UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

# Detection of Encrypted Traffic Generated by Peer-to-Peer Live Streaming Applications Using Deep Packet Inspection

**André Filipe Ferreira Esteves**

Dissertation submitted in candidature for the degree of Master of Science in
**Computer Science and Engineering**
(2nd Cycle Degree)

Supervisor: Dr. Mário Marques Freire

**Covilhã, October 2011**

# Acknowledgements

First i would like to start thanking the person who give the chance to integrate his dynamic research team were work and fun walk side by side, to professor Mário Freire my gratitude, for all the scientific knowledge acquired along this work, for the motivation, and also for the almost unlimited resources provided to make this research possible.

I would also like to thank David Milheiro for the help mounting the research test-bed, and João Gomes and professor Pedro Inácio for the great suggestions to improve my work. I am also grateful to the University of Beira Interior, in special to the Computer Science Department and to the Multimedia Signal Processing research group for all the support and equipment provided for this work, and also for having such delight place to work.

To my parents and my brother for all the motivation and support over the years, as without them i would not have made here. For all the work they have put on themselves so i could finish my academic studies. To my brother for making me realize that i could take a little break once a time. To my girlfriend who helped me in every way she could, either giving ideas or correcting my English. To her, my eternal love.

For last, but not the least to all my friends who supported me along this work: David Monteiro, Rui Raposo, Rui Brás, Hugo Santos, André Gomes, Bruno Campos, and to all the others, my sincere gratitude.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia through the project TRAMANET: Traffic and Trust Management in Peer-to-Peer Networks, with contracts PTDC/EIA/73072/2006 and FCOMP-01-0124-FEDER-007253.

# Resumo Alargado

A popularidade e o número de aplicações que usam o paradigma de redes par-a-par (P2P) têm crescido substancialmente na última década. Estas aplicações deixaram de serem usadas simplesmente para partilha de ficheiros e são agora usadas também para distribuir conteúdo multimédia. Hoje em dia, estas aplicações têm meios de cifrar o conteúdo da comunicação ou empregar técnicas de ofuscação directamente no protocolo. Nesta dissertação, foi realizada uma investigação para identificar fluxos de tráfego encriptados, que foram gerados por três aplicações populares de distribuição de conteúdo multimédia em redes P2P: TVUPlayer, Livestation e GoalBit. Para este trabalho, foi criada uma plataforma de testes que pretendia simular um cenário quase real, e o tráfego que foi capturado, continha uma grande variedade de aplicações. O método proposto nesta dissertação recorre à técnica de Inspecção Profunda de Pacotes (DPI), e por isso, foi necessário analisar o conteúdo dos pacotes a fim de encontrar padrões que se repetissem, e que iriam mais tarde ser usados para criar um conjunto de regras SNORT para detecção de pacotes chave na rede, gerados por estas aplicações, afim de se poder correctamente classificar os fluxos de tráfego. Após descobrir que a aplicação Livestation deixou de funcionar com P2P, apenas as duas regras criadas até esse momento foram usadas. Quanto à aplicação TVUPlayer, foram criadas várias regras a partir do tráfego gerado por ela mesma e que tiveram uma boa taxa de precisão. Várias regras foram também criadas para a aplicação GoalBit em que foram usados quatro cenários: com e sem encriptação usando a opção de transmissão tracker, e com e sem encriptação usando a opção de transmissão sem necessidade de tracker (aqui foi usado o protocolo Kademlia). O método foi avaliado experimentalmente na plataforma de testes criada para o efeito, sendo demonstrado que a precisão do conjunto de regras para a aplicação GoalBit é de 97%.

# Abstract

The number of applications using the peer-to-peer (P2P) networking paradigm and their popularity has substantially grown over the last decade. They evolved from the file-sharing applications to media streaming ones. Nowadays these applications commonly encrypt the communication contents or employ protocol obfuscation techniques. In this dissertation, it was conducted an investigation to identify encrypted traffic flows generated by three of the most popular P2P live streaming applications: TVUPlayer, Livestation and GoalBit. For this work, a test-bed that could simulate a near real scenario was created, and traffic was captured from a great variety of applications. The method proposed resort to Deep Packet Inspection (DPI), so we needed to analyse the payload of the packets in order to find repeated patterns, that later were used to create a set of SNORT rules that can be used to detect key network packets generated by these applications. The method was evaluated experimentally on the test-bed created for that purpose, being shown that its accuracy is of 97% for GoalBit.

**Keywords**: Live Streaming, Media Streaming, Peer-to-Peer Streaming

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Focus and Scope

Peer-to-peer (P2P) networks appeared as a new paradigm after the success of a popular music sharing application, called Napster [1], which allowed the download of music through the computer of each user. However, the way Napster was implemented had some major drawbacks: the first was the single point of failure due to the centralized server that managed the connections; the second was the copyright infringements due to the unauthorized distribution of illegal content. After Napster lawsuits to shutdown, new P2P systems like Gnutella and Fastrack [2, 3] emerged without the need of a central server, since any computer system with an Internet connection and using this kind of applications could provide additional resources or services with low cost support, since no extra hardware was needed. In addition of being easy to manage, this kind of systems is tolerant to faults due to its decentralized approach, since if a peer fails one still may get the information from other peer in the network. Another advantage is the easiness of installing the needed software. However, there are also drawbacks due to the use of P2P systems. One is associated with the way performance may be affected when resources are accessed too many times, since, in that cases, users may notice a performance degradation. Another concern is the security of a P2P system, because one cannot guarantee that all peers will administrate their machine appropriately in order to do not share illegal material. Besides the main concerns related to P2P systems, another issue is how to control so much information and with that, becomes difficult to maintain a backup of that information.

Recently, it was observed that P2P file-sharing, as a percentage of the overall Internet traffic, has begun to decrease [3]. In this study, it was reported that P2P file-sharing had a huge decline between 2007 and 2009, mainly due to law infringements, traffic shaping in organizations, and specially due to more legal and convenient ways of file sharing like Megaupload [4], RapidShare [5], Netflix [6], etc. Another contribution for P2P file-sharing decline was due to security issues, like malware infections, files that carry viruses and trojan horses, and many other issues concerning the security of P2P systems. Although a decline of the percentage has been noticed, the volume of P2P file-sharing traffic has increased due to larger sizes of the files transferred.

In [3, 7] it is shown that video traffic is increasing and is expected to be the dominant Internet activity in the next few years. As one can see from the success of Youtube [8], on-line video is growing and will keep on growing trough video-streaming, video-on-demand and videoconferences, among other possibilities. Due to the advantage of the scalability of P2P systems against the restrictions of multicast networks, it has been investigated the possibility of dividing multimedia content and streaming it into the Internet using peer-to-peer technology [8, 9, 10, 11]. Multicast networks could also be used to stream a video over the Internet, but the limited scalability, the expensive costs, and the fact that multicast networks are restricted to infrastructures managed by single network operators, put restrictions on that media delivery.

By using P2P, users can watch their favourite movie and still upload parts of the movie to other users so they can also watch, and so forward. P2P media streaming uses the data sharing mode of "watching while it makes download", and that may have some disadvantages like, as peers are constantly entering and leaving the network they may affect the bandwidth, the use of an overlay topology that have a single point of failure causing the rest of the peers to stay without watching the stream, and of course the peer bandwidth can result in a good quality stream or a bad one. Live P2P media streaming systems can be classified into three categories based on the overlay network structure: tree-based, mesh-based and a hybrid combination of the previous two.

The tree-based systems have a well organized tree structure were peers join the network in a certain tree level and receive the video from their parent peer and forward the received video to its children peers [12]. The major disadvantage of this system is their vulnerability to peer churn. When a parent peer leaves the network all of its children (all subtree below the peer that leaves will be affected) will temporarily see their video delivery disrupted. Another problem with this kind of systems is the peer bandwidth efficiency degradation due to the fact leaf nodes that are a large portion of peers in the network do not contribute with their uploading bandwidth and to address this problem multi-tree based approaches have been proposed. In multi-tree systems, the streaming is divided into multiple sub-streams creating therefore multiple sub-trees one for each sub-stream. Figure 1.1 is an example of a peer-to-peer tree-based streaming system.

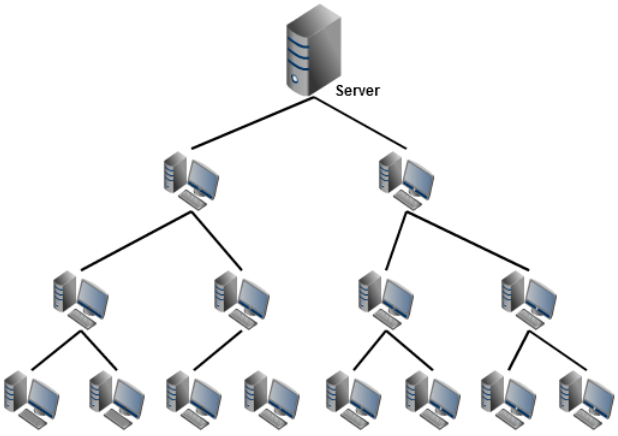


Figure 1.1: Schematic representation of a P2P tree-based system.

In a mesh-based system there is no strict topology. Here peers are distributed based on their content and bandwidth availability. A peer can download/upload from and to multiple neighbours simultaneously and if one peer leaves, the peer can still download from the remaining neighbours. The working principle is similar to the one of BitTorrent [13], i.e. , a node register to the system and receive a list of addresses from a tracker, the tracker is nothing more than a super-node which tracks peers that are downloading and peers that downloaded a file. After its registration, a node contacts the peers from the list the tracker gives to it and receive a list of chunks that they have and that are able to share. Nevertheless, besides the high reliability and availability of a mesh-based system, maintaining this kind of systems is more expensive than a tree-based one. TVUplayer [14], and GoalBit[15] are examples of mesh-based P2P streaming applications. Figure 1.2 presents a schematic representation of a mesh-based system. Hybrid

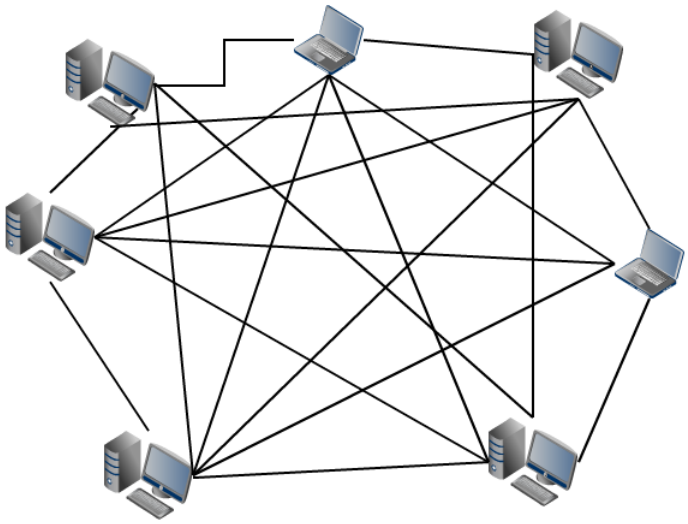

Figure 1.2: Schematic representation of a P2P mesh-based system.

systems combine the best features of tree-based and mesh-based structures [16]. They have high reliability as mesh-based structures due to the possibility of making multiple links between any two nodes, got lower delay as in tree-based structures, and got lower overhead when compared to mesh-based structures. The main disadvantage is the high topology maintenance cost, when compared with the low cost and easy implementation of a tree-based structure. All this evolution in media streaming led to a high network traffic volume, which in turn led to network performance issues. These issues include network congestion, bad or insufficient QoS in media streaming services, and bandwidth demands. The use of P2P media streaming, if not suitably authorized, may lead to a degradation of the network, which may compromising the performance of critical networked applications or network-based tasks performed in a giving institution. A study presented in [17] contains both fixed and mobile data providers and possible by the voluntary participation of Sandvine [18]. In this study, Sandvine concluded, that real-time entertainment (which includes P2P live streaming and Video on Demand (VoD) streaming) has grown over the last years, starting to be a dominant category in several regions presented in this study, even surpassing P2P file sharing traffic quota. It should be noted that this study encompasses fixed and mobile networks. For example in North America, it was observed that real-time entertainment was the largest contributor to data consumption on both fixed and mobile access (43% of peak period traffic and 41% respectively) networks. As for Asia-Pacific region, due to the success of the peercasting applications, such as PPStream and PPLive, P2P live streaming and VoD streaming had an increase of traffic volume. Nonetheless, P2P file-sharing continues to be a major component of traffic and exhibits remarkably consistent upstream levels throughout over a 24 hour period.

The study states that Latin America continues to have a high traffic volume of P2P file-sharing, but real-time entertainment is emerging as the dominant source of content, showing that the behaviour of the subscriber is shifting towards favouring on-demand applications. At last, in Europe, Web browsing is the category that accounts for more byte usage. Meanwhile, BitTorrent is the most used P2P filesharing protocol (this is the case everywhere except in Latin America, where Ares protocol prevails), representing almost 30% of upstream peak period traffic and

slightly more than 8% of downstream peak period traffic. Figure 1.3 shows the traffic growth of real-time entertainment between 2009 and 2010 during peak hours and with a fixed access in North America.

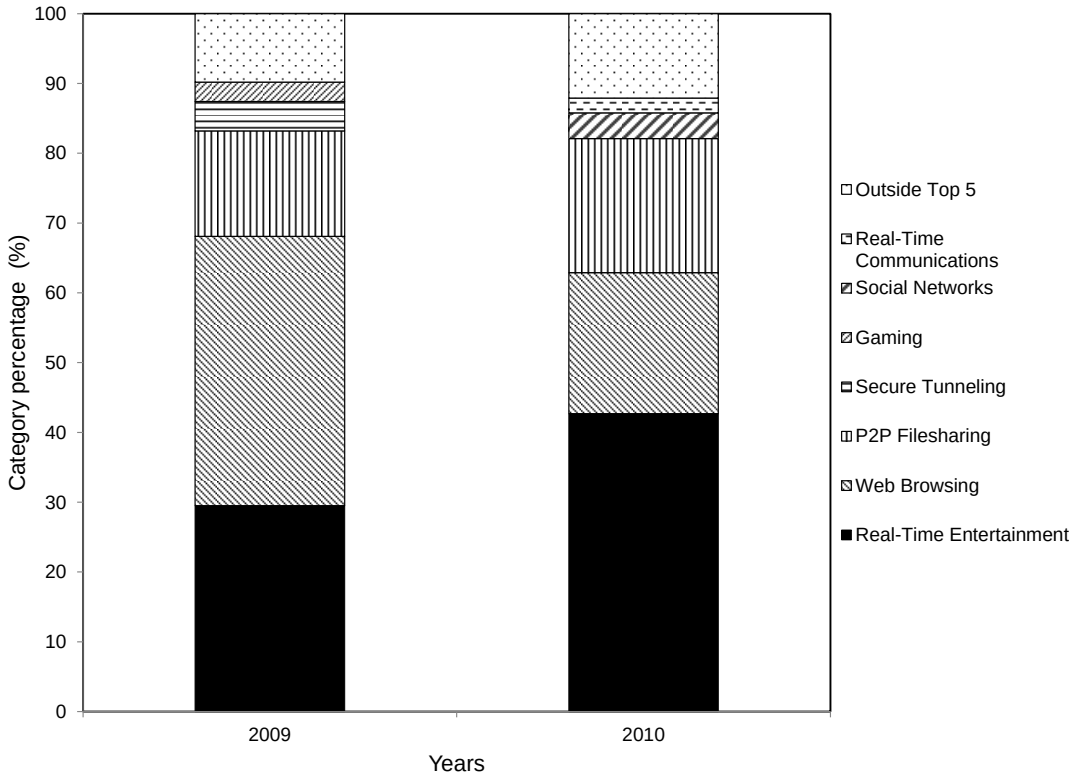


Figure 1.3: Percentage of traffic usage between 2009 and 2010 in North America. Adapted from [17].

