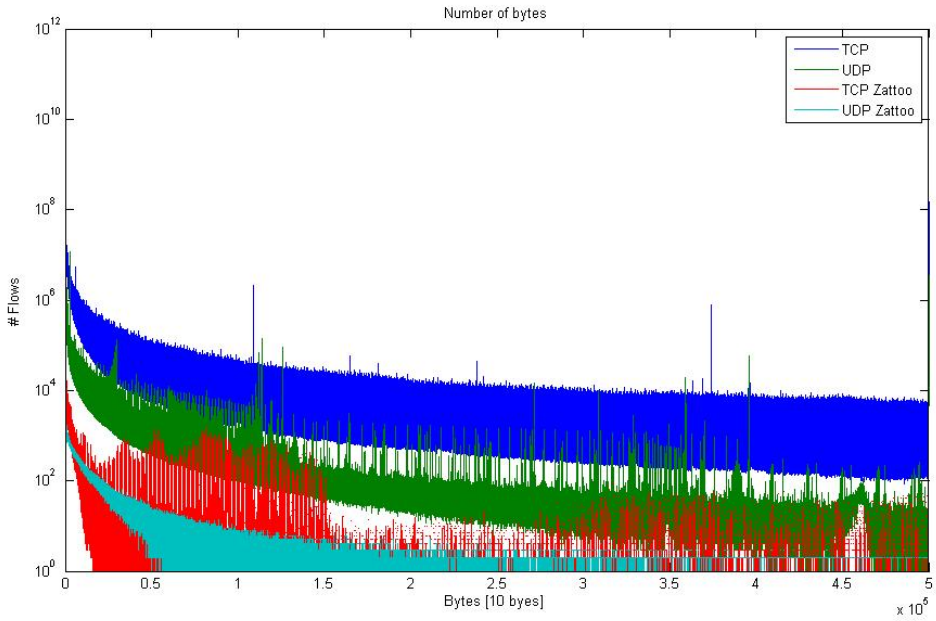

Figure 5.14: The Zattoo flows distributed over the flow size

5.15 is considered, it becomes clear, that this metric is not very helpful to distinguish between Zattoo traffic and the total amount of traffic.

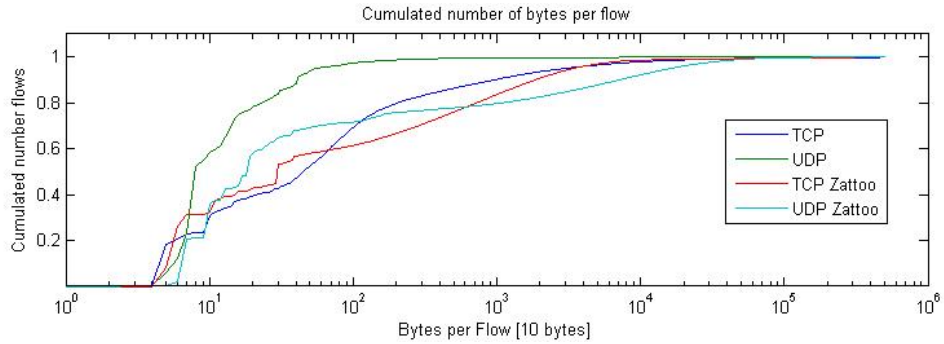

Figure 5.15: The cumulated number of Zattoo flows distributed over the flow size

5.1.5 Packets per Flow

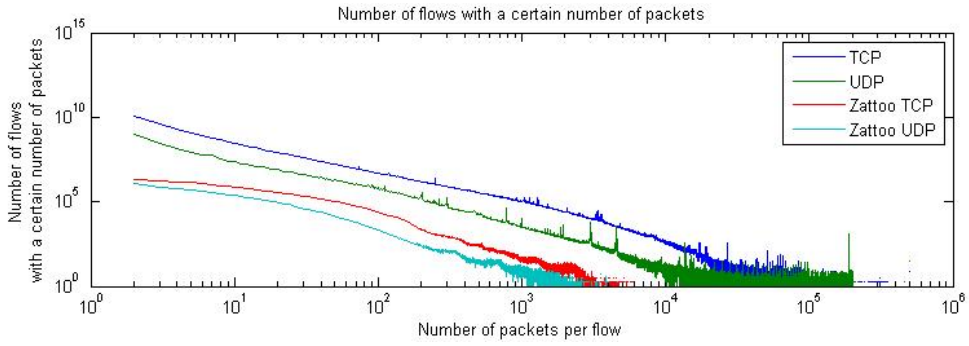

Figure 5.16: The Zattoo flows distributed over the number of packets per flow

We have made the same operations with the metric number of packets per flow. The result are not really different to them from the metric bytes per flow. Only in figure 5.17 some small distinction between Zattoo traffic and the total amount of traffic comes out, since the cumulated distribution of the flows with a certain number of packets raises slower for the Zattoo traffic.

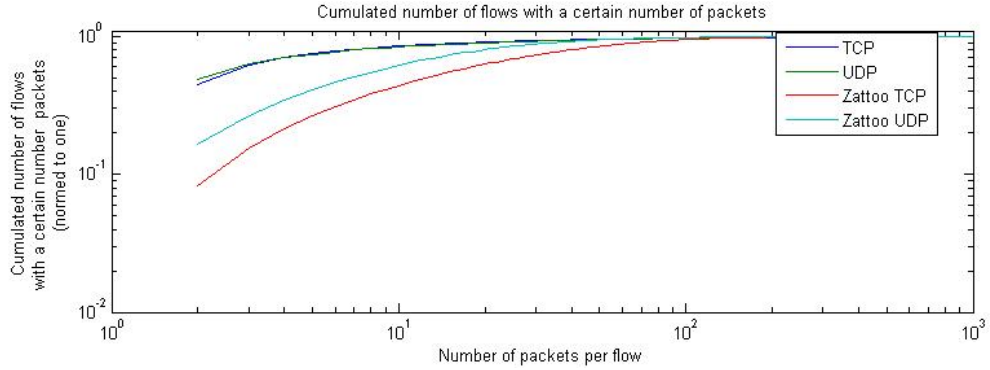

Figure 5.17: The cumulated number of Zattoo flows distributed over the number of packets per flow

5.1.6 Distribution of the Zattoo bytes and flows on the non-standard port numbers

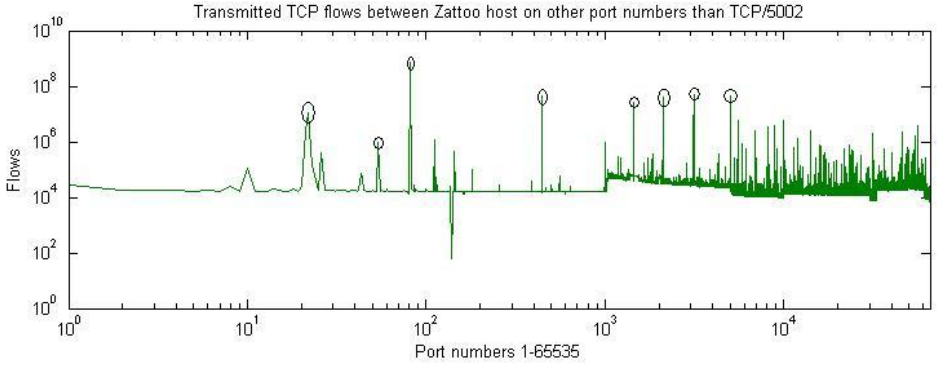

Figure 5.18: TCP flows between Zattoo host on non-standard port numbers

Since we know that the Zattoo apllication is able to transmit its traffic on arbitrary port numbers, we wanted to check if there are some other port numbers than 5002 for TCP and 5003 for UDP which are preferred by the Zattoo P2P system. For TCP, we got the port numbers 22, 54, 81, 444, 1434, 2129, 3129 and 5003, for UDP the port numbers 1, 38, 54, 124, 501, 1033, 1117, 2123, 4122, 5851, 8090 and 23128 with visible more flows than the other port numbers. This port numbers are marked in the figures 5.18 and 5.19.
From the distribution of the Zattoo bytes over the non-standard ports also no additional knowledge about the usage of the port numbers can be gained. For UDP, some port numbers between 30000 and 60000 carry more traffic than the rest of the port numbers. Generally it can be said, that the port numbers above 1024 transport much more traffic than the privileged ones.

5.2 Overall Traffic Statistics

In the tables 5.1, 5.2 and 5.3 we present accumulated traffic statistics mainly the number of internal and external Zattoo hosts, their connections to Zattoo Inc. and the sent and received Zattoo flows, packets and bytes, both for TCP and UDP. We accounted totally 98'748 internal Zattoo hosts, which had contact to 1'401'561 external zattoo hosts. It is clear, that because of

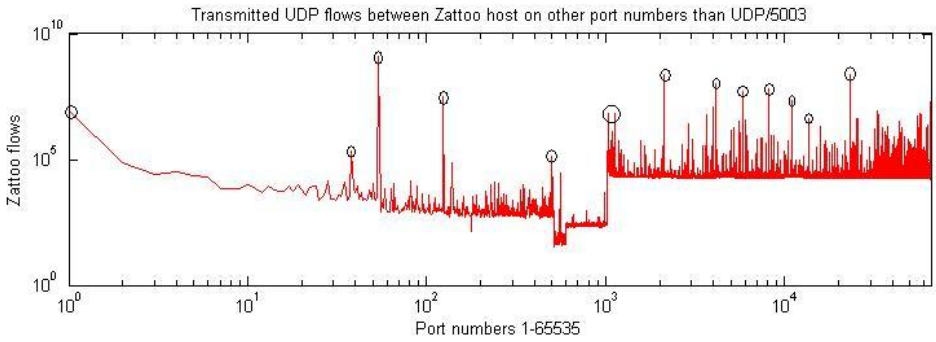

Figure 5.19: UDP flows between Zattoo host on non-standard port numbers

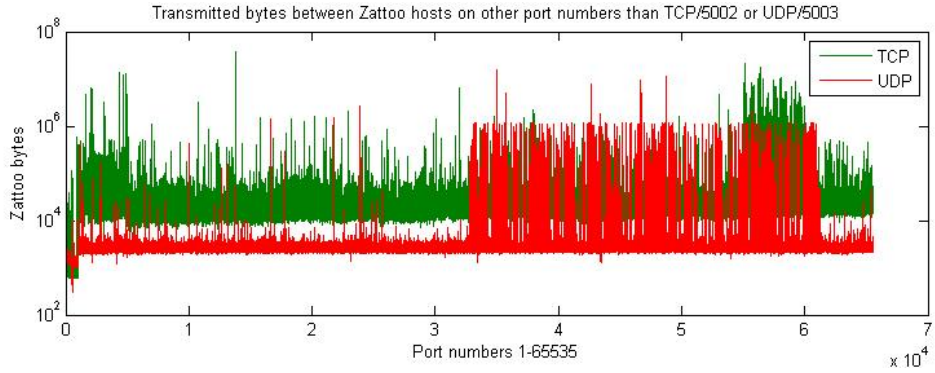

Figure 5.20: Bytes between Zattoo host on non-standard port numbers

our Aging process, we counted both some internal and some external Zattoo hosts more than once, since we deleted them if they had no Zattoo traffic for more than 24 hours.
Overall, it can be noted that as for total SWITCH traffic [68], also for the Zattoo application the hosts within SWITCH produce more traffic than they consume. From these tables it is more or less visible, that the summer holidays at the Swiss universities started around end of June.

Data	Week	start Date (start Time = 06:00:00)	Result	Nbr. of Zattoo hosts	Nbr. of external Zattoo hosts	Conn. from/to Zattoo.com	Conn. from/to ZattooAd	All received Zattoo TCP flows	All sent Zattoo TCP flows	All Zattoo TCP flows	All received Zattoo TCP packets	All sent Zattoo TCP packets	All Zattoo TCP packets
212	1	30.03.	success	4997	84082	34441	41596	117187	319156	436343	2189079	60789301	62978380
213	2	06.04.	success	3684	65053	19905	24029	10789	240237	251026	482051	43228434	43710485
214	3	13.04.	success	4783	73311	29408	46136	17209	357664	374873	60544	60632698	60693242
215	4	20.04.	success	5047	74803	26545	88011	41758	872199	913957	9492497	134940764	144433261
216	5	27.04.	success	6326	76118	42598	99701	70471	980875	1051346	4175004	143447437	147622441
217	6	04.05.	success	7222	88307	49997	91201	31511	1259489	1291000	4248410	157614382	161862792
218	7	11.05.	success	6688	88882	34101	71073	13878	729855	743733	47552	97209734	97257286
219	8	18.05.	success	11454	93898	36332	89799	24286	1047182	1071468	364854	136355845	136720699
220	9	25.05.	error										
221	10	01.06.	success	7345	84052	44795	88947	68771	1261641	1330412	33237465	205670858	238908323
222	11	08.06.	success	6660	102555	48503	80363	33047	989943	1022990	2127834	129088736	131216570
223	12	15.06.	success	4452	76258	38568	62761	85412	1507028	1592440	54058639	240277188	294335827
224	13	22.06.	success	7624	63044	57507	90102	216655	2381445	2598100	54492077	301709000	356201077
225	14	29.06.	success	7064	103835	45761	84310	524546	884851	1409397	5147341	101180140	106327481
226	15	06.07.	success	4315	80800	27293	56021	684702	1310639	1995341	41711861	187423158	229135019
227	16	13.07.	success	3388	59644	19394	37811	123996	915429	1039425	16638283	140830404	157468687
228	17	20.07.	success	3086	70549	20703	32330	30354	342164	372518	781487	92058439	92839926
229	18	27.07.	success	2148	56364	11384	16017	355867	569134	925001	36974332	145784857	182759189
230	19	03.08.	success	2465	60006	11239	23309	53377	970084	1023461	39754519	181931447	221685966

Table 5.1: Overview of the results from the 19 weeks traffic observation

Data	Week	All received Zattoo TCP bytes	All sent Zattoo TCP bytes	All Zattoo TCP bytes	All received ZattooUDP flows	All sent Zattoo UDP flows	All Zattoo UDP flows	All received Zattoo UDP packets	All Sent Zattoo UDP packets	All Zattoo UDP packets
212	1	139520570	6333638340	6473158910	222424	200054	422478	5850500	11919162	17769662
213	2	27081838	5761364419	5788446257	123806	141920	265726	3048868	8415957	11464825
214	3	38569095	4661414987	4699984082	222325	248640	470965	6247310	15571742	21819052
215	4	13726583434	16619641924	30346225358	263196	520771	783967	10790297	30142166	40932463
216	5	4848906263	14037894602	18886800865	262178	484863	747041	6195495	22029600	28225095
217	6	6246628772	11363089792	17609718564	314847	579927	894774	8504767	30804073	39308840
218	7	29865069	9155035609	9184900678	313990	439032	753022	8338832	21726006	30064838
219	8	474649870	10025706607	10500356477	524242	677841	1202083	13688922	33692090	47381012
220	9									
221	10	49312713131	40018144505	89330857636	379760	575506	955266	9668853	30317950	39986803
222	11	2242689425	11102409418	13345098843	137082	137905	274987	242301	310689	552990
223	12	80010591372	73687738154	1.53698E+11	215711	98939	314650	711873	208841	920714
224	13	71564630152	69989644356	1.41554E+11	128934	220894	349828	228750	428869	657619
225	14	319350685	14116247026	14435597711	247071	118531	365602	2886956	241919	3128875
226	15	56692461090	49940624370	1.06633E+11	162693	186592	349285	298823	402963	701786
227	16	24184032716	31791626315	55975659031	102464	109536	212000	183529	241519	425048
228	17	199791497	10443851948	10643643445	138816	72607	211423	726720	178744	905464
229	18	54556679933	54261002177	1.08818E+11	122866	68170	191036	723587	135632	859219
230	19	58973927004	64536685251	1.23511E+11	88349	97847	186196	147194	198859	346053