Also in this study, it is reported that for mobile access from January to September 2010, real-time entertainment traffic rose substantially: from about 26% to 41.3%, while P2P filesharing traffic suffered a major decline, representing now only 5,6% of peak period bytes. Web browsing remained almost the same with only a slightly drop of about 5%. Social Networking is growing progressively generating 8,3% of mobile traffic during the peak period. In figure 1.4 it is presented a traffic comparison on mobile networks between 2009 and 2010. Figure 1.5 describes the changes in the traffic of applications and services that are driving data usage in the European region. The data gathered in the study presented in [17] is completely subscriber-anonymous. No information regarding specific content or subscriber identity was collected. Without the possibility of detecting P2P traffic, the network administrator cannot take measures against the degradation of the network performance with possible implications for critical applications or services that depend on the network. Recently, P2P applications provide means to encrypt the traffic allowing it to stay undetected until it reaches the end user.

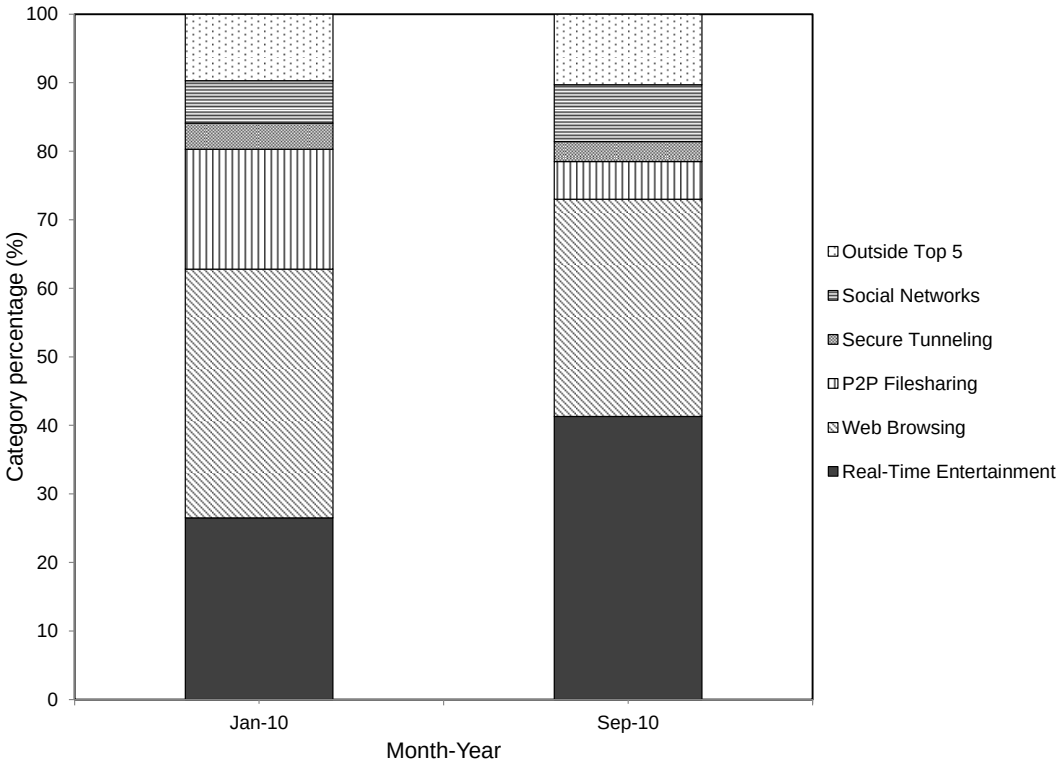


Figure 1.4: Percentage of traffic usage between January and September 2010 in North America using a mobile access. Adapted from [17].

## 1.2 Problem Statement and Objectives

Nowadays, one concern for network administrators is the unauthorized use of P2P media streaming applications that may obfuscate their tracks, through the use of encryption methods or enable tunnelling and proxy support, and may flood the network with huge volumes of multimedia data. The use of obfuscation methods make difficult an accurate identification of the encrypted P2P traffic in order to block it if needed. This dissertation investigates a possible solution to the problem described above, i.e, to accurately identify encrypted traffic generated by peer-to-peer media streaming applications.

The main objective of this dissertation is to propose and validate a solution to the problem of detecting encrypted traffic generated by peer live streaming applications, using techniques of deep packet inspection (DPI), with or without partition schemes of the contents of the packages. The solution makes use of a popular open-source network intrusion detection system (NIDS), Snort [19], and will comprehend the definition of a set of suitable rules for Snort in order to allow the identification of encrypted traffic generated P2P media streaming applications. The research work reported in this dissertation considers the following applications: Livestation [20], TVUPlayer [14] and Goalbit [15], albeit Livestation has discontinued the use of P2P approaches. In order to achieve this main objective, several steps were pursued:

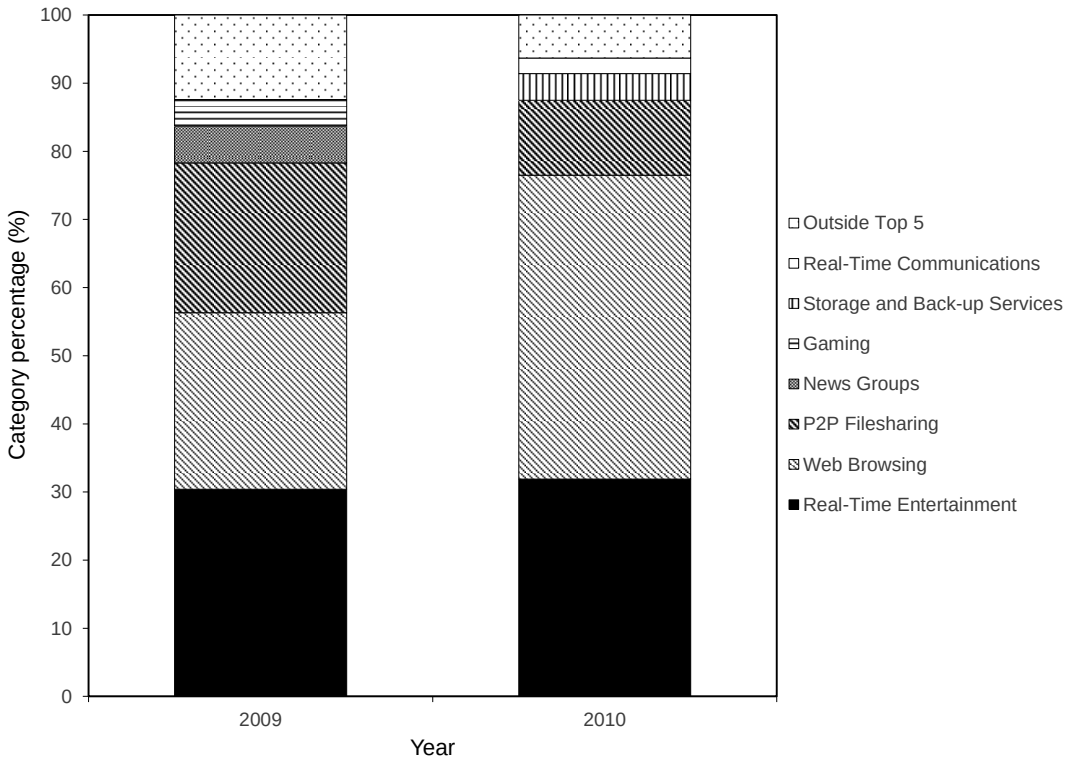


Figure 1.5: Percentage of traffic usage between 2009 and 2010 in Europe. Adapted from [17].

- Study of P2P live streaming concepts;
- Study of the state of art about the detection of encrypted traffic generated by P2P live streaming applications;
- Investigation of a method for detection of encrypted P2P traffic based on deep packet inspection and using an intrusion detection system;
- Implementation of a laboratory testbed for detection of encrypted traffic generated by P2P live streaming applications;
- Definition and implementation of the set of rules for Snort to detect P2P encrypted traffic;
- Performance studies of the proposed solution for detecting P2P encrypted traffic.

## 1.3 Adopted Approach for Solving the Problem

The approach began by reviewing the state of art on P2P traffic detection to check if current solutions also work for detecting media streaming with encrypted traffic. After analysing current solutions and their limitations, a laboratory belonging to the Department of Computer Science of the University of Beira Interior was chosen to conduct the research on the detection of encrypted P2P traffic generated by the three above mentioned media streaming applications. The laboratory consists of six racks, each comprising four computers, making a total of 24 computers, each one connected to the switch in rack by Unshielded Twisted Pair (UTP) Cat.5e/6 cabling. The switches in the six racks are then connected to the auxiliary gateway also by UTP Cat.5e cabling. The auxiliary gateway connects to the main gateway using the same type of cabling, which then connects to a layer-two Enterasys C2H128-48 switch also by UTP Cat.5e/6 cabling, which in turn connects to the network backbone device (an Enterasys E7 switch) in the Department of Computer Science building using an optical fibre uplink. All external communications with the University are managed by the Enterasys SSR main router located in the University Computing Center. Details about the topology of the research laboratory, the characteristics of the hardware such as CPU and RAM and the software that was running on the above mentioned computers can be found in [21]. After the installation of all needed applications, one started the process of traffic capture. The captured traffic was after used to look for patterns on the packets belonging to the three applications so SNORT signatures for detection of the traffic could be created.

Snort is a open-source Network Intrusion Detection System (NIDS) created by Martin Roesch in 1998. It is capable of real-time traffic analysis and packet logging within a Internet Protocol (IP) network. After its initial release as a lightweight NIDS, Snort became a mature feature-rich Intrusion Prevention System (IPS), making it a *de facto* standard on intrusion detection and prevention. It performs protocol analysis, content searching, and content matching. Can also be used to detect probes or attacks, including, but not limited to, common gateway interface, buffer overflows, server message block probes, and stealth port scans. Snort can be configured into three modes:

- *Sniffer mode*, in which basically it reads packets from the network and display them in the console (screen);
- *Packet Logger mode*, instead of displaying the packets in the screen, it saves them into a file in the disk;
- *Network Intrusion Detection System mode*, this is the most complex mode, in which after an analysis of the network traffic against a user rule set, it decides what action is performed.

Only the Network Intrusion Detection System mode was used along this dissertation, because the other two modes were almost of no interest for this dissertation, since they do not process packets against a user rule set. NIDS mode has multiple output options, but once again only one was used, and that was fast alert mode output. Fast alert mode option writes the alert in a simple format with a timestamp, the specific alert message, and the source and destination IPs/ports. The following command is an example, but similar to the one we used on every performed test:

```
./snort -c snort.conf -A fast -h 192.168.1.0/24
```

The command used on the test had another particularity, since we aimed high-speed network operation (100/1000 Mbps), we needed to have another program for writing the alerts into a database. Snort has the option to log an alert in a binary form, increasing this way his own speed, and leaving the writing to another useful program called Barnyard2 [22]. Barnyard2 is an open source interpreter for Snort unified2 binary output files. Its primary use is for allowing Snort to write to disk in an efficient manner and leaving the task of parsing binary data into various formats to a separate process that will not cause Snort to miss network traffic.

Before proceeding to Snort installation and configuration, we need to install the new Data Acquisition (DAQ) library that was released with Snort version 2.9. The DAQ replaces direct calls to pcap functions with an abstraction layer that facilitates operation on a variety of hardware and software interfaces without requiring changes to Snort. It is possible to select the DAQ type and mode when invoking Snort to perform pcap readback or inline operation. The version used was the `daq-0.5.tar.gz` that can be downloaded in [19]. We extracted the source code and then compile it using the common methods:

- `./configure`
- `make`
- `make install`
- `ldconfig`

Remember that for using the above commands, root privileges are necessary, so it was used `sudo` before the commands. The last command is used, when we need to create the necessary links and cache to the most recent shared libraries found in the directories specified on the command line, in the file /etc/ld.so.conf, and in the trusted directories (/lib and /usr/lib).

After installing DAQ, we also need to install another library called `libdnet`. The same process of the DAQ is applied here too, without the `ldconfig` command. Extract the source code, compile it and after that create a symbolic link between files in different directories with `ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libdnet.1`.

To install Snort, we could get it from the repositories, but that would not guarantee that Snort was on its latest version. Snort 2.9.05 was downloaded from [19]. Snort was extracted as normal for compilation, that in this case have some options that were needed:

- `./configure -enable-ipv6 -enable-gre -enable-mpls -enable-targetbased -enable-decoder-preprocessor-rules -enable-ppm -enable-perfprofiling -enable-zlib -enable-active-response -enable-normalizer -enablereload -enable-react -enable-flexresp3`
- `make`
- `make install`

The options of `./configure` are described further down on snort.conf file. For running Snort it is also needed to set up an environment, that consists on creating some directories, create a Snort user, and move some files. The file snort.conf contains the main configuration for running

Snort, and is sectioned in:

1. Set the network variables for your network
2. Configure the decoder
3. Configure the base detection engine
4. Configure dynamic loaded libraries
5. Configure preprocessors
6. Configure output plugins
7. Customize the rule set
8. Customize preprocessor and decoder rule set
9. Customize shared object rule set

In section one of snort.conf file, the network variable HOME_NET was set to 10.0.2.0/24 and the variable EXTERNAL_NET was set to !$HOME_NET, i.e., all that was not in 10.0.2.0/24 was considered external. In section four, the paths for the libraries and the preprocessor engine were changed to their real locations. To use Barnyard, we needed to change the output plugin to `output unified2: filename snort.log, limit 128` in section six.

We used MySQL as the relational database management system for Snort application. No extraordinary configuration is needed, just need to create a database based on the scheme file found on Snort directory. As part of Snort installation, the latest rules had to be downloaded. All rules created for this thesis were moved to the Snort rules directory and added to snort.conf in section seven. Rules in Snort have two parts: the rule header, and the rule options. The rule header contains the rules action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information about which parts of the packet should be inspected to determine if the rule action should be taken.

After running experiences using Livestation, TVUPlayer and Goalbit, one carried out an exhaustive work looking for patterns in the payload of encrypted packets generated by these applications. A method was then derived, which is based on the identification of some packets that are not encrypted and the association of these packets with the corresponding flow. The patterns for a given P2P application were coded into Snort rules and using the proposed method all packets of the flow to which the identified packets belong to were identified as belonging to that P2P application. The remainder of the dissertation addresses this solution.

## 1.4 Main Contribution

The main contribution of this dissertation is a method for classifying encrypted traffic generated by Livestation, TVUPlayer, and GoalBit applications. This method includes a set of Snort rules

created after a careful analysis of the encrypted traffic in order to find a pattern in the payload of the packets that is repeated through the flow of packets. Regarding GoalBit, four different stream transmission configurations were used in order to cover all the possible outcomes in terms of repetitive patterns.

This contribution, considering both trackerless and tracker-based configurations of GoalBit and also TVUPlayer and Livestation applications, is planned to be submitted for publication in an international journal. An earlier version of this article only considering GoalBit in a trackerless configuration has been published in the Proceedings of the 10th IEEE International Symposium on Network Computing and Applications (IEEE NCA11), held in Cambridge, Massachusetts USA, between 25 and 27 of August, 2011.

## 1.5 Dissertation Overview

This dissertation is presented as a compilation of articles, and for that is organized has follows: in Chapter 1, it is presented an overview of the P2P technology. It is described the problems associated with high traffic volumes and the need to classify network traffic. It is also stated the problem for which this dissertation provides a possible solution, as well as the main contributions of this work to the advance of the state of art in P2P traffic detection. In the last section of the first chapter, it is described the structure of this dissertation.

In Chapter 2, it is presented the first article that was presented at the 10th IEEE International Symposium on Network Computing and Applications (IEEE NCA11), which describes the research work related to the detection of GoalBit encrypted traffic when using the trackerless-based stream transmission feature.

Chapter 3 presents an article to be submitted for publication in an international journal, that is an extended version of the one presented on Chapter 2, in which we also take into account the cases of TVUPlayer and Livestation, but also experiment with four different transmission configurations in GoalBit, with and without encryption using a tracker-based approach, and with and without encryption using a trackerless-based approach.

Finally in Chapter 4, main conclusions are presented, as well as directions for future work.