Table 5.2: Overview of the results from the 19 weeks traffic observation (continued)

Data	Week	All received Zattoo UDP bytes	All Sent Zattoo UDP bytes	All Zattoo UDP bytes
212	1	5196117693	10899476140	16095593833
213	2	2709252846	7690373984	10399626830
214	3	5589059517	14223367448	19812426965
215	4	9730091698	27677797077	37407888775
216	5	5464638001	20093257784	25557895785
217	6	7563306338	28049717225	35613023563
218	7	7425224916	19849065005	27274289921
219	8	12238934838	30848077052	43087011890
220	9			
221	10	8564920264	27786695233	36351615497
222	11	44355903	62479552	106835455
223	12	119717005	47751210	167468215
224	13	49398189	97741352	147139541
225	14	342620261	49164051	391784312
226	15	56666548	82523752	139190300
227	16	31463862	61262630	92726492
228	17	114513342	41422971	155936313
229	18	123284527	28422308	151706835
230	19	27079547	53376702	80456249

Table 5.3: Overview of the results from the 19 weeks traffic observation (last part)

5.3 Chronological Sequences of the weeks 10 to 19

The results presented in this sections are calculated for each week and pieced together manually. All values are the sums over an interval with the length of one hour. In figure 5.21 the number of active Zattoo hosts within an hour is shown. The differences between weekdays and weekends are clearly visible and also the daytime is observable. Here also the beginning of the summer holidays can be recognized at end of June, begin of July.

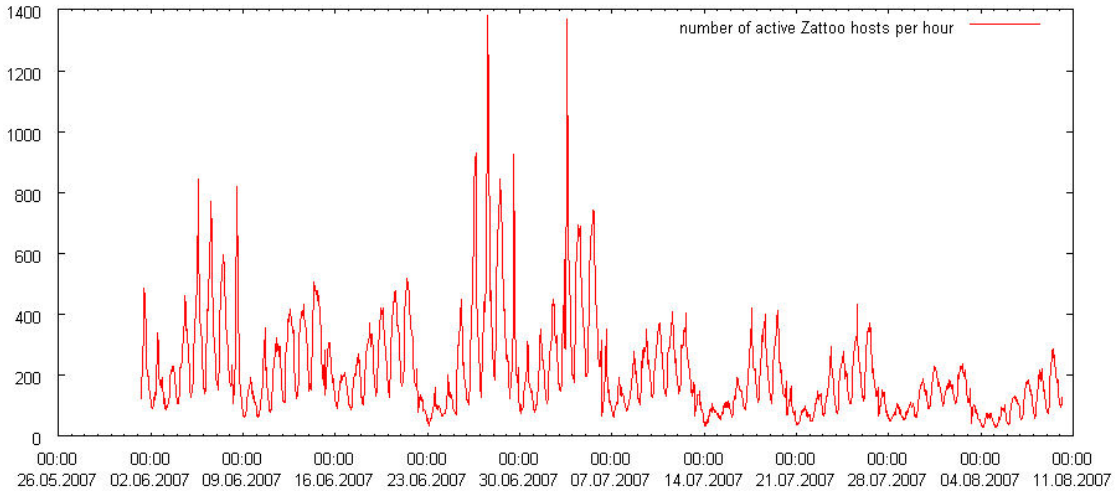

Figure 5.21: Active internal Zattoo hosts per hour

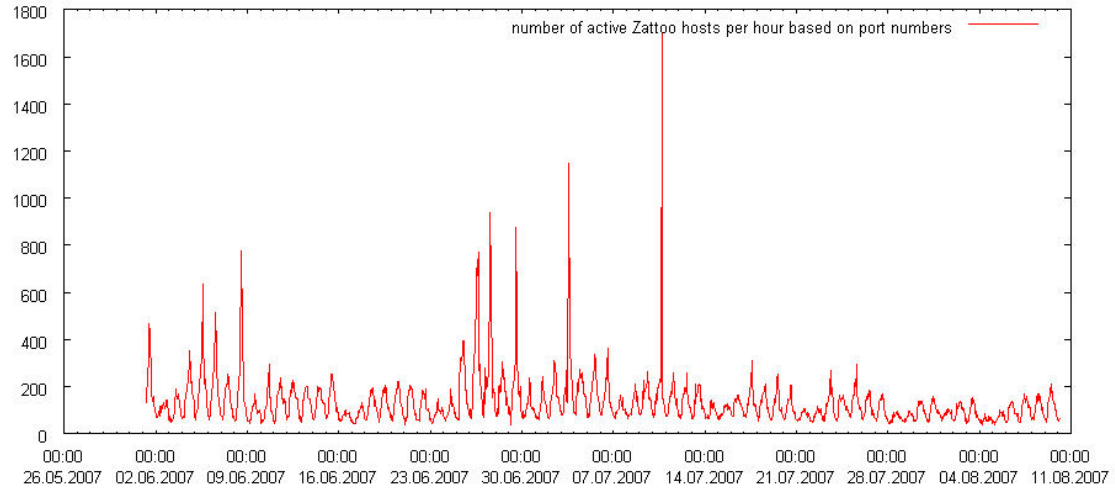

Figure 5.22: Active “portBased_zattoo_hosts” per hour

Tn comparison to the result of the port based approach, shown in figure 5.22, we can state that the results might be accurate, because of the same tendencies and because of the greater impact of human behavior as the differences due to daytimes and to weekdays in the results of our methodology.
We present in the figures 5.23, 5.24 and 5.25 overview of the Zattoo bytes, packets and flows per interval versus to total amount of transmitted bytes, packets and flows from the SWITCH network. In contrast to the figures 5.21 and 5.22 the weekdays and daytimes are more visibly in the overall sums from SWITCH than in the Zattoo traffic. If one compares the distances between the overall curves and the curves of the Zattoo traffic, it attract attention that these distances are very large for the flows, smaller for the packets and even smaller for the bytes. It seems to be, that the Zattoo flows have more bytes and packets per flow than a medial flow from the SWITCH network.

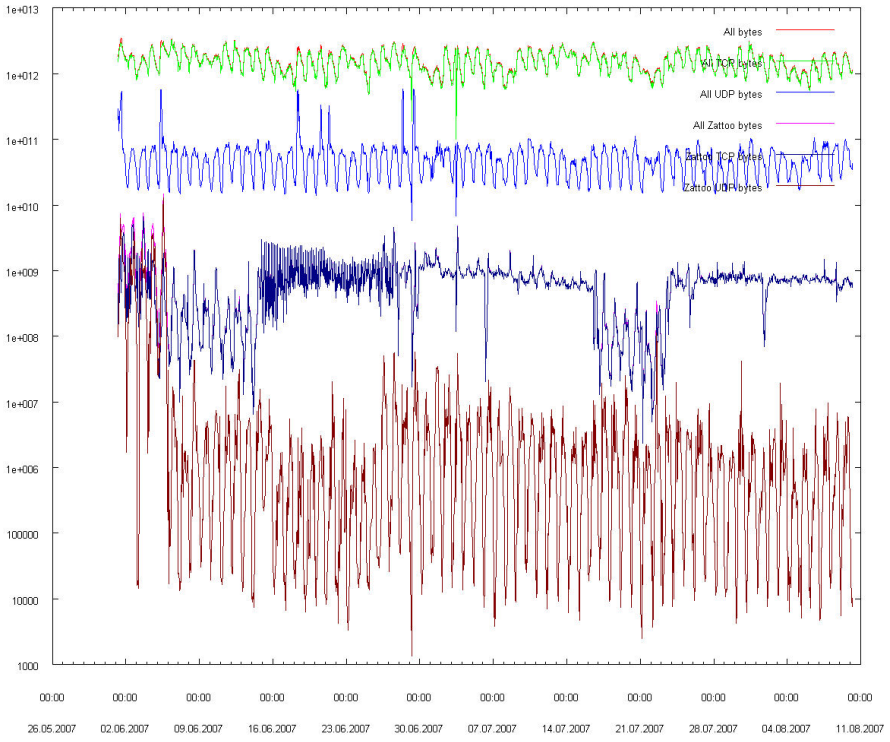

Figure 5.23: The temporal development of the Zattoo bytes in comparison to the total bytes

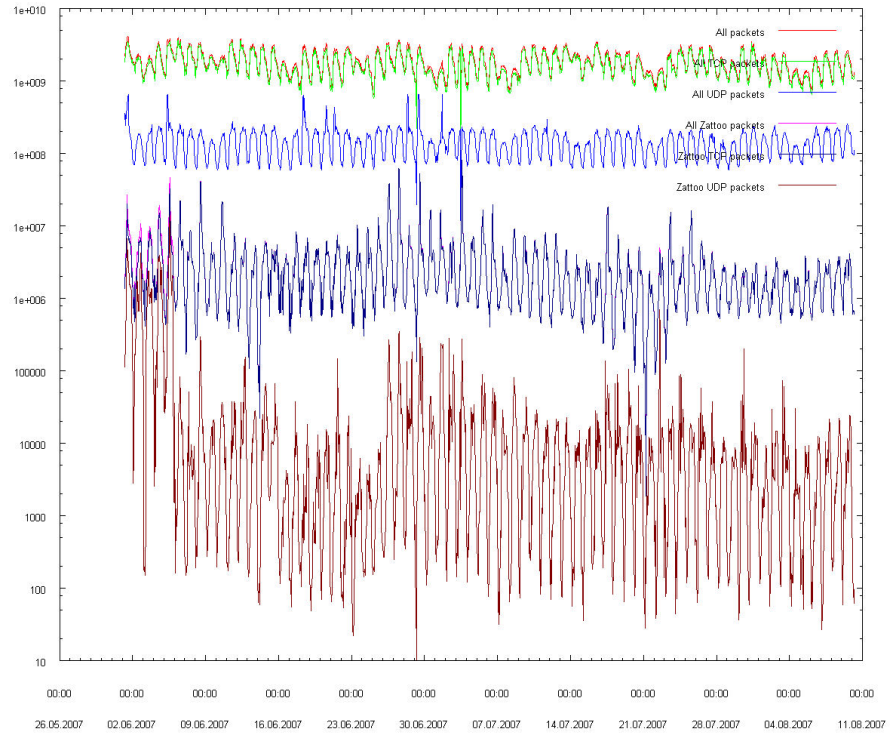

Figure 5.24: The temporal development of the Zattoo packets in comparison to the total packets

Because of the file size of the concatenated weekly results, images from the hourly distributions of Zattoo flows and hosts over the average packet size, the average packet inter-arrival time and the data rate can hardly be computed. MATLAB did not success on the computation, neither on a Lenovo T60, T2500 2GHz DualCore with 2 GB RAM nor on a node of the (new) TIK cluster, 2* DualCore Opteron 275 with 8 GB RAM. Only for the average packet size, some results could

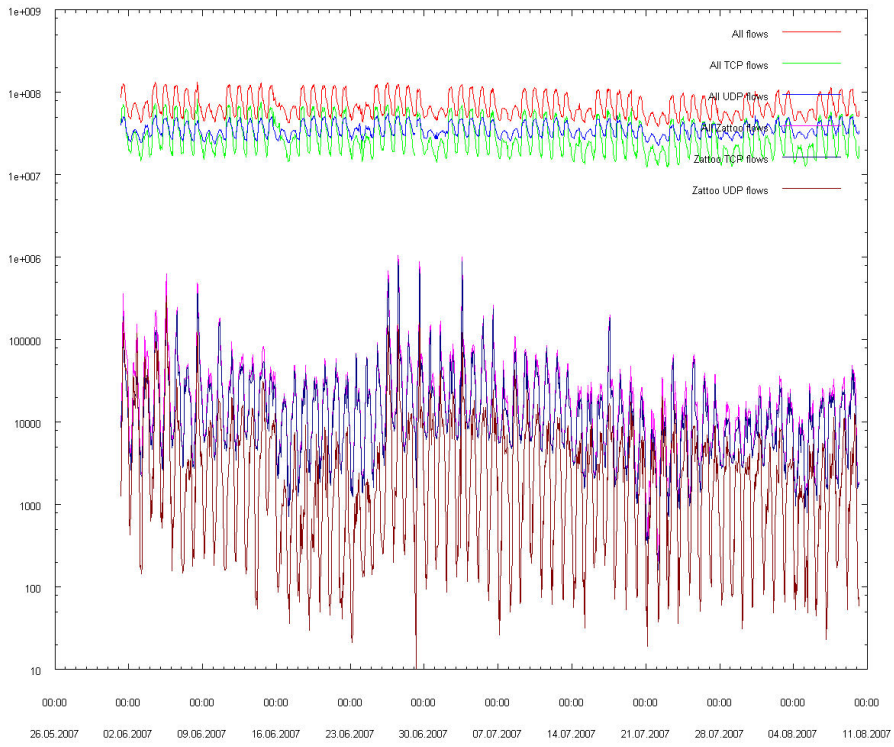

Figure 5.25: The temporal development of the Zattoo flows in comparison to the total flows

be gathered, but they only state, that distributions of the Zattoo flows and hosts is rather stable over the observed time period.

Chapter 6

Conclusion

6.1 Conclusions out of the survey of the existing approaches

Our main conclusion in this field is that the difficulties to detect an application build by application designers with knowledge about network traffic analysis and application detection are almost not assessable. It is always possible to detect one specific network application in the traffic, even if as most powerful technique active probing is used. But what if the developing team regularly releases an automated update which is might be able to introduce effective changes to the behavior of this application? In such a situation, the team which wants to detect the application has to be equally tough to release updates for their detection methodology.
In the usual case, where the goal of application designers is "only" to evade NATs and firewalls where the aim is to reach all users and not to be invisible under all circumstances, it is normally easier to detect the applications traffic. But also here, there is the same problem: The big number of newly emerged applications. P2P systems (file-sharing and streaming network for IPTV and VoIP applications), IM applications, etc. make it difficult for people who want to detect them, because of several reasons: Many new versions or update, encrypted traffic, not-open specifications and so on.
Independent of the type of application, there is always the necessity to describe the application's behavior from a network-layer view. But what is the network-layer view? Clearly all possible traffic attributes from table 2.1 and also all features from [47]. And if these possible candidate metrics for an application detection methodology are not enough, the possible infinite space of multi-flow features makes it even more difficult to decide which features can be adequate together with how much computational costs for the application which has to be detected. On the other side, dependent on the application, its behavior has to be described by parameters which are visible on the network-layer. This parameter are normally based on a priori knowledge, which makes it difficult to automate the generation of such application behavior.