## References


[1] "Napster Homepage." http://www.napster.com/. (Accessed October 13, 2011).

[2] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Commun. ACM*, vol. 53, pp. 72–82, October 2010.

[3] S. Ortiz Jr., "Is peer-to-peer on the decline?," *Computer*, vol. 44, pp. 11–13, February 2011.

[4] "Megaupload." http://www.megaupload.com/. (Accessed October 13, 2011).

[5] "Rapidshare." https://www.rapidshare.com/. (Accessed October 13, 2011).

[6] "Netflix." https://signup.netflix.com/global. (Accessed October 13, 2011).

[7] N. Leavitt, "Network-usage changes push internet traffic to the edge," *Computer*, vol. 43, pp. 13–15, October 2010.

[8] Y.-C. Tu, J. Sun, M. Hefeeda, and S. Prabhakar, "An analytical study of peer-to-peer media streaming systems," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, pp. 354–376, Nov. 2005.

[9] M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," *Proceedings 22nd International Conference on Distributed Computing Systems*, pp. 363–371, 2002.

[10] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will IPTV ride the peer-to-peer stream? [Peer-to-Peer Multimedia Streaming]," *IEEE Communications Magazine*, vol. 45, pp. 86–92, June 2007.

[11] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications*, vol. 1, pp. 18–28, Jan. 2008.

[12] Q. Zhu, R. Wang, D. Qian, and F. Xiao, "Re-exploring the potential of using tree structure in p2p live streaming networks," in *Proceedings of the 2009 Sixth IFIP International Conference on Network and Parallel Computing*, NPC '09, (Washington, DC, USA), pp. 125–132, IEEE Computer Society, 2009.

[13] "Bittorrent." http://www.bittorrent.com/. (Accessed October 13, 2011).

[14] "TVUNetworks Homepage." http://www.tvunetworks.com/. (Accessed October 13, 2011).

[15] "GoalBit Homepage." http://goalbit.sourceforge.net/. (Accessed October 13, 2011).

[16] N. Zong, Y. Zhang, V. Pascual, C. Williams, and L. Xiao, "P2P Streaming Protocol (PPSP) Requirements. (2010)." http://tools.ietf.org/html/draft-zong-ppsp-reqs-04, 2010. (Internet Draft. Accessed April 10, 2011.).

[17] "Sandvine 2010 Global Internet Phenomena Report." http://www.sandvine.com/downloads/documents/2010GlobalInternetPhenomenaReport.pdf. (Accessed October 13, 2011).

[18] "Sandvine homepage." http://www.sandvine.com/. (Accessed October 13, 2011).

[19] "SNORT Homepage." http://www.snort.org. (Accessed October 13, 2011).

[20] "Livestation Homepage." (accessed October 13, 2011).

[21] A. F. Esteves, P. R. M. Inácio, M. Pereira, and M. M. Freire, "On-line Detection of Encrypted Traffic Generated by Mesh-Based Peer-to-Peer Live Streaming Applications: The Case of GoalBit," in *Proceedings of 10th IEEE International Symposium Network Computing and Applications (NCA)*, pp. 223–228, Aug. 2011.

[22] "Barnyard2 Project Page." http://www.securixlive.com/barnyard2/. (accessed October 13, 2011).

# Chapter 2

## On-Line Detection of Encrypted Traffic Generated by Mesh-Based Peer-to-Peer Live Streaming Applications: The Case of GoalBit

André F. Esteves, Pedro R. M. Inácio, Manuela Pereira, and Mário M. Freire

*Proceedings of 2011 Tenth IEEE International Symposium on Network Computing and Applications (NCA 2011)*, Cambridge, MA, USA, August 25-27 2011

IEEE Computer Society Press, Los Alamitos, CA, ISBN:978-0-7695-4489-2, pp. 223-228.

# On-Line Detection of Encrypted Traffic Generated by Mesh-Based Peer-to-Peer Live Streaming Applications: The Case of GoalBit

André F. Esteves*, Pedro R. M. Inácio†, Manuela Pereira‡, and Mário M. Freire§
Instituto de Telecomunicações, Department of Computer Science, University of Beira Interior
Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
Email: *m3412@ubi.pt,{†inacio,‡mpereira,§mario}@di.ubi.pt

***Abstract*—The number and popularity of applications developed over the Peer-to-Peer (P2P) network paradigm has been growing over the last decade, some of which are dedicated to streaming multimedia content. To deceive traffic shaping mechanisms or improve the security of the communications, these applications generate encrypted traffic or resort to several obfuscation techniques, making it difficult to manage this kind of traffic at the network level. In this work, we propose a method that explores transmission vulnerabilities of the encrypted traffic allowing its detection. Hence, an experimental testbed was created to capture a diversity of traffic, which includes flows of a widely used P2P media streaming application called GoalBit. The collected traces of traffic were then analysed, and a set of rules was created for the SNORT network intrusion detection system, which allows the successful detection of the encrypted traffic generated by GoalBit. The accuracy of this system was then validated experimentally.**

## I. Introduction

Empowered by web-based video streaming services, such as YouTube, and by the success of IP television (IPTV), video traffic has been grown significantly over the last few years, being foreseen a sevenfold increase between 2009 and 2014 [1]. Another contribution for the rise of the video traffic is the emergence of peer-to-peer (P2P) media streaming services, also known as P2P TV services. In fact, media streaming over the Internet may be implemented either through IP multicast functionalities or through peer-to-peer systems. Since the former approach only works on network infrastructures managed by a single network operator due to limitations of IP multicast facilities, P2P-based approach has been successfully investigated to overcome these limitations, having the potential to provide a scalable worldwide infrastructure [2].

The popularity and diversity of applications using the peer-to-peer (P2P) networking paradigm has substantially grown over the last decade. By exploring the resiliency and worldwide adoption of this technology, these applications are not only used for content sharing and distribution, but also for media streaming. To protect the privacy of the users, bypass filtering/blocking mechanisms or overcome network limitations, it is common for these applications to employ evasion techniques, provide encryption for the contents and include the option to apply protocol obfuscation. To avoid being classified with basis on port numbers, these applications offer the possibility to adjust the port number as a configuration parameter, or simply agree on the usage of a random one during the P2P link establishment phase. On the one hand, these functionalities favour the adoption of the applications, since they adapt to different network conditions and sometimes operate in very controlled scenarios but, on the other, they make the task of classifying and managing the network traffic a lot harder. This problem has been motivating several research works on this area, as further elaborated below.

Methods for classification of peer-to-peer traffic can nowadays be classified into three categories [3] : i) traffic classification based on port numbers, ii) traffic classification based on deep packet inspection (DPI), and iii) traffic classification in the dark.

Methods for P2P application detection based on port numbers are ineffective because most of the currently available P2P applications use either arbitrary ports or well-known ports that are assigned to specific application-layer protocols such as Hypertext Protocol (HTTP) to deceive classification mechanisms. Due to this reason, this kind of methods is nowadays considered obsolete.

The category of deep packet inspection methods search on the entire data field of IP packets, often referred to as IP packet payload, for specific character patterns (also called signatures) associated with specific application-layer protocols. Usually those signatures are previously stored in a database. This kind of methods is used today not only for detection of application protocols, but also for detection of virus and malware through content filtering and also for intrusion detection and protection. Key issues in this kind of methods are the performance under high-speed network operation and the detection of encrypted traffic. A suitable overview of recently investigated methods of this category is provided in [3]. Some also claim that DPI may lead to legal and privacy issues [4]–[6], since DPI implies that the network manager is somehow accessing the contents of the communications. Nonetheless, the DPI techniques have a high accuracy rate for known protocols or when the traffic is not encrypted. Nonetheless, in [7], it is stated that it is possible to detect and classify P2P applications that use encrypted traffic without losing much accuracy.

The category of traffic classification in the dark relies on the

classification of the IP packets using behavioural or statistical patterns at a flow-level or through specific properties of the packets (excluding the data field of the packets), such as source/destination addresses or packet sizes. Encryption of the data field of the packets is not a problem for this kind of methods because they do not analyse the data field contents, but usually they suffer from lack of accuracy. This category of methods comprehends a plethora of strategies, including statistical approaches, e. g. [8], behavioural approaches, which may be based on specific flavours such as social interaction, e.g. [9], entropy, e.g. [10], recurrence quantification analysis [11], and a variety machine learning (ML) approaches for traffic classification, which may be further categorized as unsupervised [12], [13], supervised [14]–[16] and semi-supervised ML approaches [17].

In this paper, we address the problem of detection of encrypted traffic generated by one of the most popular mesh-based P2P live streaming applications that generate encrypted traffic in a freely available version: the GoalBit [18] application. Along the paper we describe a solution for this problem based on deep packet inspection and present its assessment in terms of accuracy.

The remainder of this paper is organised as follows. Section II reviews several approaches to classify encrypted P2P traffic. Section III describes the method and the experimental test-bed for detection of encrypted traffic generated by BoalBit and Section IV presents and discusses the obtained results. Section V contains the main conclusions.

## II. Related Work

Albeit a large amount of papers have been published regarding Internet traffic classification, only a very few of them address the problem of identification of traffic generated by peer-to-peer media streaming applications. In the following, it is presented an overview about papers addressing the specific issue of the identification of traffic generated by P2P media streaming applications.

In [19], a comparison among three different traffic classifiers for peer-to-peer television (P2P TV) applications is presented: a DPI method, a method based on single-class Support Vector Machines (SVMs), known as IPSVM classifier [20], and a method based on multi-class SVMs, known as KISS classifier [21], which needs to perform payload analysis, being therefore ineffective for encrypted traffic. The considered applications were used without encryption or obfuscation techniques. It is shown that the DPI approach is very effective mainly due to the fact that those applications do not use encryption or obfuscation techniques, being appointed the main disadvantage of this approach, which relies on the process of deriving the signatures that is still manual, time-consuming and error-prone. It is also shown that other two methods relying on single-class or multi-class SVMs replace the signature database with a suitable automatic training phase and they guarantee very good results provided that the training set is appropriate.

In [22], Xu et al. proposed a content-based partitioning scheme to discover shared data and identify the P2P data transfer behavior, being applied to some specific applications of P2P file sharing, P2P live streaming and P2P video on demand. It is shown through experiments that the proposed scheme has a high accuracy. However, the proposed method needs to perform payload analysis being therefore ineffective for classification of encrypted traffic.

In [23], Bermolen et al. proposed a behavioral approach based on support vector machines for fine-grained classification of P2P media streaming applications. It is shown that the proposed method, known as Abacus, correctly classifies about 95% of packets, bytes and peers in the worst case. It is also shown that the number of false alarms is very small, being below 0.1% in the worst case.

In [24], Finamore et al. presented a comparison between Abacus [23] and Kiss [21]. It is shown that both approaches are comparable in terms of accuracy in classifying P2P media streaming applications, at least regarding the percentage of correctly classified bytes and for non-encrypted traffic. However, in terms of the computational cost of the classifiers, Abacus presents a better performance than KISS, but KISS is much more general, as it can classify other kinds of applications as well. However, KISS is ineffective for encrypted traffic whereas Abacus does not suffer from this drawback.

As reported in some of the papers mentioned above, most of the research work about detection of encrypted traffic has abandoned approaches based on deep packet inspection for detection of encrypted traffic with the argument that these techniques are not applicable to that kind of traffic. In this paper, we explore the use of deep packet inspection for detection of encrypted traffic by exploring transmission vulnerabilities of encrypted traffic, since in some critical points of the link control or establishement phases, or even during some parts of data transmission, encryption is not used.

## III. Method and Experimental Setup for Detection of Encrypted Goalbit Traffic

The experimental test-bed was implemented in the MSP-Cv research lab at the University of Beira Interior. The lab contains 24 desktop computers running Fedora 9 or Windows Server 2003 operating systems. There is also a computer functioning as an auxiliary gateway between these desktop computers and the main Internet gateway. This auxiliary gateway manages all incoming and outgoing traffic in that lab, runs the Fedora 9 operating system and has, among other, the `tcpdump` [25], SNORT, `OpenDPI` [26], `l7-netpdlclassifier` [27] and `l7-filter` [28] applications installed. `Tcpdump` was used for capturing traffic and for analysing it. After analysing the traffic to find patterns, several rules were created using the SNORT syntax.

The other classification applications mentioned above were used to provide a base of comparison in order to validate the set of rules created for SNORT, starting with `OpenDPI` which is an open-source version of Protocol and Application Classification Engine (PACE), a traffic classification engine

TABLE I
CHARACTERISTICS OF HARDWARE AND SOFTWARE USED IN THE TEST-BED ON THE RESEARCH LAB.

| Type | Operating System | CPU | RAM | Software |
|---|---|---|---|---|
| Desktop | Windows Server 2003 | Pentium D 2.66 Ghz | 1 GB | GoalBit, Skype, Youtube, eMule, TVUPlayer, UStream, etc. |
| Laptop | Windows 7 | Core 2 Duo 2.4 Ghz | 3 GB | GoalBit, Windows Live Messenger, HTTP and audio streaming |
| Laptop | Ubuntu 10.10 | Core Duo 1.8 Ghz | 1 GB | Skype |
| Desktop | Fedora 9 | Pentium D 2.66 Ghz | 1 GB | Skype |
| Gateway | Fedora 9 | Xeon 2.26 Ghz | 6 GB | tcpdumd, SNORT, l7-netpdlclassifier, l7-filter, OpenDPI |

from ipoque. `OpenDPI` uses mainly DPI to classify network traffic and has the particularity of not supporting the detection of encrypted protocols. `L7-netpdlclassifier` was developed by the Computer Networks Group (NetGroup) of *Politecnico di Torino* and makes use of the NetBee [29] library. The signatures are written using the Netpdl language (a XML-based syntax), the output it produces is a packet-based output file (i.e., the classification result for each packet). `L7-filter` is a well known open-source application that makes use of DPI for network traffic classification, this application uses regular expressions to define the payload signatures and produce a flow-based output (i.e., the classification result for each flow).

The 24 desktop computers were running different applications, namely P2P-TV (TVUPlayer, GoalBit, Sopcast, PPStream), P2P file-sharing applications (eMule, FrostWire), VoIP applications (different versions of Skype), online audio streaming and online flash video streaming (Youtube and UStream).

One of the most important requirements for running P2P software is the hard-drive space. On the other hand, real-time network monitoring requires more computational resources, since processing the captured traffic and running the intrusion detection system at the same time consumes a lot of Central Processor Unit (CPU) power and Random Access Memory (RAM). Table I summarizes the characteristics of the hardware such as CPU and RAM and also the software that was running on the above mentioned computers.

In the lab, each set of four desktops are connected to a single switch using Unshielded Twisted Pair (UTP) Cat.5e cabling, which is then connected to the auxiliary gateway using the same type of cabling, which in turn connects to the network backbone device (an Enterasys E7 switch) using optical fiber. All external communications with the University are managed by the Enterasys SSR main router located in the University Computing Center. Figure 1 shows the topology of the testbed.

GoalBit is an open-source video streaming application that runs on Windows, GNU/Linux, Mac and even Solaris. It was

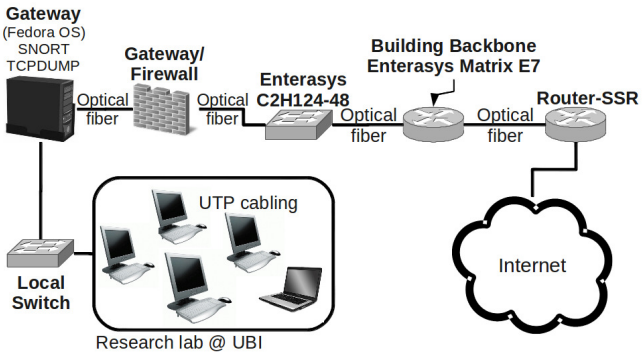


Fig. 1. Experimental testbed at the research lab.

initially developed by Uruguayan researchers for experiments concerning P2P networks. GoalBit enables the user, not only to watch streamed channels, but also broadcast their own multimedia contents. GoalBit supports several video sources such as files, Hypertext Protocol / Microsoft Media Services / Real-Time Protocol (HTTP/MMS/RTP) streams, webcams, DVD devices, etc. It also supports multiple video formats like Windows Media Video (WMV), Divx, H.264 and others.