6.2 Conclusions out of the IPTV detection

To design an application detection methodology which performs also in a network with the size of the SWITCH network, NetFlow data seems to be the best starting point. Even with this compression of the information carried in the packet headers the amount of data to process is huge! But on the other side, with this level of information many traffic features are not available or can only be approximated with some average. This makes it impossible to use many of the newer approaches, e.g. [20]. But overall, NetFlow seems to be a good trade-off between computational complexity, needed disk capacity and amount of information about the traffic.
Because we had no reference dataset or reference approach with the available dataset to calculate an accurate base truth for an evaluation of our detection methodology, we are clearly not in the position to judge if our approach is accurate enough or not. The only possibility we have, is to use a "classic" port number based method. With the results of this approach, which is widely

known as inaccurate, we have at least a second way to detect Zattoo traffic. By knowing that our second detection is not very precise, i.e. it might have a lot of false negatives and also some false positives, we can conclude that our results should at minimum have the same tendencies like the port number based detection results.
Our multi-flow approach has the advantage to decrease the number of false positives, since each host has a connection to Zattoo Inc in his history and therefore, no traffic from other applications should be marked as Zattoo traffic. On the other hand, the decision to have a connection history for the hosts, increases the consumption of memory dramatically. This is also the point, where we met our infrastructural limits and this problem was successfully solved by introducing the process of Aging.
Additionally, we found that the distributions of the Zattoo flows and the Zattoo hosts over metrics average packet size and average packet inter-arrival time can be used to instantiate further helpful thresholds for the Zattoo flows, which might make it possible to expand the detection with features which are independent of the port numbers.

Chapter 7

Outlook

A lot of work has to be done until one has the capabilities to build a systematic modeling for application detection. There are the two sides: the collected or real-time traffic data and the applications which have to be detected.
For both side, a description methodology has to be developed. For the collected traffic data, whatever it consists of payload traces or NetFolw data, such a description methodology has to be clear and understandable from the view of the application layer. On the other side, the attributes and behavior of the target applications as to be described that their representation in the model can be converted into terms witch can be expressed on the network layer.

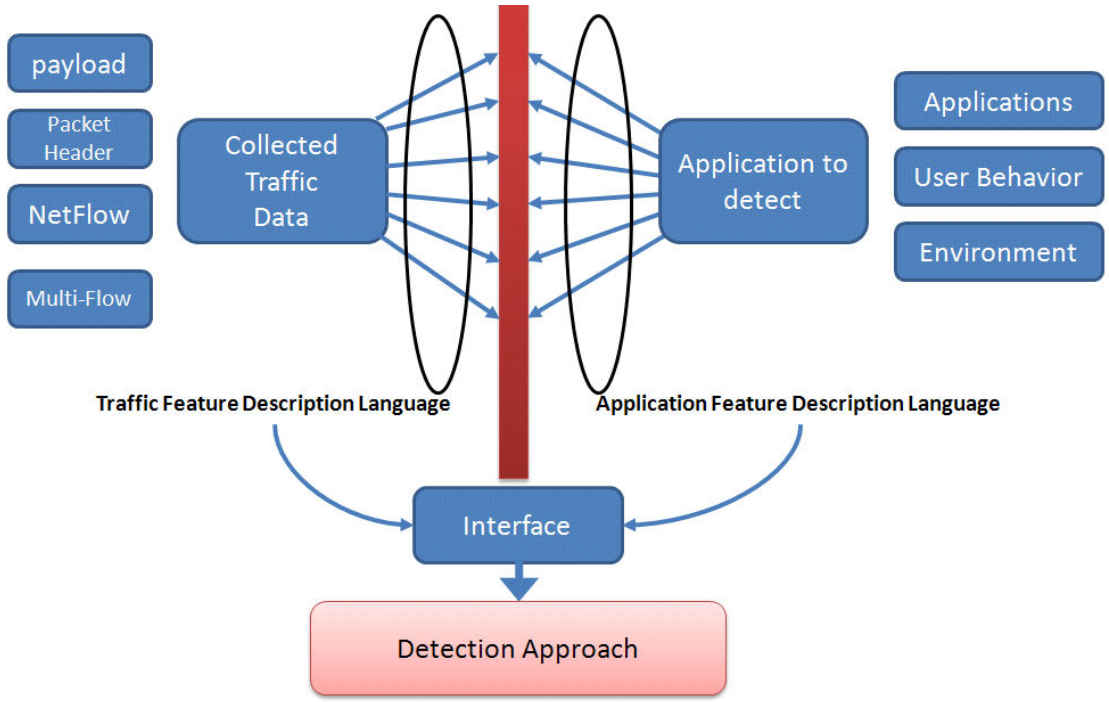

Figure 7.1: One possible architecture of a model for application detection

Figure 7.1 shows one simple possibility of how such a detection modeling can look like. If all behaviors of the traffic and the applications are abstracted into their description languages which can be read by an interface. Such an interface might put together the needs of both sides and than final construct a generic detection approach for the target applications, based on the available kind of traffic data.
Also for the detection of a P2P IPTV system, there is much work to do. The goal shout be to design a detection algorithm which is as independent as possible of any port numbers, so any P2P IPTV system can be detected, not only Zattoo. One way might be to used the knowledge about the flow properties of such a system from this work and combine it with [68].

Appendix A

Originalproblem

A.1 Introduction

Since the foundation of the modern Internet in 1992, it was subject to significant structural and technical changes. On the one hand, these changes are inflicted by its massive growth in users and content. But on the other hand, they are a result of the fact that the Internet is responsible for a constant output of new services and applications.
Furthermore, because the traffic mix in the Internet is a result of these services and applications, the detection of new, yet unknown applications or the identification and tracking of known ones is crucial to the following two (research) fields:

- **Traffic Engineering:** Traffic engineering uses statistical techniques to predict and engineer the behavior of telecommunications networks such as the Internet. It is mainly used for infrastructure planing and/or improving the quality of service by treating some specific types of traffic different than others. Examples are e.g. the growing demand for interconnection network bandwidth to support the sharing of large datasets in grid-computing (so called grid bulk transfers). A recent publication in this area is [54]. The publication presents a system for packet level based application identification in real-time(in combination with a protocol analyzer) with focus on applications for grid bulk transfers. This system could then be combined with approaches to re-engineer the bulk traffic onto dedicated resources. Other examples of applications that are of interest to traffic engineering are e.g. VoIP (e.g. Skype-Traffic [59] or IPTV.

- **(Network) Security:** Application identification methods are crucial factors in the following areas of (network) security:
  - Identifying unwanted traffic (e.g. in an enterprise network).
  - Identifying malicious applications (bot-software, spyware)
  - Identifying hosts that offer disallowed services or that run banned applications (e.g. Skype or p2p software).
  - Analysis of the update behavior of users (delay in downloading security patches).
  - Profiling of users.

Basically, application identification methods can be either active or passive. Active methods are based on sending probes to trigger application specific responses while passive methods require only wiretaps. Because of the fact that active probing is limited to testing a small set of hosts in a specific network, we set the focus on passive application identification methods.
Basically there are three widely-used approaches for passive application traffic identification:

- **Application signatures:** The application signature approach searches for application protocol specific patterns inside packet payloads. There are several publications on identification and evaluation of specific applications (e.g. [63] for p2p and [23] for chat applications) and on different types of signatures and signature matching from the area of intrusion detection on network level. But as proposed in [55], application signatures could also be extract automatically. Nevertheless, application signatures have some limitations. First, it is difficult if to quickly adapt to unknown, recently introduced application protocols since

the protocol state machine for each application should be known. Second, application-level pattern search in transport packets (usually achieved by reconstruction of individual flows), generates significant processing load. Application of such systems to higher-speed network links (1 Gb/s and higher) usually results in overload for the software, resulting in dropped packets. Therefore, this approach is not suitable for application detection in the large but it is often used to evaluate the accuracy of other methods (on low-speed links/in testbeds). [33] is an example of such a cross-evaluation for which the application signatures are available at [30] (for a more detailed technical report on this, see [34]). Another limitation of the signature based approach is that it does usually not work if applications use payload encryption. Nevertheless, if applications use a custom handshake procedure/protocol, they might still be identified (as shown in [55]).