Users can choose broadcasting between streaming using trackers (BitTorrent-based) or a trackerless mechanism as the one implemented by Kademlia. They can also choose to encrypt the traffic contents, change the bit-rate or even the chunk size. In this work, two video files were used for streaming experiments: one with the duration of 40m and the other with the duration of 1h35m. During the experiments, the options to encrypt the traffic using the private key function available at the broadcast feature was selected. The bit-rate and the chunk size were not altered, being left to default values of 285 and 65536 respectively.

The computers running the application were all logically connected to the same network of the research lab, except for a laptop that was connect to a wireless network with an address space different from the wired one. The computer connected to the wireless network was broadcasting the stream. Because the main broadcasting computer was outside the research network, we were able to capture all communications between the internal and external computers. The set of SNORT rules for GoalBit was created after an exhaustive analysis of the captured traffic in a controlled network environment. We included the set of rules in Table II.

The main idea behind the proposed method relies on the finding of GoalBit signatures (repetitive string series) in the data field of IP packets, in some critical points of the link phase (e.g. establishing phase and further non-encrypted communication during data transfer phase), when encryption is not used, which allows the identification of the sessions associated with the GoalBit application. Then all traffic associated with that flows is identified as being generated by GoalBit, albeit we are only able to detect the parts of which that are not encrypted. Those signatures, to be identified in IP packets carrying data from GoalBit application, are expressed in terms

TABLE II
SET OF SNORT RULES FOR DETECTION OF GOALBIT IP PACKETS.

| Rule number & Type of traffic | Rule definition in SNORT syntax |
|---|---|
| Rule 1000506/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Tracker Cookie"; content:"\|47 67 61 6c 42 69 74 20 70 72 6f 74 6f 74 6f 63 6f 6c\|"; dsize:77; threshold:type both, track by_src,count 1,seconds 10; sid:1000506; rev:1;` |
| Rule 1000508/ UDP traffic | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 c2 01 e3\|";content:"\|00 c2 01 e3\|"; dsize:474; threshold:type both, track by_src,count 3,seconds 10; sid:1000508; rev:1;)` |
| Rule 1000509/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 0d 06 00 00\|"; flow: established; content:"\|00 0d 06 00 00\|";threshold:type both, track by_src,count 3,seconds 10; sid:1000509; rev:2;)` |
| Rule 1000510/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 05 0a 00 00\|";content:"\|00 05 0a 00 00\|"; dsize:9; threshold:type both, track by_src,count 3,seconds 10; sid:1000510; rev:1;)` |
| Rule 1000560/ UDP traffic | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|56 9c dd ed\|";content:"\|56 9c dd ed\|"; dsize:24; threshold:type both, track by_src,count 3,seconds 10; sid:1000560; rev:1;)` |
| Rule 1000561/ UDP traffic | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|e3 0c 02\|";content:"\|e3 0c 02\|"; dsize:25; threshold:type both, track by_src,count 3,seconds 10; sid:1000561; rev:1;)` |
| Rule 1000562/ UDP traffic | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|e3 0b\|";content:"\|e3 0b\|"; dsize:33; threshold:type both, track by_src,count 3,seconds 10; sid:1000562; rev:1;)` |
| Rule 1000564/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP \|00 00 00 00 01 00 00\|";content:"\|00 00 00 00 01 00 00\|"; offset:10; depth:17; dsize:17; threshold:type both, track by_src,count 3,seconds 10; sid:1000564; rev:1;)` |
| Rule 1000565/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP Payload Size (1460bytes)";content:"\|62 72 6f 61 64 63 61 73 74 65 72 7b 70 69 65\|"; depth:90; dsize:1460; threshold:type both, track by_src,count 3,seconds 10; sid:1000565; rev:1;)` |
| Rule 1000566/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP Payload Size (1456bytes)";content:"\|67 6f 61 6c 62 69 74 5f 74 72 61 63\|"; dsize:1460; threshold:type both, track by_src,count 3,seconds 10; sid:1000566; rev:1;)` |
| Rule 1000567/ TCP traffic | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP \|00 09\|";content:"\|00 09\|"; dsize:13; threshold:type both, track by_src,count 3,seconds 10; sid:1000567; rev:1;)` |

of Snort rules, being then Snort used to detect such signatures.

## IV. RESULTS AND DISCUSSION

The results discussed in this section represent the work performed for the traffic traces captured during three different days as shown in Table III. Based on the signatures discovered using human observation, it was possible to identify packets carrying GoalBit encrypted traffic and, from there, classify the flows generated by the application. As expected (we are dealing with encrypted traffic), the true positive rate for the classification of the traffic at the packet level was below 25%. Nonetheless, a small number of packets was sufficient to identify most of the flows generated by the GoalBit application. In order to reduce the disk space usage (another computational burdens), the rule set was created using a threshold count of 3 at each 10 seconds, i.e., at each 10 seconds, SNORT only emits a maximum number of 3 alerts.

The TCP rules were created based on periods in which encryption was not being used, like the start of the transmission and also some parts in the middle of the transmission. Rule `1000509` was observed on GoalBit traffic but when tested against the traces we captured also matched non-GoalBit traffic, accounting this way for the false positive rate. Rule `1000506` was being triggered both when encryption was used but also when it was not, the main issue in using this rule is that it only was triggered when we have the first bytes of the TCP session, but it is still quite useful to detect GoalBit traffic. Two rules were created after observing packets with large payload sizes, they are rule `1000565` with 1460 bytes and rule `1000566` with 1456 bytes respectively, the packets with this payload signatures can be found scattered all over the GoalBit traffic, however there are few packets triggered by these two rules, meaning that these packets were not using encryption.

The UDP rules were created also by observing GoalBit traffic, but in this case the signatures observed were from the Kademlia protocol used by GoalBit in two of the traces, for

TABLE III
CHARACTERISTICS OF CAPTURED TRAFFIC

| Date | Start time | End Time | Volume(GB) | Number of Packets |
|---|---|---|---|---|
| 17-01-2011 | 17h46 | 19h31 | 3.4 | 6570257 |
| 21-01-2011 | 18h12 | 21h04 | 5.9 | 9634160 |
| 04-03-2011 | 12h17 | 19h31 | 29.7 | 58995815 |

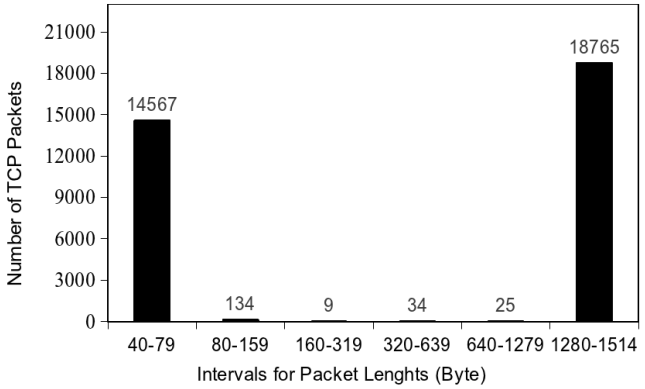


Fig. 2. Number of TCP packets in the collected traces vs. the packet size.

this reason all applications that make use of this protocol are also triggered as GoalBit traffic. The UDP rules do not appear in the charts, because they were triggering all applications that use Kademlia protocol. Nonetheless the UDP packets were carrying information not relative to the content being distributed, but the information carried in the UDP packets is useful for peers communicating among themselves and also for making them known in the network.

Due to the low true positive rate a flow classification approach was used. In the flow classification we could identify multiple packets using the rule set and because the flow actually belonged to GoalBit we classified all the packets on that flow as GoalBit. Using the flow approach we managed to increased the true positive rate by 77%, making a total of 96% true positive rate.

As we were running the rule set, some false positives were raised, due to a signature that identifies GoalBit as well as other traffic, accounting for a total of 1.3%. We have a small percentage of false negatives 2.7% due to multiple flows only carrying ACK,SYN and PUSH flag packets. In figure 2, we can see that in GoalBit the packet sizes do not vary that much since we only count a significant number of packets with size 40 to 79 bytes and from 1280 to 1514 bytes.

To obtain a comparative perspective of the performance of the SNORT system with our rule set, we ran it against other DPI software like OpenDPI, l7-filter and l7-netpdlclassifier. We can see in figure 3 that the applications tested failed to classify GoalBit, the detection rate referred in figure 3 is the total accuracy percentage of the set of rules created without false positive or false negatives.

We conducted several experiments by dividing the trace files into ten pcap files. The results showed a great amount

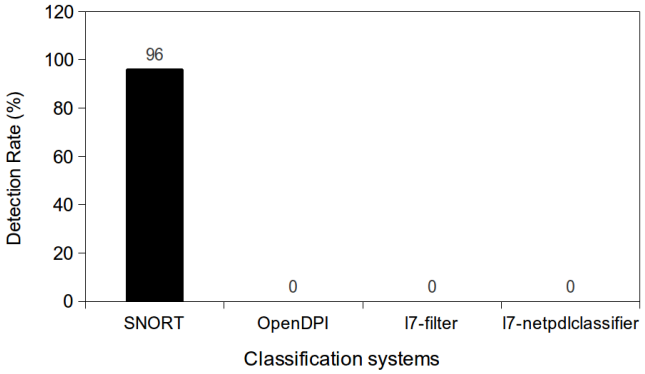


Fig. 3. Comparison of the detection rate of SNORT with the rules proposed herein, OpenDPI, l7-filter and l7-netpdlclassifier for the captured traces.

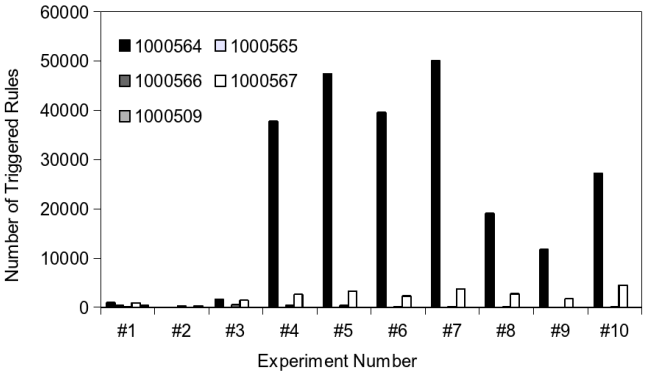


Fig. 4. Number of SNORT rules triggered during 10 different experiments.

of triggered TCP rules from experiment four to the last one, as shown in figure 4.

As in our testbed the true positive rate was a little low as said before, we started by classifying the traffic using a flow approach which will be explained below. We grabbed the ten experiment files and we filtered out the IPs from the machines that were just running GoalBit, then protocols like ARP, NBNS, ICMP, HTTP, MSN and ICMPv6 were also removed because they were of no use to create the set of rules. From there, we then filtered a random TCP stream using the Wireshark "Follow TCP stream" feature that only shows the chosen TCP stream. After the previous step we had a trace file containing only the chosen TCP stream, we ran SNORT to see if it triggered any rules, and as hoped it triggered several rules has shown in figure 5.

## V. CONCLUSION

P2P applications became very popular, not only for content sharing and distribution but also for media streaming. In the last few years an attempt to obscure or even encrypt the traffic generated by those P2P applications has been made in order to increase privacy or to avoid the identification of traffic generated by such applications to escape traffic shaping or blocking. In this paper, we investigated the detection of encrypted traffic generated by GoalBit, one of the most popular

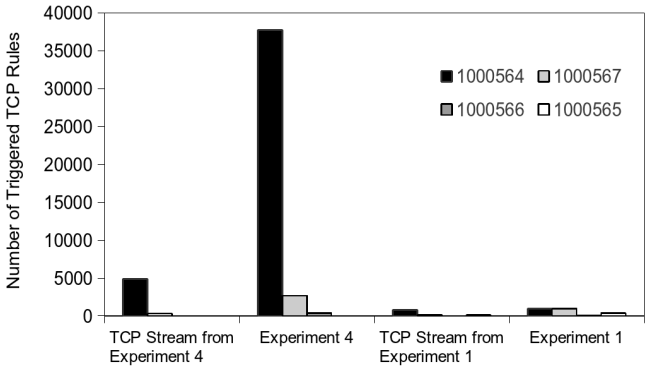


Fig. 5. Number of TCP SNORT rules triggered for two GoalBit TCP flows extracted from the traffic experiments.

P2P media streaming applications. We proposed a method based on deep packet inspection designed to operate on-line with the aim of detecting encrypted traffic generated by GoalBit. The proposed method looks for GoalBit signatures in packets, which are coded as SNORT rules, this system being used to detect encrypted GoalBit traffic. The proposed method is evaluated experimentally through a test-bed created for that purpose, being shown through experiments that the accuracy of the proposed method is about 96%. Results also show that the signatures devised along the research work are sufficient to distintingly identify a few packets generated by GoalBit, which are enought to point out most of the flows of traffic generated by this application. The design of several other signatures for similar P2P applications is part of our future research directions.

## ACKNOWLEDGMENT

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia through the project TRAMANET: Traffic and Trust Management in Peer-to-Peer Networks, with contracts PTDC/EIA/73072/2006 and FCOMP-01-0124-FEDER-007253.

## REFERENCES

[1] N. Leavitt, "Network-Usage Changes Push Internet Traffic to the Edge," *Computer*, vol. 43, no. 10, pp. 13 –15, 2010.

[2] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi, "Building a cooperative P2P-TV application over a wise network: the approach of the European FP-7 strep NAPA-WINE," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 20 –22, 2008.

[3] Joao V. Gomes, Pedro R. M. Inácio, Manuela Pereira, Mário M. Freire, and Paulo P. Monteiro, "Detection and Classification of Peer-to-Peer Traffic: A Survey," (Accessed April 10, 2011). [Online]. Available: http://floyd.di.ubi.pt/nmcg/pdf/2011-P2PSurvey.pdf

[4] Z. Guo and Z. Qiu, "Identification peer-to-peer traffic for high speed networks using packet sampling and application signatures," in *Proceedings of 9th International Conference on Signal Processing, 2008 (ICSP 2008)*, 2008, pp. 2013 –2019.

[5] S. Jordan, "Some Traffic Management Practices Are Unreasonable," in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009 (ICCCN 2009)*, 2009, pp. 1–6.

[6] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "PortLoad: Taking the Best of Two Worlds in Traffic Classification," in *Proceedings of INFOCOM IEEE Conference on Computer Communications Workshops*, 2010, pp. 1–5.

[7] European Advanced Networking Test Center, "P2P Taste Test," January 2009. [Online]. Available: http://www.internetevolution.com/document.asp?doc_id=178633(accessedApril10,2011)

[8] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli, "A statistical approach to IP-level classification of network traffic," in *Proceedings of IEEE International Conference on Communications, 2006 (ICC'06)*, vol. 1, 2006, pp. 170–176.

[9] M. Iliofotou, H. chul Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, "Graph-Based P2P Traffic Classification at the Internet Backbone," in *Proceedings of IEEE INFOCOM Workshops*, 2009, pp. 1 –6.

[10] J. Gomes, P. Inacio, M. Freire, M. Pereira, and P. Monteiro, "Analysis of Peer-to-Peer Traffic Using a Behavioural Method Based on Entropy," in *Proceedings of IEEE International Performance, Computing and Communications Conference, 2008 (IPCCC 2008)*, 2008, pp. 201–208.

[11] F. Palmieri and U. Fiore, "A nonlinear, recurrence-based approach to traffic classification," *Comput. Netw.*, vol. 53, pp. 761–773, April 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1517860.1518207

[12] S. Zander, T. Nguyen, G.Armitage, "Automated traffic classification and application identification using machine learning," in *Proceedings of 30th IEEE Conference on Local Computer Networks*, 2005, pp. 250–257.

[13] Erman J, Arlitt M, Mahanti A., "Traffic classification using clustering algorithms," in *Proceedings of SIGCOMM06 Workshops*, Pisa, Italy, 2006, pp. 281–286.

[14] A. Este, F. Gringoli, L. Salgarelli, "Support Vector Machines for TCP traffic classification," *Computer Networks*, vol. 53, pp. 2476–2490, 2009.

[15] Auld T, Moore AW, Gull SF., "Bayesian neural networks for Internet traffic classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, 2007.

[16] Moore AW, Zuev D., "Internet traffic classification using Bayesian analysis techniques," in *Proceedings of ACM Sigmetrics*, 2005, pp. 50–60.

[17] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, "Offline/realtime traffic classification using semisupervised learning," *Performance Evaluation*, vol. 64, pp. 1194–1213, 2007.

[18] "GoalBit Homepage." [Online]. Available: http://goalbit.sourceforge.net/(accessedApril10,2011)

[19] N. Cascarano, F. Risso, A. Este, F. Gringoli, A. Finamore, and M. Mellia, "Comparing P2PTV traffic classifiers," in *Proceedings of the IEEE International Conference on Communications (ICC 2010)*. Cape Town, South Africa: IEEE, May 2010, p. 6 pages.

[20] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Elsevier Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[21] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "KISS: Stochastic packet inspection," in *Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA '09)*, vol. 5537. Aachen, Germany: Springer-Verlag, Berlin, Heidelberg, May 2009, pp. 117–125.

[22] K. Xu, M. Zhang, M. Ye, D. M. Chiu, and J. Wu, "Identify P2P traffic by inspecting data transfer behavior," *Computer Communications*, vol. 33, no. 10, pp. 1141 – 1150, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-4Y65SDF-1/2/664c2d6ccc40a5869afe5238b0c4493e

[23] P. Bermolen, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Abacus: Accurate behavioral classification of P2P-TV traffic," *Computer Networks*, vol. 55, no. 6, pp. 1394 – 1411, 2011.

[24] A. Finamore, M. Meo, D. Rossi, and S. Valenti, "Kiss to Abacus: A Comparison of P2P-TV Traffic Classifiers," in *Proceedings of the Second International Workshop on Traffic Monitoring and Analysis (TMA '10)*. Berlin Heidelberg: Springer-Verlag, 2010, pp. 115–126.

[25] "Tcpdump Homepage." [Online]. Available: http://www.tcpdump.org/(accessedApril10,2011)

[26] "OpenDPI Homepage." [Online]. Available: http://www.opendpi.org/(accessedApril10,2011)

[27] "L7-Netpdl Classifier Homepage." [Online]. Available: http://netgroup.polito.it/research-projects/l7-traffic-classification(accessedApril10, 2011)

[28] "L7-filter Homepage." [Online]. Available: http://l7-filter.sourceforge.net(accessedApril10,2011)

[29] "NetBee Library Homepage." [Online]. Available: http://www.nbee.org(accessedApril10,2011)

[This page was intentionally left blank]

# Chapter 3

## Identification of Encrypted Traffic Generated by Peer-to-Peer Live Streaming Applications Using Deep Packet Inspection

André F. Esteves, Pedro R. M. Inácio, Manuela Pereira, and Mário M. Freire

*Working Article to be Submitted for Publication in an International Journal*

**Noname manuscript No.**
(will be inserted by the editor)

# Identification of Encrypted Traffic Generated by Peer-to-Peer Live Streaming Applications Using Deep Packet Inspection

**Andre F. Esteves · Pedro R. M. Inacio · Manuela Pereira · Mario M. Freire**

Received: October 24, 2011 / Accepted: date

**Abstract** The popularity and number of applications using the peer-to-peer (P2P) networking paradigm has substantially grown over the last decade. These applications are not only used for content sharing, but also for media streaming, and they commonly encrypt the communication contents or employ protocol obfuscation techniques. In this article, we propose a method for identification of encrypted traffic flows generated by three of the most popular live streaming applications: TVUPlayer, GoalBit and Livestation. Our approach consists on devising a set of Snort rules that can be used to detect key network packets generated by these applications resorting to Deep Packet Inspection (DPI), which can then be used to identify the respective encrypted traffic flow. The proposed method is evaluated experimentally on a test-bed created for that purpose, being shown that its accuracy is about 85% for TVUPlayer, 97% for GoalBit and 100% for Livestation.

**Keywords** Peer-to-peer media streaming · Live streaming · Peer-to-peer networks · Traffic obfuscation · Traffic classification · Deep packet inspection

An earlier version of this paper has been presented at 10th IEEE International Symposium on Network Computing and Applications (IEEE NCA 2011), Cambridge, MA, USA August, 25 - 27, 2011 [21].

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia through the project TRAMANET: Traffic and Trust Management in Peer-to-Peer Networks, with contracts PTDC/EIA/73072/2006 and FCOMP-01-0124-FEDER-007253.

Andre F. Esteves, Pedro R. M. Inacio, Manuela Pereira, Mario M. Freire
Instituto de Telecomunicaes, Department of Computer Science, University of Beira Interior, 6201-001 Covilh, Portugal
Tel.: +351-2755-319891
Fax: +351-2755-319899
E-mail: m3412@ubi.pt, (inacio, mpereira, mario)@di.ubi.pt

## 1 Introduction

It has been recently predicted that Internet video traffic will increase significantly over the next years, being foreseen a sevenfold increase between 2009 and 2014 [37]. Besides the success of some web-based video streaming services, such as YouTube [65] and Tudou [58], and by the increasing interest in IP television (IPTV), peer-to-peer media streaming services are also emerging as a new paradigm for media streaming. In fact, media streaming over the Internet may be implemented either through IP multicast functionalities or through peer-to-peer (P2P) systems. Since the former approach is only feasible on network infrastructures managed by a single network operator due to limitations of IP multicast facilities, P2P-based approaches have been successfully investigated to overcome these limitations, having the potential to provide a scalable worldwide infrastructure [39]. Even services such as YouTube and Tudou are reported to use or are preparing to use P2P approaches, as mentioned in [68], in order to accelerate their downloading rates and to reduce the transmission cost.

The popularity and diversity of applications using the peer-to-peer (P2P) networking paradigm has substantially grown over the last decade. By exploring the resiliency and worldwide adoption of this technology, these applications are not only used for content sharing and distribution, but also for media streaming. To protect the privacy of the users (e.g., in Voice over Internet Protocol (VoIP) calls), bypass filtering/blocking mechanisms or overcome network limitations (e.g., firewall policies demanding for a session to be initiated from the inside of the network), it is common for these applications to employ evasion techniques, provide encryption for the contents and include the option to apply pro-

tocol obfuscation. To overcome firewall policies, a P2P application may, e.g., use the HyperText Markup Protocol (HTML) to start a session or authenticate to a server (an external third party) via the Transmission Control Protocol (TCP) port 80, for link establishment purposes. To avoid being classified with basis on port numbers, these applications offer the possibility to adjust the port number as a configuration parameter, or simply agree on the usage of a random one during the P2P link establishment phase. On the one hand, these functionalities favour the adoption of the applications, since they adapt to different network conditions and sometimes operate in very controlled scenarios but, on the other, they make the task of classifying and managing the network traffic a lot harder. This problem has been motivating several research works on this area, as further elaborated below.

P2P systems are suitable for transmitting media content [4, 53], but they have the disadvantage of generating traffic that may grow without control leading to a degradation of the network performance, with possible implications for network performance of the Internet service providers (ISPs) as perceived by Internet users [14], or may compromise critical networked applications of enterprises or institutions or other enterprise or institutional tasks that depend on the network to be carried out, albeit recently network friendly approaches, e.g., [2], have been followed for the development of applications for P2P media streaming (see e.g. WineStreamer [57]). Nevertheless, in most of those situations, network administrators may need to limit the transmission rate or even block connections for P2P traffic. However, recently, as mentioned above, several P2P applications evolved in order to implement protocol obfuscation or payload encryption making difficult to detect such kind of traffic, being therefore of great importance for network management to identify such encrypted traffic generated by P2P media streaming applications. Moreover, network managers often need to identify and manage this kind of traffic flows in real-time [7, 34], which imposes an additional challenge to overcome this problem.

During last years, a plethora of methods for classification of P2P network traffic have been reported, which may be broadly classified into three categories [10, 47, 60]: i) traffic classification based on port numbers, ii) traffic classification based on deep packet inspection (DPI), and iii) traffic classification in the dark. A general survey about Internet traffic identification is provided in [10]. A survey about approaches based on machine learning for Internet traffic classification is presented in [47]. A more updated survey and focused on the identification of P2P traffic is provided in [60].

Methods for P2P application detection based on port numbers are ineffective because most of the currently available P2P applications use either arbitrary ports or well-known ports that are assigned to specific application-layer protocols such as Hypertext Protocol (HTTP) to deceive classification mechanisms. Albeit in some specific cases they may be useful namely when combined with other features (see e.g. [40]), due to the above reason, this kind of methods is nowadays considered obsolete.

The category of deep packet inspection methods search on the entire data field of IP packets, often referred to as IP packet payload, for specific character patterns (also called signatures) associated with specific application-layer protocols. Usually those signatures are previously stored in a database. This kind of methods is used today not only for detection of application protocols, but also for detection of virus and malware through content filtering and also for intrusion detection and protection. This kind of methods has the advantage of high accuracy for known protocols or when the traffic is not encrypted. Key issues are the performance under high-speed network operation and the detection of encrypted traffic. An overview of recently investigated methods of this category is provided in [60]. Sometimes, it is claimed that DPI may lead to legal and privacy issues [3, 30, 33], since DPI implies that Internet Service Providers (ISP) or network managers are somehow accessing the content of the communications.

The category of traffic classification in the dark relies on the classification of the IP packets using behavioural or statistical patterns at a flow-level or through specific properties of the packets (excluding the data field of the packets), such as source/destination addresses or packet sizes. Encryption of the data field of the packets is not a problem for this kind of methods because they do not analyse the data field contents, but usually they suffer from lack of accuracy. This category of methods comprehends a plethora of strategies, including statistical approaches, e. g. [15], behavioural approaches, which may be based on specific flavours such as social interaction [32], entropy [28], recurrence quantification analysis [50], data transfer behaviour [63], and a variety approaches relying on machine learning (ML) for traffic classification, which may be further categorized as unsupervised [18, 67], semi-supervised [19, 52], or supervised ML approaches [6, 8, 16, 45, 48, 51, 66].

A few hybrid classifiers combining approaches of the categories above mentioned have also been reported. In [13], Chen et al. proposed an online hybrid traffic classifier for P2P traffic combining a classifier imple-

mented in hardware through a network processor and a classifier implemented through software based on a machine learning approach. The classifier implemented in hardware combines a method based on TCP/UDP ports and a method based on deep packet inspection, whereas the classifier implemented through software uses a Flexible Neural Tree model. In [29], Gomes et al. proposed a traffic classification mechanism combining a method for traffic behaviour characterization and deep packet inspection.

This article describes the research work we have conducted towards the identification of encrypted traffic flows generated by three of the most popular P2P live streaming applications: TVUPlayer, GoalBit and Livestation, albeit Livestation has discontinued the use of P2P approaches. Our approach consists on finding rules or signatures for detection of a specific P2P application in Internet Protocol (IP) packets belonging to the parts of the communication that follows unencrypted. These parts may either be related with the *link establishment phase* or with the *signalling protocol*. The proposed method makes use of an open source and widely used intrusion detection system, called Snort [54]. The rules are coded into the Snort syntax for application detection through Snort. The detected packets are then contextualized in a flow, and used to identify the encrypted traffic. This article comprises an extended and more detailed version of [21], since this earlier version only considered GoalBit in a trackerless configuration and this article considers both trackerless and tracker-based configurations of GoalBit and also takes into account two other popular applications: TVUPlayer and Livestation.

The remainder of this article is structured as follows. Section 2 discusses related works on this area of knowledge. Section 3 provides an overview about P2P media streaming applications, explaining their way of functioning. Section 4 describes the network test-bed used in the scope of this work. Section 5 summarizes and discusses the rules and the results concerning their accuracy. Section 7 contains the final conclusions and directions for future work.

## 2 Related Work on Identification of Encrypted P2P Media Streaming Traffic

A large amount of papers has been published about P2P traffic classification with particular emphasis on the traffic detection and classification in the dark. As a consequence, several methods have been proposed to classify P2P traffic, without the specific goal of detecting encrypted P2P media streaming traffic, but including this kind of applications in the set of applications to detect and classify. Examples of methods that illustrate these situations include: the methods proposed by Liu et al. [42], Carela-Espanol et al. [41], Chen et al. [13], and Wang et al. [61], which were used to classify several applications including PPLive, the methods proposed by Liu et al. [24] and Gao et al. [17], which were used to classify PPLive and PPStream among others and the method proposed by Dusi et al. [11] which was used to classify PPLive and SopCast among other applications. However, in most of these cases, the P2P media streaming traffic was not encrypted, being therefore no particular attention paid to traffic obfuscation, since this kind of classification techniques does not analyze the content of the packet payload. In the following, it is presented an overview about methods addressing the specific issue of the identification of traffic generated by P2P media streaming applications.

In [38], Lee et al. proposed NeTraMark, a network traffic classification benchmark that includes implementations of eleven existing classification algorithms ranging from port and deep-packet inspection to graph-based classifiers. Among several categories of applications to be classified, it comprehended the streaming category, where Abacast, an hybrid P2P application, was included.

In [31], Hu et al. presented a profile-based approach to identifying traffic flows belonging to a given target application. They built behavioral profiles of the target application that describe dominant patterns of the application. Based on the behavior profiles, a two-level matching method is used to identify the traffic generated by that target application. The method has been tested with BitTorrent and PPLive with an average accuracy ranging from 92.6 to 98.7. Since, this method is based on the packet header and flow statistics (packet size and inter-arrival time), it may be used to identify encrypted traffic provided that the encryption is limited to the packet payload.

In [12], a comparison among three different traffic classifiers for peer-to-peer television (P2P TV) applications is presented: a DPI method, a method based on single-class Support Vector Machines (SVMs), known as IPSVM classifier [20], and a method based on multi-class SVMs, known as KISS classifier [22], which needs to perform payload analysis, being therefore ineffective for encrypted traffic. The considered applications were used without encryption or obfuscation techniques. It is shown that the DPI approach is very effective mainly due to the fact that those applications do not use encryption or obfuscation techniques, being appointed the main disadvantage of this approach, which relies on the process of deriving the signatures that is still manual, time-consuming and error-prone. It is also shown that

other two methods relying on single-class or multi-class SVMs replace the signature database with a suitable automatic training phase and they guarantee very good results provided that the training set is appropriate.

In [63], Xu et al. proposed a content-based partitioning scheme to discover shared data and identify the P2P data transfer behavior, being applied to some specific applications of P2P file sharing, P2P live streaming and P2P video on demand. It is shown through experiments that the proposed scheme has a high accuracy. However, the proposed method needs to perform payload analysis being therefore ineffective for classification of encrypted traffic.

In [9], Bermolen et al. proposed a behavioral approach relying on the count of packets and bytes exchanged among peers during small time-windows and using support vector machines for fine-grained classification of P2P media streaming applications. It is shown that the proposed method, known as Abacus, correctly classifies about 95% of packets, bytes and peers in the worst case. It is also shown that the number of false alarms is very small, being below 0.1% in the worst case. The study included the following P2P applications: PPLive, TVAnts, SopCast, and Joost, where Joost (which after became a non-P2P web-based application since October 2008) was the only one allowing traffic obfuscation.

In [23], Finamore et al. presented a comparison between Abacus [9] and Kiss [22]. It is shown that both approaches are comparable in terms of accuracy in classifying P2P media streaming applications, at least regarding the percentage of correctly classified bytes and for non-encrypted traffic. However, in terms of the computational cost of the classifiers, Abacus presents a better performance than KISS, but KISS is much more general, as it can classify other kinds of applications as well. However, KISS is ineffective for encrypted traffic whereas Abacus does not suffer from this drawback.