- **Transport layer ports:** Application identification based on transport layer port information does not have the (potential) performance problems of the application signature approach and it can even be useful in case of encrypted payload. Nevertheless, this method requires human intervention in order to adapt to modified or recently introduced protocols. Another problem with this method is that many applications have begun to use ephemeral port numbers to deliberately avoid port-based identification. Some prominent publications using this approach: [60] analyzes p2p traffic across large networks and [24] analyzes the application traffic in a large IP network (sprint). That this method can highly underestimate the actual application traffic volume is shown in [31].

- **Network/transport layer application pattern recognition:** This approach uses a set of heuristics to map network traffic to a specific application. [31] and [32] use simple network/transport layer patterns like the simultaneous usage of UDP and TCP ports and the packet size distribution of an application flow between components of the application to identify p2p traffic.

  A more general heuristics that involves social, functional and application level heuristics is presented in [33]. As a means to combine the information from all three levels, the authors propose a graph based representation called ”graphlets”.

  Heuristics based methods can give good performance for existing application protocols and may even be used to discover unknown protocols. Nevertheless, two problems exist with this approach: it may be straightforward to construct a new application protocol that avoids any particular heuristic, and it may be difficult to eliminate false positives.

Furthermore, there exist some cross-layer approaches. [13] presents an approach for peer-to-peer traffic identification and optimization based on active networking and [23] combines application signatures and heuristics to identify chat applications (e.g. IRC or ICQ).
Depsite the fact that there is a lot of work on application identification, a coherent model to describe applications on the network level is still missing. A first step toward such a model should be done in this thesis. As a side-effect, some in-depth analysis of the development of two to three applications using our network traces from the SWITCH network will be provided. Any evaluations of this data has to be done on our computing cluster (confidentiality reasons).

Available Data

Usually, network traces used for application identification are either flow traces like e.g. specified by the Cisco NetFlow [1] format or packet traces (full-payload or packet headers only). Therefore, an important precondition to do research on application identification is to have such data at hand. In the course of the DDoSVax [75] project, an infrastructure to collect and store information about the network traffic crossing the borders of the Swiss Education and Research Network SWITCH [65] was set up at our institute. This dataset is a valuable source if the pattern for a specific application is known and if the spreading/use of this application is of interest.

A.1.1 Cluster and NetFlow Data Set

Facts about our NetFlow data set:

[1] This data contains e.g. information about which Internet hosts were connected to which others and how much data was exchanged over which protocols.

- **Time interval:** March 2003 - today
- **Hourly volume (compressed):** 500-2000 MB
- **Total volume (compressed) until 22.02.07:** 22 TB
- **Storage locations:**
  - Archive on jabba (tabe-library). Avg. download speed: 5 MBytes/s
  - Data of the last 20 days on aw3 (/aux/ddosvax/pulled/old/)
- **Completeness:** Only a few gaps (outages) or corrupt files (exact number unknown). A partial (incomplete) log already exists (Task: complete it).

Facts about our (new) cluster:

- **1 Login-Server (kom-pc-aw4):** AMD Athlon MP 2800+ (dual core), 2.1 GHz, 2 GB RAM (old file server)
- **1 Fileserver:** 2x DualCore Opteron 275 with 8 GB RAM and approx. 2.5 TB data storage (redundant, RAID 6).
- **5 Nodes:** 2x DualCore Opteron 275 with 8 GB RAM and 4 TB data storage (no backup, no redundancy!).
- **Network:** 1 GB Ethernet

A.2 The Task

This master thesis consists of the following two major tasks:

1. **Survey of existing approaches to application identification**
2. **A study of IPTV traffic in the SWITCH network**

The tasks and their subtasks are described in the following two subsections.

A.2.1 Survey of existing approaches to application identification

This task consists of three major subtasks:

- **Search and study related work:** An in-depth study of existing application identification models, algorithms and evaluation methodologies. The basis for this task is an exhaustive search and filtering of related work in the field of application detection, identification and classification. To start with, we give a small set of relevant papers that is to be extended with other work found in the Internet. We expect that this set is then reduced to the most relevant publications. As most relevant approaches we consider approaches that:
  - came up with a new algorithm/model/technique
  - successfully apply a algorithm/model/technique from a different field to application identification
  - identified significant characteristics of one or multiple applications (empirical and/or theoretical results)
  - applied novel evaluation methodologies
- **Derrive a set of characteristics and classification scheme:** After the study, a set of characteristics that captures the logical and technical differences of the existing application classification approaches as much as a scheme for classifying them has to be developed.
- **Compile a survey paper:** The most relevant approaches have to be evaluated and classified according to the characteristics and the classification scheme. The results are then compiled into a survey paper for publication in a scientific journal.

A.2.2 A study of IPTV traffic in the SWITCH network

This task consists of three major subtasks:

- **Study of IPTV applications:** Identify relevant IPTV services and applications used in the Internet. Study the behavior of these applications using publications on their protocols and communication behavior. Download and install the most interesting IPTV client(s) and study their behavior using online measurements.

- **Develop a detection algorithm:** The results obtained from the study of IPTV client(s) have to be compiled into a detection algorithm. The algorithm has to be implemented in C and should be well integrated in our toolset for evaluating Netflow data. The algorithm and it's implementation should be designed with time- and resource constraints in mind.
- **Analysis of IPTV traffic and refinement of the detection algorithm:** The last subtask consists of an analysis of the development of the IPTV traffic in the SWITCH network. For the analysis, a set of metrics that is likely to expose traffic characteristics that are specific to IPTV traffic is defined. It is then evaluated how the appearance of the IPTV traffic changed the overall traffic characteristics in the network. Furthermore, it is analyzed if the set of metrics contains metrics that can be used to refine the detection algorithm in favor of higher accuracy or in order to depend less on a priori knowledge.

A.3 Deliverables

During this thesis the following deliverables will be produced:

1. A concise and detailed documentation of the conducted work.
2. The code and installation instructions for the analysis tools.
3. A ready-to-use installation on our computing cluster along with its documentation.

A.3.1 Documentation structure

The report should contain the steps conducted along with the findings/consequences/results, lessons learned, summary and an outlook on future work [2]. All designs should be described in detail and design decisions should be motivated. Evaluations have to be academically sound. In case of measurements of stochastic quantities, this means e.g. multiple runs and indication of confidence intervals. Finally, the report should contain an addendum with installation and usage instructions for the developed tools.
The code of all of the in this thesis developed tools should come along with a detailed documentation. The code should follow a coherent and clean coding and commenting style so that the code is easy to understand and use. If applicable, the documentation could be generated using automated documentation tools (e.g. doxygen).
If important new research results are found, a paper might be written as an extract of the thesis and submitted to a computer network or security conference.

A.4 General Information

Dates/Meetings:

- This master thesis starts on Monday, 19.03.2007 and finishes on Monday, 19.09.2007. It lasts six months in total.
- Informal meetings with the tutors will be held at least once a week.

Presentations:

- Two intermediate informal presentations for Prof. Plattner and all tutors will be scheduled two and four months into the thesis.
- A final presentation at TIK will be scheduled close to the completion date of the thesis.

Supervisors:

Tutor: Bernhard Tellenbach, tellenbach@tik.ee.ethz.ch +41 44 632 70 06, ETZ G97
Co-Tutor: Daniela Brauckhoff, brauckhoff@tik.ee.ethz.ch, +41 44 632 70 50, ETZ G97
//Co-Tutor: Arno Wagner, wagner@tik.ee.ethz.ch, +41 44 632 70 04 , ETZ G95

[2] Additional information and tips are available on the thesis wiki at `http://tikiwiki.ethz.ch/thesis/index.php/Main/HomePage`.

Appendix B

Zattoo detection Algorithm

Here, we show some important parts of the source code of our detection methodology for the Zattoo IPTV application.