## 3 Overview About P2P Media Streaming

Recently, several papers addressing P2P media streaming issues have been published including a few surveys. A survey on peer-to-peer video streaming systems has beed presented by Liu et al. in [43], a survey about P2P overlay streaming clients has been presented by Sentinelli et al. in [5], an overview about incentive mechanisms in P2P media streaming systems has been reported by Su and Dhaliwal in [55] and an Internet-Draft surveying P2P streaming applications has been proposed by Gu et al. [64]. A survey about resilient P2P video streaming has been presented by Abboud et al. in [1] and a survey about security and privacy issues in P2P media streaming has been reported by Gheorghe et al. in [25].

P2P media streaming systems may be broadly classified into three categories according to the structure of the overlay [64]: tree-based, mesh-based and hybrid systems. The category of tree-based systems may be further split into two categories: single-tree streaming and multi-tree streaming. The category of mesh-based systems may also be subdivided into two categories: pull-based and push-pull [55].

Tree-based systems implement a tree distribution graph being the root node the video source server. Each node joining the tree will receive the video stream from his parent peer which is then transmitted to its children peers [43]. This kind of systems presents two drawbacks: first, the media streaming bandwidth from root to any node can not exceed the smaller bandwidth of any internal node along the path; second. leaf nodes, at the bottom of the tree, do not contribute with any bandwidth. However, this problem may be overcome through multiple trees [55]. On the other hand, this structure has typically a small maintenance cost [64].

Mesh-based systems organize peers in a dynamic mesh, in which a peer receives media chunks from multiple peers. This kind of systems does not need to construct and maintain a streaming topology as in tree-based systems and is less vulnerable to the dynamics of peer participation [55]. The reliability of the transmission in this structure is higher and allows higher bandwidth usage regarding tree-based systems [64].

Hybrid systems combine the best features of tree-based and mesh-based systems but also push-based and pull-based content delivery to improve the performance. They have high reliability as mesh-based systems, lower delay than mesh-based systems, lower overhead associated each video block transmission and high topology maintenance cost as in mesh-based systems [64].

Table I summarizes popular P2P media streaming applications, providing their classification in terms of the structure of the overlay and their ability to generate encrypted traffic. In this article, we are particularly concerned with P2P applications that allow protocol obfuscation. Therefore one of the selected applications to be used in this article was GoalBit because it allows protocol obfuscation.

## 4 Method and Experimental Test-bed for Detection of Encrypted Traffic

The experimental test-bed was implemented in the Multimedia Signal Processing (MSP-Cv) Lab of Instituto de Telecomunicacoes at Department of Computer Sci-

**Table 1** List of popular P2P media streaming applications with information if they allow traffic encryption.

| Structure of the overlay | Popular P2P streaming applications | Data encryption |
|---|---|---|
| Mesh-based P2P streaming systems | Octoshape | Yes |
| | PPLive | No |
| | Zattoo | Yes |
| | PPStream | No |
| | SopCast | No |
| | TVants | No |
| | GoalBit | Yes |
| Tree-based P2P streaming systems | PeerCast | No |
| | Conviva | No |
| Hybrid P2P streaming systems | New Coolstreaming | No |
| | MultiPeerCast | No |

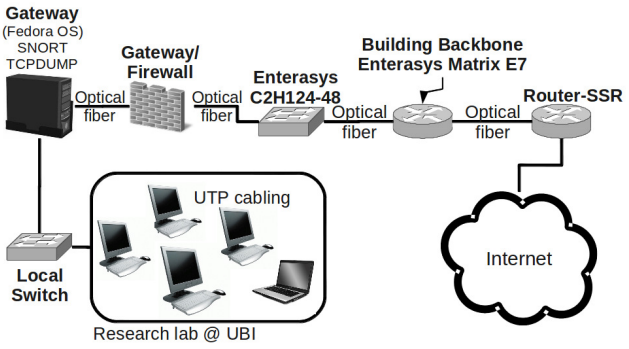


**Fig. 1** Experimental test-bed at the MSP-Cv research lab.

ence of the University of Beira Interior. The lab contains 24 desktop computers running Fedora 9 or Windows Server 2003 operating systems. There is also a computer functioning as an auxiliary gateway between these desktop computers and the main lab gateway. This auxiliary gateway manages all incoming and outgoing traffic in that lab, runs the Fedora 9 operating system and has the `tcpdump` [56], SNORT, `OpenDPI` [49], `l7-netpdlclassifier` [36] and `l7-filter` [35] applications installed. `Tcpdump` was used for capturing traffic and for analysing it. After analysing the traffic to find patterns, several rules were created using the SNORT syntax.

The signatures of payloads of packets generated by media streaming applications to be identified are expressed in terms of Snort rules. The method for specification of Snort rules for detection of packets generated by those media streaming applications requires the execution of the following steps: i) identification of signatures associated with a given application that can be revealed through the analysis of the payload of a corresponding IP packet in a given phase of the connection; ii) writing Snort rules incorporating these signatures in order to detect all packets with that particular signatures and identification of the corresponding flow. The identification of signatures associated with a given application packets was made manually through an exhaustive observation of patterns in the sequence of packets generated by that media streaming application, even with obfuscation (encryption of the payload). Those common patterns (specific string series) observed in the payload were then used to manually write the rules for the detection engine of the Snort, which allows the identification of a given packet that contains that particular pattern in a particular position or after a given offset in its payload.

The other classification applications mentioned above were used to provide a base of comparison in order to validate the set of rules created for SNORT, starting with `OpenDPI` which is an open-source version of Protocol and Application Classification Engine (PACE), a traffic classification engine from ipoque. `OpenDPI` use mainly DPI to classify network traffic and as the particularity of not supporting the detection of encrypted protocols. `L7-netpdlclassifier` was developed by the Computer Networks Group (NetGroup) of *Politecnico di Torino* and makes use of the NetBee [46] library. The signatures are written using the Netpdl language (a XML-based syntax), the output it produces is a packet-based output file (i.e., the classification result for each packet). `L7-filter` is a well known open-source application that makes use of DPI for network traffic classification, this application uses regular expressions to define the payload signatures and produce a flow-based output (i.e., the classification result for each flow).

One of the most important requirements for running P2P software is the hard-drive space. On the other hand, real-time network monitoring requires more computational resources, due to the fact that processing the captured traffic and the intrusion detection system consume a lot of Central Processor Unit (CPU) power and Random Access Memory (RAM). Table 2 summarizes the characteristics of the hardware such as CPU and RAM and also the software that was running on the above mentioned computers. In the lab, each set of four desktops are connected to a single switch using Unshielded Twisted Pair (UTP) Cat.5e/6 cabling, which is then connected to the auxiliary gateway using the same type of cabling, which in turn connects to the network backbone device (an Enterasys E7 switch) using optical fiber. All external communications with the University are managed by the Enterasys SSR main router located in the University Computing Center. A schematic for the topology of the test-bed is included in Figure 1. The main idea behind the proposed method relies on the finding of Livestation, TVUPlayer and Goalbit sig-

**Table 2** Hardware and software characteristics of the equipment used in the research lab test-bed.

| Type and Operating System | CPU | RAM | Software |
|---|---|---|---|
| Desktop Windows Server 2003 | Pentium D 2.66Ghz | 1GB | GoalBit, Skype, Youtube, eMule, TVUPlayer, Livestation, SopCast, PPStream, UStream, etc. |
| Laptop Windows 7 | Core 2 Duo 2.4Ghz | 3GB | GoalBit, Windows Live Messenger, HTTP and audio streaming |
| Laptop Ubuntu 10.10 | Core Duo 1.8Ghz | 1GB | Skype |
| Desktop Fedora 9 | Pentium D 2.66Ghz | 1GB | Skype |
| Gateway Fedora 9 | Xeon 2.26Ghz | 6GB | tcpdumd, SNORT, l7-netpdlclassifier, l7-filter, OpenDPI |

natures (repetitive string series) in the data field of IP packets, in some critical points of the link phase (e.g. establishing phase and further non-encrypted communication during data transfer phase), when encryption is not used, which allows the identification of the sessions associated with the used P2P-TV applications. Based on the signatures discovered using human observation, it was possible to identify packets carrying Livestation, TVUPlayer, and GoalBit traffic and, from there, classify the flows generated by these applications. Besides the option of identifying packet traffic, there is also the option to identify traffic by flows, and there is were our method also relies to improve accuracy. In our method, the traffic flow classification is achieved after identifying packets that belong to Livestation, TVUPlayer, or Goalbit traffic flow within a clean filtered trace file containing only the application in question, and applying then to the trace file containing all the captured traffic to evaluate the method accuracy.

## 5 Application Details and SNORT Rules

### 5.1 Livestation

Livestation is a P2P-TV platform that provides live streaming channels of several international television news channels. The users of this application can watch those channels in three different ways: the desktop player, the Livestation website [44], or using a mobile device (e.g., Iphone, Blackberry, Playstation Portable (PSP)).

To watch the free channels, one must previously create a free account, which is used for keeping and applying the configurations relative to that specific user at logon/logout. Livestation currently offers live TV news channels like BBC World News, Euronews in a diversity of languages, Al Jazeera, and France24 among others. To watch a channel, one just have to select it from the list of user provided channels. Livestation also provide a chat and Twitter feature, so one can comment a channel or discuss some relevant news.

The establishing link phase of Livestation is where DPI can detect some important signatures, because it is before the login that encryption is not yet applied, but also during data transfer phase some non-encrypted communication can be detected. Livestation only generate TCP and HTTP flows and there was where we based our analysis. That fact allowed the creation of the rules presented on table 3. To build these rules, traffic traces of Livestation were captured and analysed. The rule set for Livestation was created using a threshold count of 3 alerts per rule at each 10 seconds, i.e., at each 10 seconds, SNORT only emits a maximum number of 3 alerts per rule.

### 5.2 TVUPlayer

TVUPlayer is amongst the most well known P2P TV applications and it can be obtained at the TVU Networks site [59]. The channel list available in TVUPlayer is divided into several categories, (e.g., *news*, *movies*, *music*, *cartoons* and *sports*) and includes international channels from anywhere in the world (e.g., ABC, FOX News, NBC). Registering at the TVUPlayer web site is free and anyone with an account, a PC and an Internet connection, can create and broadcast or re-broadcast a channel uninterruptedly through the TVUBroadcast application, making it available in TVUPlayer for anyone to watch. It has a user-friendly interface, offering simple selection of channels, where one can inclusively also see the country of origin of the streamed channel. The channels used within the scope of this work were all certified by TVU Networks, which are identified with a TVU Networks logo.

TVUPlayer generate TCP, UDP and some HTTP traffic. UDP is mainly generated when the stream is being distributed, TCP and HTTP traffic is generated when a request for information about the channel is issued or when signalling and peer connection control messages are sent. Table 4 contains the set of rules

**Table 3** Set of SNORT rules for detection of Livestation IP packets. These rules apply to TCP flows.

| Rule number | Definition in SNORT syntax |
|---|---|
| Rule 1000511 | `alert tcp $HOME_NET any -> $EXTERNAL_NET 443 (msg:"Livestation Handshake"; flow:from_client, established; content:"\|16 03 00 00 59 01 00 00 55 03\|";threshold:type threshold, track by_src,count 3,seconds 10;sid:1000511;rev:2;)` |
| Rule 1000512 | `alert tcp $EXTERNAL_NET 443 <> $HOME_NET any (msg:"Livestation Application Data"; flow:from_server, established; content:"livestation.com";threshold:type threshold, track by_src,count 3,seconds 10;sid:1000512;rev:2;)` |

**Table 4** Set of SNORT rules for detection of TVUPlayer IP packets. These rules apply to UDP flows.

| Rule number | Definition in SNORT syntax |
|---|---|
| Rule 1000500 | `alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"LocalRule: P2PTV TVUPlayer (packet payload size 57bytes)"; content:"\|01 00 00 00 00 2d 00\|"; threshold:type threshold, track by_src, count 3,seconds 30; sid:1000500; rev:1;)` |
| Rule 1000501 | `alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule: P2PTV TVUPlayer (packet payload size 32bytes)"; content:"\|01 00 00 00 00 00 14 00\|"; threshold:type threshold, track by_src, count 3,seconds 10; sid:1000501; rev:1;)` |
| Rule 1000502 | `alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule: P2PTV TVUPlayer (packet payload size 32bytes)"; content:"\|00 00 14 00 00 04\|"; threshold:type threshold, track by_src, count 3,seconds 30; sid:1000502; rev:2;)` |
| Rule 1000503 | `alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule: P2PTV TVUPlayer (https)"; content:"\|00 3b 00 01\|"; threshold:type threshold, track by_src, count 3,seconds 10; sid:1000503; rev:1;)` |
| Rule 1000504 | `alert udp any any -> any any (msg:"LocalRule: P2PTV TVUPlayer (packet payload size 1280bytes)"; content:"\|01 01 00 00 00 04 f4\|"; threshold:type threshold, track by_src, count 3,seconds 30; sid:1000504; rev:1;)` |
| Rule 1000505 | `alert udp any any -> any any (msg:"LocalRule: P2PTV TVUPlayer \|01 01 00 00 00\|"; content:"\|01 01 00 00 00\|"; threshold:type threshold, track by_src, count 3,seconds 30; sid:1000505; rev:1;)` |

created after analysing the traffic generated by TVUPlayer. These rules are all applied to UDP traffic, since it is in the UDP traffic that the contents are transmitted and, in this case, communications are not encrypted. Rules 1000501 and 1000503 were created using a threshold count of 3 alerts per rule at each 10 seconds, and rules 1000500, 1000502, 1000504 and 1000505 were created using a threshold count of 3 alerts per rule at each 30 seconds. The threshold is different in some rules, due to the amount of alerts triggered, and to spare disk space different thresholds were used.

### 5.3 GoalBit

GoalBit is an open-source video streaming application that runs on Windows, GNU/Linux, Mac and even Solaris. It was initially developed by Uruguayan researchers for experiments concerning P2P networks. GoalBit enables its user to not only watch to streamed channels, but to also broadcast his or her own multimedia contents. GoalBit supports several video sources such as files, Hypertext Protocol / Microsoft Media Services / Real-Time Protocol (HTTP/MMS/RTP) streams, webcams , DVD devices, etc. It also supports multiple video formats like Windows Media Video (WMV), Divx, H.264 and others.

We tested four different scenarios using GoalBit: two scenarios where encryption was being used and two scenarios where encryption was turned off. For each one of the scenarios, two types of peer connections were used: a tracker-based streaming system (similar to BitTorrent), provided by the GoalBit Starter Suite found in [26]; and a trackerless-based streaming system using the Kademlia protocol. The Kademlia protocol configuration implemented for testing purposes, was for the local network only, since the computers were all in the same IP address space network. This type of configuration start by manually adding the IPs of the computers running GoalBit to a file made for that purpose, and which has a template that we can follow. In that file, we add the IPs, the port used for each computer since

it needs to be different for each one, the hexadecimal identification code generated in the template file. After that we ran GoalBit in each computer by console, using the following command:

goalbit -vvv –enable-kad=1 –kad-ini-file = configuration_file.met2

However Kademlia protocol does not need to be configured always by this way, this configuration was need because all the computers were running in the same IP address space. The GoalBit Starter Suite is a web application written in Hypertext PreProcessor (PHP) and MySQL, with a tracker incorporated that makes possible for viewers and broadcasters to communicate between them. If one wants to use this second option, he or she must install Apache, PHP, and MySQL (instructions for installing the Starter Suite can be found in [27]). When we used the tracker-based system, Goalbit generated TCP traffic exclusively, contrarily to the TCP and UDP traffic generated in the trackerless-based system.

The computers running the application were all logically connected to the same network of the research lab, except for a laptop that was connected to a wireless network with an address space different from the wired one. The computer connected to the wireless network was the one broadcasting the stream.

Encryption in GoalBit is provided as an output parameter, so the broadcaster can choose if wants to encrypt the stream content with a symmetric key using the Advanced Encryption Standard (AES) algorithm, or leave the streamed contents available for everyone. The rules presented in table 5 and 6 are the result of analysing both tracker-based and trackerless-based traffic, in which rules 1000508, 1000562, 1000563, and 1000565 are exclusive to encrypted traffic, and the rest of the rules are common to traffic with encryption and without it. Table 5 includes the rules for identification of TCP flows, while table 6 includes the rules for identification of UDP flows. In order to reduce the disk space usage (another computational burden), GoalBit rule set was created using a threshold count of 3 alerts per rule at each 10 seconds.

## 6 Experiments and Result Discussion

The results discussed in this section represent the work performed for the traffic traces captured during four different days as shown in table 7. Also in table 7 and to clarify, in experiment #1 we used Kademlia trackerless feature with encryption provided by GoalBit, in experiment #2 tracker-based with and without encryption was used provided by GoalBit Starter Suite, and finally in experiment #3 we used the Kademlia feature without encryption.

During the experiments and in order to generate credible background traffic, the 24 desktop computers were running different applications, namely P2P-TV (TVUPlayer, GoalBit, Sopcast, PPStream,Livestation), P2P file-sharing applications (eMule, FrostWire), VoIP applications (different versions of Skype), online audio streaming and online flash video streaming (Youtube and UStream).

Based on the signatures discovered using human observation, it was possible to identify packets carrying Livestation, TVUPlayer, and GoalBit traffic and, from there, classify the flows generated by these applications. As expected (we are dealing with encrypted traffic), the true positive rate for the classification of the traffic at the packet level was below 25%. Nonetheless, a small number of packets was sufficient to identify most of the flows generated by the application, by configuring SNORT to classify the entire flow to which the detected packet belongs to.

For a better understanding of how our method was accurate, we did not ran the set of rules against the entire trace file at the first time, i.e., the true positive rate was calculated after using only traffic filtered from the trace file, that contained only the P2P-TV application generated traffic, and then we ran the set of rules against all the trace file to check the accuracy of our method. False positives, can be explained as the amount of traffic classified as belonging to one of the three P2P-TV applications related on this paper that in reality is not. In these experiments, the amount of false positives reach a maximum of 3%, being a small amount, comparatively to the true positive rate.

The total number of TCP packets captured during the experiments for Goalbit can be found on figure 2. The chart depicts the number of packets against their size (in bytes), and it is useful to make a note on the fact that this type of traffic is mainly constituted by very large packets (carrying the stream contents per se) and by small packets, mainly constituted by TCP acknowledgements and peer signalling communication. Livestation newest versions stopped being P2P-based and started to be managed by central-servers who assigns a stream for each user. After analysing the traffic generated by Livestation through Wireshark [62], it was not any sign of more than one broadcaster or any other IP address besides the computer receiving the stream and the broadcaster itself. Rules 1000511 and 1000512 were created after checking the repetition of signatures on packets generated by Livestation. Although the count of triggered rules was small as one can see in figure 5, it allowed to detect the flow of the application.

**Table 5** Set of SNORT rules for detection of GoalBit IP packets. These rules apply to TCP flows.

| Rule number | Definition in SNORT syntax |
|---|---|
| Rule 1000506 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Tracker Cookie"; content:"\|47 67 61 6c 42 69 74 20 70 72 6f 74 6f 74 6f 63 6f 6c\|"; dsize:77; threshold:type both, track by_src,count 1,seconds 10; sid:1000506; rev:1;)` |
| Rule 1000509 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 0d 06 00 00\|"; flow: established; content:"\|00 0d 06 00 00\|";threshold:type both, track by_src,count 3,seconds 10; sid:1000509; rev:2;)` |
| Rule 1000510 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 05 0a 00 00\|";content:"\|00 05 0a 00 00\|"; dsize:9; threshold:type both, track by_src,count 3,seconds 10; sid:1000510; rev:1;)` |
| Rule 1000564 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP \|00 00 00 00 01 00 00\|";content:"\|00 00 00 00 01 00 00\|"; offset:10; depth:17; dsize:17; threshold:type both, track by_src,count 3,seconds 10; sid:1000564; rev:1;)` |
| Rule 1000565 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP Payload Size (1460bytes)";content:"\|62 72 6f 61 64 63 61 73 74 65 72 7b 70 69 65\|"; depth:90; dsize:1460; threshold:type both, track by_src,count 3,seconds 10; sid:1000565; rev:1;)` |
| Rule 1000566 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP Payload Size (1460bytes)";content:"\|67 6f 61 6c 62 69 74 5f 74 72 61 63\|"; dsize:1460; threshold:type both, track by_src,count 3,seconds 10; sid:1000566; rev:1;)` |
| Rule 1000567 | `alert tcp any any -> any any (msg:"LocalRule: GoalBit Pattern TCP \|00 09\|";content:"\|00 09\|"; dsize:13; threshold:type both, track by_src,count 3,seconds 10; sid:1000567; rev:1;)` |

**Table 6** Set of SNORT rules for detection of GoalBit IP packets. These rules apply to UDP flows.

| Rule number | Definition in SNORT syntax |
|---|---|
| Rule 1000508 | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|00 c2 01 e3\|";content:"\|00 c2 01 e3\|"; dsize:474; threshold:type both, track by_src,count 3,seconds 10; sid:1000508; rev:1;)` |
| Rule 1000560 | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|56 9c dd ed\|";content:"\|56 9c dd ed\|"; dsize:24; threshold:type both, track by_src,count 3,seconds 10; sid:1000560; rev:1;)` |
| Rule 1000561 | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|e3 0c 02\|";content:"\|e3 0c 02\|"; dsize:25; threshold:type both, track by_src,count 3,seconds 10; sid:1000561; rev:1;)` |
| Rule 1000562 | `alert udp any any -> any any (msg:"LocalRule: GoalBit Pattern \|e3 0b\|";content:"\|e3 0b\|"; dsize:33; threshold:type both, track by_src,count 3,seconds 10; sid:1000562; rev:1;)` |

**Table 7** Characteristics of captured traffic

| Experiments | Date | Start time | End Time | Volume (GB) | Number of Packets | GoalBit Streaming Configuration |
|---|---|---|---|---|---|---|
| Experiment 1 | 04-03-2011 | 12h17 | 19h31 | 29.7 | 58995815 | Tracker-less with Encryption |
| Experiment 2 | 18-04-2011 | 17h46 | 19h31 | 5.7 | 6534533 | Tracker-Based with and without Encryption |
| Experiment 3 | 19-04-2011 | 14h15 | 16h05 | 5.1 | 5567863 | Tracker-less without Encryption |

Turning to the rules created based on observation of traffic generated by TVUPlayer, UDP traffic was the one carrying the stream contents so makes sense to look for signatures on UDP traffic and after human observation, it were discovered several signatures that were repeating themselves over the packets of the flow of the traffic generated by TVUPlayer. As said on section 5, TVUPlayer also generated TCP and HTTP traffic but no signatures were found on these packets that could be used for creating rules. Like Livestation the count of triggered rules is presented on figure 5.

GoalBit TCP rules were created based on periods in which encryption was not being used, like the start of the transmission and also some parts in the mid-

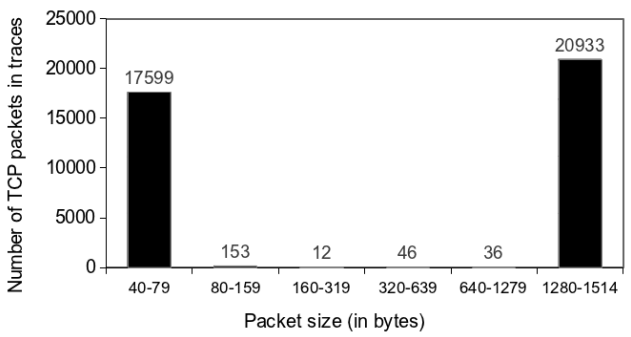


**Fig. 2** Number of TCP packets in the collected traces vs. the packet size.

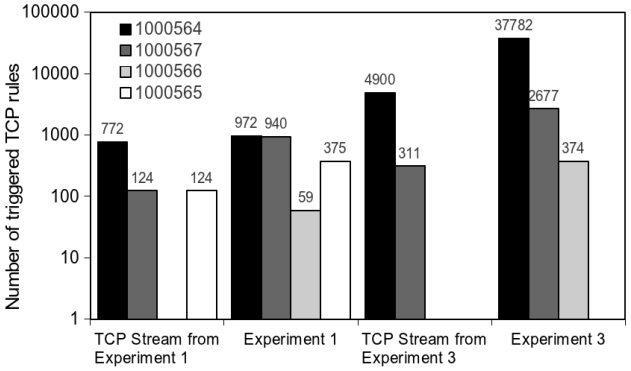


**Fig. 3** Count of triggered TCP rules by two TCP streams filtered from GoalBit traffic experiments. For the sake of clarity, the y-axis is in logarithmic scale.

dle of the transmission. Rule `1000509` was observed on GoalBit traffic but when tested against the traces we captured also matched non-GoalBit traffic, accounting this way for the false positive rate. Rule `1000506` was being triggered both when encryption was used but also when it was not, the main issue in using this rule is that it only was triggered when we have the first bytes of the TCP session, but it is still quite useful to detect GoalBit traffic. Two rules were created after observing packets with large payload sizes, they are rule `1000565` with 1460 bytes and rule `1000566` with 1456 bytes respectively, the packets with this payload signatures can be found scatter all over the GoalBit traffic, however they are few packets triggered by these two rules, meaning that these packets were not using encryption.

The UDP rules were created also by observing GoalBit traffic, but in this case the signatures observed were from the Kademlia protocol used by GoalBit in two of the traces, for this reason all applications that make use of this protocol are also triggered as GoalBit traffic. The UDP rules do not appear in the charts, because they were triggering all applications that use the Kademlia protocol. Nonetheless the UDP packets were carrying information not relative to the content being distributed, but the information carried in the UDP packets was useful for peers communicating between themselves and also for making them known in the network.

Due to the low true positive rate a flow classification approach was used. In the flow classification we could identify multiple packets using the rule set and because the flow actually belonged to GoalBit we classified all the packets on that flow as GoalBit. Using the flow approach we managed to increased the true positive rate by 77%, making a total of 96% true positive rate. As we were running the rule set, some false positives were raised, due to a signature that identifies GoalBit as well as other traffic, accounting for a total of 1.3%. We have a small percentage of false negatives 2.7% due to multiple flows only carrying ACK,SYN and PUSH flag packets.

As in our test-bed the true positive rate was a little low as said before, we started by classifying the traffic using a flow approach which will be explained below. We grabbed in a random trace and we filtered out the ip's from the machines that were just running GoalBit, then protocols like ARP, NBNS, ICMP, HTTP, MSN and ICMPv6 were also removed because they were of no use to create the set of rules. From there, we then filtered a random TCP stream using the Wireshark "Follow TCP stream" feature. After the previous step we had a trace file containing only the chosen TCP stream, we ran SNORT to see if it triggered any rules, and as hoped it triggered several rules has shown in figure 3. Table 8 summarizes the results in terms of accuracy and completness, as defined in [10], for the identification of Livestation, TVUPlayer and GoalBit.

To a better performance of our rule set we run it against other DPI software like OpenDPI, l7-filter and l7-netpdlclassifier. We can see by figure 4 that the applications tested failed to classify GoalBit, the detection rate referred in figure 4 is the total accuracy percentage of the set of rules created without false positive or false negatives, i.e., it is the percentage of detection at packet level relative to the total number of packets generated by the applications.

For the sake of clarity, l7-filter and l7-netpdlclassifier with the latest set of rules provided by their contributors, have not yet built rules that could identify Livestation, TVUPlayer, and GoalBit applications. Detection rate percentage was an average calculation from multiple results. Livestation got a 100 percent detection rate because we identified the first few packets of the flow and due to being transmitted by only one IP we could identify the whole flow as being a Livestation flow. The accuracy means that the traffic detected really belongs

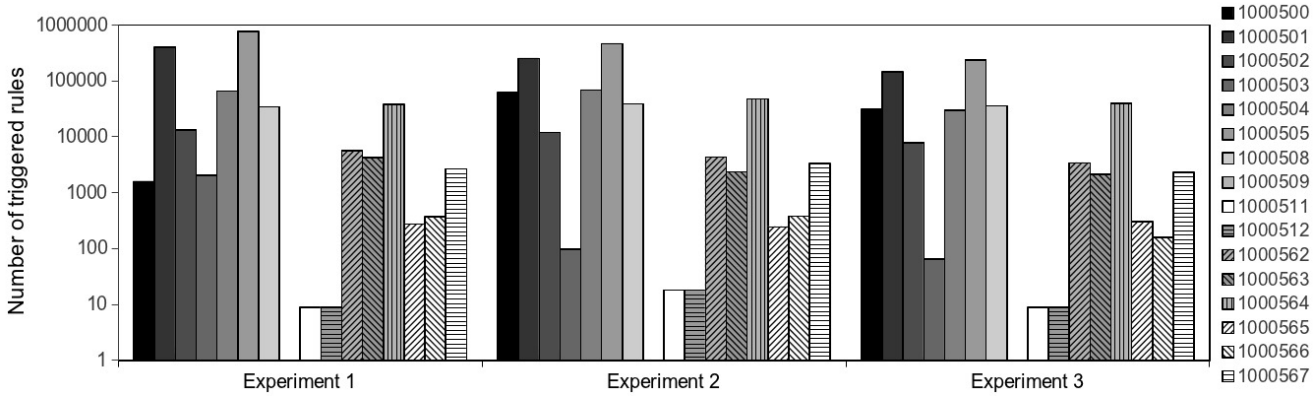


**Fig. 5** Number of rules triggered in each one of the three days of experiments. For the sake of clarity, the y-axix is in logarithmic scale. Experiment 1 refers to the traces collected in March 4, 2011; experiment 2 refers to the traces collected in January 17, 2011; experiment 3 refers to the traces captured in January 21, 2011 (as described in table 7).

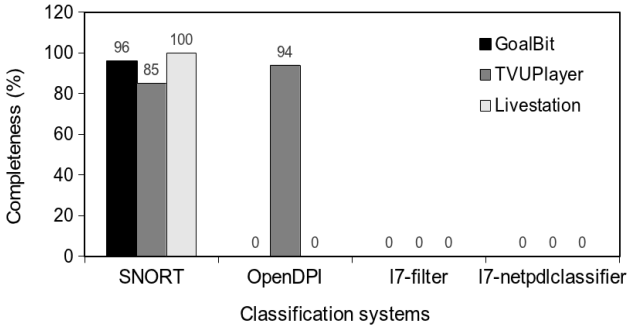


**Fig. 4** Comparison of completeness percentage between the SNORT rules, OpenDPI, l7-netpdlclassifier and l7-filter for the captured traces.

**Table 8** Accuracy and Completeness results

| | Livestation | TVUPlayer | GoalBit |
|---|---|---|---|
| Accuracy | 100% | 95% | 97% |
| Completeness | 100% | 85% | 96% |

to the applications that indeed generated it. Completeness means that the detected traffic belongs to a specific rule set. Table 8 present the accuracy and completeness results for this research.

## 7 Conclusion and Future Work

This article tackles the problem of identifying encrypted traffic generated by P2P live streaming applications. Our approach resorts to DPI and relies on the assumptions that there are parts of the communication that follow unencrypted and that such parts can be used to correctly identify the remaining traffic associated to the corresponding flow of the media streaming application. It is shown that the proposed method achieves an accuracy of 100% for Livestation, 95% for TVUPlayer and 97% for GoalBit. The corresponding completenesses are 100%, 85% and 96%. It was also put in evidence the better performance of the Snort with the defined rules regarding other systems, namely OpenDPI, l7-netpdlclassifier and l7-filter. The possibility to automatize the procedure of deriving the rules for SNORT from the traces is left for future work.

**Acknowledgements** The authors would like to acknowledge David A. Carvalho for his assistance in the configuration of the network test-bed.