B.1 Data Structures for the Zattoo Hosts

B.1.1 Internal Zattoo Host

```
struct zattoo_host {
  uint32_t   addr;
  uint32_t   nbr;
  long long  connections_to_zattoo;
  long long  connections_to_zattoo_Ad;

  long long  First;
  long long  Last;

  long long  nbr_of_received_udp_flows;
  long long  nbr_of_received_udp_bytes;
  long long  nbr_of_received_udp_packets;
  long long  duration_of_received_udp_connections;

  long long  nbr_of_sent_udp_flows;
  long long  nbr_of_sent_udp_bytes;
  long long  nbr_of_sent_udp_packets;
  long long  duration_of_sent_udp_connections;

  long long  nbr_of_received_tcp_flows;
  long long  nbr_of_received_tcp_bytes;
  long long  nbr_of_received_tcp_packets;
  long long  duration_of_received_tcp_connections;

  long long  nbr_of_sent_tcp_flows;
  long long  nbr_of_sent_tcp_bytes;
  long long  nbr_of_sent_tcp_packets;
  long long  duration_of_sent_tcp_connections;

  long long  otherPorts_nbr_of_received_udp_flows;
  long long  otherPorts_nbr_of_received_udp_bytes;
  long long  otherPorts_nbr_of_received_udp_packets;
  long long  otherPorts_duration_of_received_udp_connections;

  long long  otherPorts_nbr_of_sent_udp_flows;
  long long  otherPorts_nbr_of_sent_udp_bytes;
```

```
  long long  otherPorts_nbr_of_sent_udp_packets;
  long long  otherPorts_duration_of_sent_udp_connections;

  long long  otherPorts_nbr_of_received_tcp_flows;
  long long  otherPorts_nbr_of_received_tcp_bytes;
  long long  otherPorts_nbr_of_received_tcp_packets;
  long long  otherPorts_duration_of_received_tcp_connections;

  long long  otherPorts_nbr_of_sent_tcp_flows;
  long long  otherPorts_nbr_of_sent_tcp_bytes;
  long long  otherPorts_nbr_of_sent_tcp_packets;
  long long  otherPorts_duration_of_sent_tcp_connections;

  double mean_Rate_TCP;
  double S_Rate_TCP;
  double variance_Rate_TCP;

  double mean_Rate_UDP;
  double S_Rate_UDP;
  double variance_Rate_UDP;

  double mean_IAT_TCP;
  double S_IAT_TCP;
  double variance_IAT_TCP;

  double mean_IAT_UDP;
  double S_IAT_UDP;
  double variance_IAT_UDP;

  double mean_AverPacketSize_TCP;
  double S_AverPacketSize_TCP;
  double variance_AverPacketSize_TCP;

  double mean_AverPacketSize_UDP;
  double S_AverPacketSize_UDP;
  double variance_AverPacketSize_UDP;

  uint16_t   udp_port;
  uint16_t   tcp_port;

  struct hashed_table * externalHostTablePerInternal;
};
```

B.1.2 Externals Zattoo Host

```
struct externalZattooHost {
uint32_t   addr;
long long Last;

long long receivedTCP_flows_WithInternalZattooHosts;
long long receivedTCP_packets_WithInternalZattooHosts;
long long receivedTCP_bytes_WithInternalZattooHosts;

long long sentTCP_flows_WithInternalZattooHosts;
long long sentTCP_packets_WithInternalZattooHosts;
long long sentTCP_bytes_WithInternalZattooHosts;

long long receivedUDP_flows_WithInternalZattooHosts;
```

```
long long receivedUDP_packets_WithInternalZattooHosts;
long long receivedUDP_bytes_WithInternalZattooHosts;

long long sentUDP_flows_WithInternalZattooHosts;
long long sentUDP_packets_WithInternalZattooHosts;
long long sentUDP_bytes_WithInternalZattooHosts;

long long otherPorts_receivedTCP_flows_WithInternalZattooHosts;
long long otherPorts_receivedTCP_packets_WithInternalZattooHosts;
long long otherPorts_receivedTCP_bytes_WithInternalZattooHosts;

long long otherPorts_sentTCP_flows_WithInternalZattooHosts;
long long otherPorts_sentTCP_packets_WithInternalZattooHosts;
long long otherPorts_sentTCP_bytes_WithInternalZattooHosts;

long long otherPorts_receivedUDP_flows_WithInternalZattooHosts;
long long otherPorts_receivedUDP_packets_WithInternalZattooHosts;
long long otherPorts_receivedUDP_bytes_WithInternalZattooHosts;

long long otherPorts_sentUDP_flows_WithInternalZattooHosts;
long long otherPorts_sentUDP_packets_WithInternalZattooHosts;
long long otherPorts_sentUDP_bytes_WithInternalZattooHosts;

struct hashed_table * internalHostTablePerExternal;
};
```

B.1.3 Port-based Zattoo Host

```
struct portBased_zattoo_host {
  uint32_t   addr;
  uint32_t   nbr;

  long long  Last;

  long long  nbr_of_received_udp_flows;
  long long  nbr_of_received_udp_bytes;
  long long  nbr_of_received_udp_packets;
  long long  duration_of_received_udp_connections;

  long long  nbr_of_sent_udp_flows;
  long long  nbr_of_sent_udp_bytes;
  long long  nbr_of_sent_udp_packets;
  long long  duration_of_sent_udp_connections;

  long long  nbr_of_received_tcp_flows;
  long long  nbr_of_received_tcp_bytes;
  long long  nbr_of_received_tcp_packets;
  long long  duration_of_received_tcp_connections;

  long long  nbr_of_sent_tcp_flows;
  long long  nbr_of_sent_tcp_bytes;
  long long  nbr_of_sent_tcp_packets;
  long long  duration_of_sent_tcp_connections;

};
```

B.2 Algorithm to calculate the Variance iteratively

```
n = 0
mean = 0
S = 0

foreach x in data:
  n = n + 1
  delta = x - mean
  mean = mean + delta/n
  S = S + delta*(x - mean)      // This expression uses the new value of mean
end for

variance = S/(n - 1)
```

Where n denotes the for us the number of flows and x is the actual value of the flow, e.g. average packet size, packet inter-arrival time or the data rate. This algorithm is taken from Knuth [40].

Bibliography

[1] Shaikh Anees, Jennifer Rexford, and Kang Shin. Load-sensitive routing of long-lived ip flows. Proceedings of the ACM SIGCOMM, ACM, August 1999.

[2] Ares. http://www.softgap.com.

[3] Chadi Barakat, Patrick Thiran, Gianluca Iannaccone, Christophe Diot, and Phillippe Owezarski. Modeling internet backbone traffic at the flow level. *IEEE TRANSACTIONS ON SIGNAL PROCESSING,*, 51, 2003.

[4] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Inherent behaviors for on-line detection of peer-to-peer file sharing, 2006.

[5] Salman Baset and Henning Schulzrinne. An analysis of the skype peer-topeer internet telephony protocol, 2004.

[6] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36, 2006.

[7] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. Early application identification, 2007.

[8] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 159–164, New York, NY, USA, 2006. ACM Press.

[9] CacheLogic. http://www.cachelogic.com.

[10] Bram Cohen. Bittorrent, http://www.bitconjurer.org/bittorrent.

[11] Fivos Constantinou and Panayiotis Mavrommatis. Identifying known and unknown peer-to-peer traffic, 2005.

[12] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Sagarelli. Traffic classification through simple statistical fingerprinting. *SIGCOM 2007*, 2007.

[13] Ivan Dedinsk, Hermann De Meer, Liangxiu Han, Laurent Mathy, Dimitrios P. Pezaros, Joe S. Sventek, and Zhan Xiaoying. Cross-layer peer-to-peer traffic identification and optimization based on active networking. In *7th Annual International Working Conference on Active and Programmable Networks (IWAN'05)*, Sophia Antipolis, French Riviera, France, November 2005.

[14] DirectConnect. Direct connect, http://www.neo-modus.com.