## References

1. Abboud O, Pussep K, Kovacevic A, Mohr K, Kaune S, Steinmetz R (2011) Enabling resilient p2p video streaming: Survey and analysis. Multimedia Systems 17(3):177–197
2. Abeni L, Bakay A, Biazzini M, Birke R, Leonardi E, Cigno RL, Király C, Mellia M, Niccolini S, Seedorf J, Szemethy T, Tropea G (2010) Network Friendly P2P-TV: The Napa-Wine Approach. In: Peer-to-Peer Computing, Delft, The Netherlands, pp 1–2
3. Aceto G, Dainotti A, de Donato W, Pescape A (2010) Portload: Taking the best of two worlds in traffic classification. In: Proceedings of INFOCOM IEEE Conference on Computer Communications Workshops , 2010, pp 1 – 5
4. Alessandria E, Gallo M, Leonardi E, Mellia M, Meo M (2011) Impact of adverse network conditions on P2P-TV systems: Experimental evidence. Computer Networks
5. Alexandro Sentinelli DLCPGPTZAJ Luca Celetto (2009) Survey on P2P Overlay Streaming Clients. Towards the Future Internet - A European Research Perspective pp 273–282
6. Angevine D, Zincir-Heywood AN (2008) A preliminary investigation of Skype traffic classification using a minimalist feature set. In: Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 08), IEEE Computer Society Press, Barcelona, Spain, pp 1075–1079

7. Apiletti D, Baralis E, Cerquitelli T, D'Elia V (2009) Characterizing network traffic by means of the NetMine framework. Comput Netw 53:774–789
8. Auld T, Moore AW, Gull SF (2007) Bayesian neural networks for Internet traffic classification. IEEE Transactions on Neural Networks 18(1):223–239
9. Bermolen P, Mellia M, Meo M, Rossi D, Valenti S (2011) Abacus: Accurate behavioral classification of P2P-TV traffic. Computer Networks 55(6):1394 – 1411
10. Callado A, Kamienski C, Szabo G, Gero B, Kelner J, Fernandes S, Sadok D (2009) A Survey on Internet Traffic Identification. IEEE Communications Surveys & Tutorials 11(3):37–52
11. Carela-Español V, Barlet-Ros P, Cabellos-Aparicio A, Solé-Pareta J (2011) Analysis of the impact of sampling on netflow traffic classification. Comput Netw 55:1083–1099
12. Cascarano N, Risso F, Este A, Gringoli F, Finamore A, Mellia M (2010) Comparing P2PTV traffic classifiers. In: Proceedings of the IEEE International Conference on Communications (ICC 2010), IEEE, Cape Town, South Africa, p 6 pages
13. Chen Z, Yang B, Chen Y, Abraham A, Grosan C, Peng L (2009) Online hybrid traffic classifier for peer-to-peer systems based on network processors. Applied Soft Computing 9(2):685 – 694
14. Ciullo D, da Rocha Neta MAG, kos Horvth, Leonardi E, Mellia M, Rossi D, Telek M, Veglia P (2010) Network Awareness of P2P Live Streaming Applications: A Measurement Study. IEEE Transactions on Multimedia 12(1):54–63
15. Crotti M, Gringoli F, Pelosato P, Salgarelli L (2006) A statistical approach to IP-level classification of network traffic. In: Proceedings of the IEEE International Conference on Communications (ICC '06), IEEE, Istambul, Turkey, vol 1, pp 170–176
16. Dainotti A, de Donato W, Pescapè A, Rossi PS (2008) Classification of network traffic via packet-level hidden markov models. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2008), IEEE, New Orleans, LA, USA, pp 1–5
17. Dusi M, Este A, Gringoli F, Salgarelli L (2010) Coarse classification of internet traffic aggregates. In: IEEE International Conference on Communications (ICC), 2010, pp 1 –6
18. Erman J, Arlitt M, Mahanti A (2006) Traffic classification using clustering algorithms. In: Proceedings of the ACM SIGCOMM Workshop on Mining Network Data (MineNet '06), ACM, New York, NY, USA, Pisa, Italy, pp 281–286
19. Erman J, Mahanti A, Arlitt M, Cohen I, Williamson C (2007) Offline/realtime traffic classification using semi-supervised learning. Elsevier Performance Evaluation 64(9-12):1194–1213
20. Este A, Gringoli F, Salgarelli L (2009) Support vector machines for TCP traffic classification. Elsevier Computer Networks 53(14):2476–2490
21. Esteves A, Inácio PRM, Pereira M, Freire MM (2011) On-Line Detection of Encrypted Traffic Generated by Mesh-Based Peer-to-Peer Live Streaming Applications: The Case of GoalBit. In: Proceedings of The 10th IEEE International Symposium on Network Computing and Applications (IEEE NCA11), Cambridge, MA USA
22. Finamore A, Mellia M, Meo M, Rossi D (2009) KISS: Stochastic packet inspection. In: Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA '09), Springer-Verlag, Berlin, Heidelberg, Aachen, Germany, Lecture Notes In Computer Science, vol 5537, pp 117–125
23. Finamore A, Meo M, Rossi D, Valenti S (2010) Kiss to Abacus: A Comparison of P2P-TV Traffic Classifiers. In: Traffic Monitoring and Analysis (TMA), Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, pp 115–126
24. Gao Z, Lu G, Gu D (2009) A novel p2p traffic identification scheme based on support vector machine fuzzy network. In: Second International Workshop on Knowledge Discovery and Data Mining, 2009. WKDD 2009., pp 909 –912
25. Gheorghe G, Cigno RL, Montresor A (2011) Security and privacy issues in p2p streaming systems: A survey
26. GoalBit Homepage (2011) `http://goalbit.sourceforge.net/`, (accessed April 10, 2011)
27. GoalBit Starter Suite Tutorial (2011) `http://sourceforge.net/apps/mediawiki/goalbit/index.php?title=Starter_Suite`, (accessed April 28, 2011)
28. Gomes JVP, Inácio PRM, Freire MM, Pereira M, Monteiro PP (2008) Analysis of peer-to-peer traffic using a behavioural method based on entropy. In: Proceedings of the 27th IEEE International Performance Computing and Communications Conference (IPCCC 2008), IEEE Computer Society Press, Los Alamitos, CA, USA, Austin, TX, USA, pp 201–208
29. Gomes JVP, Inácio PRM, Freire MM, Pereira M, Monteiro PP (2010) The nature of peer-to-peer traffic. In: Shen X, Yu H, Buford J, Akon M (eds) Handbook of Peer-to-Peer Networking, Springer US, New York, NY, USA, pp 1231–1252
30. Guo Z, Qiu Z (2008) Identification Peer-to-Peer Traffic for High Speed Networks using Packet Dampling and Application Signatures. In: IEEE (ed) Proceedings of 9th International Conference on Signal Processing, 2008, Inst. of Inf. Sci., Beijing Jiaotong Univ., Beijing, pp 2013 – 2019
31. Hu Y, Chiu DM, Lui JCS (2009) Profiling and identification of p2p traffic. Comput Netw 53:849–863
32. Iliofotou M, Kim HC, Faloutsos M, Mitzenmacher M, Pappu P, Varghese G (2009) Graph-based P2P traffic classification at the Internet backbone. In: Proceedings of the 28th IEEE Conference on Computer Communications Workshops (INFOCOM '09), IEEE Press, Piscataway, NJ, USA, Rio de Janeiro, Brazil, pp 37–42
33. Jordan S (2009) Some traffic management practices are unreasonable. In: Proceedings of the 2009 Proceedings of 18th International Conference on Computer Communications and Networks, IEEE Computer Society, Washington, DC, USA, ICCCN '09, pp 1–6
34. Keralapura R, Nucci A, Chuah CN (2010) A novel self-learning architecture for p2p traffic classification in high speed networks. Computer Networks 54(7):1055–1068
35. L7-filter Homepage (2011) `http://l7-filter.sourceforge.net`, (accessed April 10, 2011)
36. L7-Netpdl Classifier Homepage (2011) `http://netgroup.polito.it/research-projects/l7-traffic-classification`, (accessed April 10, 2011)
37. Leavitt N (2010) Network-Usage Changes Push Internet Traffic to the Edge. Computer 43:13–15
38. Lee S, Kim H, Barman D, Lee S, Kim Ck, Kwon T, Choi Y (????) Netramark: a network traffic classification benchmark. SIGCOMM Comput Commun Rev 41:22–30

39. Leonardi E, Mellia M, Horvath A, Muscariello L, Niccolini S, Rossi D (2008) Building a cooperative P2P-TV application over a wise network: the approach of the European FP-7 strep NAPA-WINE [very large projects]. Communications Magazine, IEEE 46(4):20–22
40. Lin YD, Lu CN, Lai YC, Peng WH, Lin PC (2009) Application classification using packet size distribution and port association. Journal of Network and Computer Applications 32(5):1023–1030, next Generation Content Networks
41. Liu F, Li Z, Nie Q (2009) A New Method of P2P Traffic Identification Based on Support Vector Machine at the Host Level. In: Information Technology and Computer Science, 2009. ITCS 2009. International Conference on, vol 2, pp 579–582
42. Liu H, Feng W, Huang Y, Li X (2007) A peer-to-peer traffic identification method using machine learning. In: International Conference on Networking, Architecture, and Storage, 2007. NAS 2007., pp 155 –160
43. Liu Y, Guo Y, Liang C (2008) A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications 1(1):18–28
44. Livestation Homepage (2011) `http://www.livestation.com/`, (accessed April 28, 2011)
45. Moore AW, Zuev D (2005) Internet traffic classification using bayesian analysis techniques. ACM SIGMETRICS Performance Evaluation Review 33(1):50–60
46. NetBee Library Homepage (2011) `http://www.nbee.org`, (accessed April 10, 2011)
47. Nguyen T, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys & Tutorials 10(4):56–76
48. Nguyen TTT, Armitage G (2008) Clustering to assist supervised machine learning for real-time IP traffic classification. In: Proceedings of the IEEE International Conference on Communications (ICC '08), IEEE, Beijing, China, pp 5857–5862
49. OpenDPI Homepage (2011) `http://www.opendpi.org/`, (accessed April 10, 2011)
50. Palmieri F, Fiore U (2009) A nonlinear, recurrence-based approach to traffic classification. Elsevier Computer Networks 53(6):761–773
51. Raahemi B, Kouznetsov A, Hayajneh A, Rabinovitch P (2008) Classification of peer-to-peer traffic using incremental neural networks (fuzzy ARTMAP). In: Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 2008), IEEE, Niagara Falls, ON, Canada, pp 719–724
52. Shrivastav A, Tiwari A (2010) Network Traffic Classification Using Semi-Supervised Approach. International Conference on Machine Learning and Computing 0:345–349
53. Silverston T, Jakab L, Cabellos-Aparicio A, Fourmaux O, Salamatian K, Cho K (2011) Large-Scale Measurement Experiments of P2P-TV Systems: Insights on Fairness and Locality. Elsevier Signal Processing: Image Communication
54. SNORT Homepage (2011) `http://www.snort.org/`, (accessed April 10, 2011)
55. Su X, Dhaliwal S (2010) Incentive mechanisms in p2p media streaming systems. IEEE Internet Computing 14(5):74 –81
56. Tcpdump Homepage (2011) `http://www.tcpdump.org/`, (accessed April 10, 2011)
57. The Napa-Wine Application Showroom (2011) `http://napa-wine.eu/cgi-bin/twiki/view/Public/NapaWineShowRoom`, (Accessed April 29, 2011)
58. Tudou (2011) `http://www.tudou.com/`, (Accessed April 29, 2011)
59. TVUNetworks Homepage (2011) `http://www.tvunetworks.com/`, (accessed April 10, 2011)
60. ao V Gomes J, Inácio PRM, Pereira M, Freire MM, Monteiro PP (2011) Detection and Classification of Peer-to-Peer Traffic: A Survey. `http://floyd.di.ubi.pt/nmcg/pdf/2011-P2PSurvey.pdf`, (Accessed April 10, 2011)
61. Wang P, Guan X, Qin T (2009) P2p traffic identification based on the signatures of key packets. In: IEEE 14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, 2009. CAMAD '09., pp 1 –5
62. Wireshark Homepage (2011) `http://www.wireshark.org/`, (accessed April 28, 2011)
63. Xu K, Zhang M, Ye M, Chiu DM, Wu J (2010) Identify P2P traffic by inspecting data transfer behavior. Computer Communications 33(10):1141 – 1150
64. Y Gu HZYZJLGCYLDMLX N Zong (2011) Survey of P2P Streaming Applications. IETF RFC draft, draft-ietf-ppsp-survey-01, March 2011
65. Youtube (2011) `http://www.youtube.com/`, (Accessed April 29, 2011)
66. Yuan R, Li Z, Guan X, Xu L (2010) An svm-based machine learning method for accurate internet traffic classification. Information Systems Frontiers 12:149–156
67. Zander S, Nguyen T, Armitage G (2005) Automated traffic classification and application identification using machine learning. In: Proceedings of the IEEE Conference on Local Computer Networks (LCN 2005), IEEE, Sydney, Australia, pp 250–257
68. Zong N, Zhang Y, Pascual V, Williams C, Xiao L (2010) P2P Streaming Protocol (PPSP) Requirements. (2010). `http://tools.ietf.org/html/draft-zong-ppsp-reqs-04`, (Internet Draft. Accessed April 10, 2011.)

# Chapter 4

# Conclusions and Future Work

This chapter will be organized in two sections. The first section relates the conclusions reached after all the research work on peer-to-peer traffic detection relative to three popular peer-to-peer media streaming application: TVUPlayer, Livestation, and GoalBit. To conclude, the last section suggests future research directions that may result from this research work.

## 4.1 Main Conclusions

In this dissertation, a method for detecting and classifying peer-to-peer media streaming traffic generated by three of the most popular applications is proposed. For this research work, a test-bed was implemented, with conditions that simulate a real case scenario, i.e., all kind of traffic was generated and not only traffic exclusively from the three peer-to-peer applications. From the captured traffic and posterior analysis, several Snort rules were successfully created.

For TVUPlayer, rules were created after the careful analysis of TCP and UDP traffic generated by the application itself, in which we discovered patterns that repeat themselves over the packets. Not all the packets carried the same patterns, and it was due to the different patterns that several rules were created. After adding the set of rules to Snort, a validation process was made: first, we filtered traffic by the application and ran the set of rules to check completeness, after that we ran the set of rules against the full trace file (that contained all kinds of traffic) to check if any false positive was triggered. As hoped, no false positive was raised, but the completeness of the rules was not total.

As for Livestation, soon after starting to capture traffic and analysing it, we came to know that Livestation was no longer working as a peer-to-peer application. Said that, not much more was made to detect or classify its traffic. Nevertheless two rules were created that successfully detect Livestation traffic. These two rules, 1000511 and 1000512 respectively, were built after analysing TCP and HTTP traffic generated by this application.

As for the last application used on this research work, GoalBit, we tested different configurations for streaming the media. The four tested scenarios were: tracker-based with and without encryption of the stream, and trackerless-based also with and without encryption of the stream. After analysing all the traffic captured when using these four configurations, several rules were created. From this set of rules, some detect UDP packets, and the others detect TCP packets. In the GoalBit case, we also ran the set of rules against the full trace file to check for false positives. We had some false positives alerts, due to the rules that detect UDP traffic, since UDP traffic was generated only when the trackerless-based (uses Kademlia algorithm) configuration was used, and since Kademlia protocol is also used by other programs, some false positives were raised.

For the created sets of rules, and we were talking of encrypted traffic, only a small amount of alerts were registered. To improve the accuracy of the method, a flow detection approach is proposed. With this approach, even if a small percentage of the traffic is identified, we may classify all that flow as being generated by the identified application.

## 4.2 Directions for Future Work

To conclude this dissertation, the author suggests some future research directions that may result from this work. The first suggestion is given the results obtained in the experiments, there is the possibility to implement the set of rules in the campus network and test their performance in the presence of real and more diverse network traffic, and also to check the network performance. Another suggestion would be the possibility to automatize the procedure of deriving the rules for SNORT from the traces. The production of an hybrid method, i.e., combining the method proposed in this dissertation with a behavioural technique is also left for future work. Defeating encryption used to camouflage peer-to-peer traffic is not an easy thing to achieve. The last suggestion was to build some sort of network module that breaks protocol encryption when DPI methods are used, and prevent the deterioration of network performance.