[15] Dinil Mon Divakaran, Hema A. Murthy, and Timothy A. Gonsalves. Traffic modeling and classification using packet train length and packet train size, 2006.

[16] eDonkey 2000. http://www.edonkey2000.com.

[17] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms, 2006.

[18] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt. Internet traffic identification using machine learning, 2006.

[19] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt. Byte me: A case for byte accuracy in traffic classification, June 2007.

[20] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Williamson. Offline/realtime traffic classification using semi-supervised learning, 2007.

[21] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, and Carey Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core.

[22] ethereal. http://www.ethereal.com.

[23] Christian Dewes Arne Wichmann Anja Feldmann. An analysis of internet chat systems. In *Internet Measurement Conference 2003*, October 2003.

[24] Chuck Fraleigh, Sue Moon, Bryan Lyles, Chase Cotton, Mujahid Khan, Deb Moll, Rob Rockell, Ted Seely, and Christophe Diot. Packet-level traffic measurements from the sprint ip backbone. *IEEE Network*, 17:6–16, December 2003.

[25] GnutellaHosts. http://www.gnutellahosts.com.

[26] David A. Helder and Sugih Jamin. End-host multicast communication using switch-tree protocols. Proceedings of the Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC), May 2002.

[27] port numbers IANA. http://www.iana.org/assignments/port-numbers.

[28] ietf. http://www.ietf.org/html.charters/ipfix-charter.html.

[29] Zattoo Inc. http://www.zattoo.com.

[30] Thomas Karagiannis. Application-specific payload bit strings. `http://www.cs.ucr.edu/~tkarag/papers/strings.txt`, November 2004.

[31] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kc Claffy, and Michalis Faloutsos. File-sharing in the internet: A characterization of p2p traffic in the backbone. Technical report, UC Riverside, CAIDA and SDSC, 2003.

[32] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc Claffy. Transport layer identification of p2p traffic, 2004.

[33] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark, August 2005.

[34] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark. technical report„ 2005.

[35] Thomas Karagiannis, Kostantina Papagiannaki, Nina Taft, and Michalis Faloutsos. Profiling the end host, 2007.

[36] KaZaA. http://www.kazaa.com.

[37] Myung-Sup Kim, Young J. Won, and James Won-Ki Hong. Application-level traffic monitoring and an analysis on ip networks. *ETRI Journal*, 27, 2004.

[38] Myung-Sup Kim, Young J. Won, and James Won-Ki Hong. Characteristic analysis of internet traffic from the perspective of flows. *Computer Communications 29 (2006)*, 2005.

[39] Andreas Kind, Dieter Gantenbein, and Hiroaki Etoh. Relationship discovery with netflow to enable business-driven it management, 2006.

[40] Donald E. Knuth. *The art of Computer Programming*, volume 2: Seminumerical Algorithms. 1998.

[41] Kun-chan Lan and John Heidemann. A measurement study of correlations of internet flow characteristics. *Computer Networks*, 50, 2006.

[42] Nikitas Liogkas, Robert Nelson, and Lixia Zhang. Exploiting bittorrent for fun (but not profit).

[43] Anukool Makhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. *SIGCOMM 05*, 2005.

[44] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. Flow clustering using machine learning techniques, 2004.

[45] Andrew W. Moore, Tom Auld, and Stephen F Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 17, 2006.

[46] Andrew W. Moore and Konstantina Papagiannaki. Toward the accurate identication of network applications, 2005.

[47] Andrew W. Moore, Denis Zuev, and Michael L. Crogan. Discriminators for use in flow-based classification, 2005.

[48] MP2P. http://www.slyck.com/mp2p.php.

[49] napster. http://www.napster.com.

[50] Cisco Netflow. *White Paper: NetFlow Services and Applications.*, 2003.

[51] Andrew M. Odlyzko. Internet traffic growth: Sources and implications. 2003.

[52] Satoshi Ohzahata, Yoichi Hagiwara, Matsuaki Terada, and Konosuke Kawashima. A traffic identification method and evaluations for a pure p2p application, 2005.

[53] J. Paisley and J. Sventek. Real-time detection of grid bulk transfer traffic. In 10th IEEE/IFIP Network Operations and Management Symposium„ April 2006.

[54] Jonathan Paisley and Joseph Sventek. Real-time detection of grid bulk transfer traffic. In *10th IEEE/IFIP Network Operations and Management Symposium*, April 2006.

[55] Oliver Spatscheck Patrick Haffner, Subhabrata Sen and Dongmei Wang. Acas: automated construction of application signatures, 2005.

[56] Zattoo Presentation. http://swinog.ch/meetings/swinog14/zattoo_swinog-14.pdf.

[57] rfc 3917. http://www.ietf.org/rfc/rfc3917.txt.

[58] Matthew Roughan, Subhabrata Sen, Oliver Spatschek, and Nick Duffield. Class-of-service mapping fo qos: A statistical signature-based approachto ip traffic classification. 2004.

[59] Neil Daswani Saikat Guha and Ravi Jain. An experimental study of the skype peer-to-peer voip system, February 2006.

[60] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *Analyzing peer-to-peer traffic across large networks*, 12(2):219–232, April 2004.

[61] Skype. http://www.skype.com.

[62] Soulseek. http://www.slsknet.org.

[63] Oliver Spatscheck Subhabrata Sen and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures, May 2004.

[64] Swait.org. http://swait.org/bots/.

[65] SWITCH. Swiss academic and research network. `http://www.switch.ch/`, 2006.

[66] tcpdump. http://www.tcpdump.org.

[67] tcptrace. http://www.tcptrace.org.

[68] Arno Wagner, Thomas Duebendorfer, Lukas Haemmerle, and Bernhard Plattner. Flow-based identification of p2p heavy-hitters, 2006.

[69] Wenjie Wang, Hyunseok Chang, Amgad Zeitoun, and Sugih Jamin. Characterizing guarded hosts in peer-to-peer file sharing systems. In *In Proceedings of IEEE Global Communications Conference (Globecom 2004)—Global Internet and Next Generation Networks, Nov 2004, Dallas, USA*.

[70] Wenjie Wang, Ye Du, and Sugih Jamin. Improving resiliency of overlay networks for streaming applications. Technical Report CSE-TR-523-06, EECS Department, University of Michigan, 2006.

[71] Wenjie Wang, David Helder, Sugih Jamin, and Lixia Zhang. Overlay optimizations for end-host multicast. In Proceedings of Fourth International Workshop on Networked Group Communication (NGC), Oct 2002.

[72] Wenjie Wang, Cheng Jin, and Sugih Jamin. Network overlay construction under limited end-to-end addressability. Technical Report CSE-TR-489-04, EECS Department, University of Michigan, 2004.

[73] Wenjie Wang, Cheng Jin, and Sugih Jamin. Network overlay construction under limited end-to-end reachability. Proceedings of IEEE INFOCOM 2005, March 2005, Miami, USA.

[74] WinMx. http://www.winmx.com.

[75] www.tik.ethz.ch. Ddosvax. `http://www.tik.ee.ethz.ch/~ddosvax/`.

[76] Sebastian Zahnder, Thuy Nguyen, and Grenville Armitage. Self-learning ip traffic classification based on statistical flow characteristics, 2005.

[77] Beichuan Zhang, Sugih Jamin, and Lixia Zhang. Host multicast: A framework for delivering multicast to end users. IEEE INFOCOM 2002.s.

[78] Beichuan Zhang, Sugih Jamin, and Lixia Zhang. Universal ip multicast delivery. In Proceedings of Fourth International Workshop on Networked Group Communication (NGC), Oct 2002.

[79] Beichuan Zhang, Wenjie Wang, Sugih Jamin, Daniel Massey, and Lixia Zhang. Universal ip multicast delivery. Computer Networks 50(6): 781-806, 2006.