HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Networking Laboratory

Andrea Buonerba

Skype Traffic Detection and Characterization

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Helsinki, September 4, 2007

Supervisor: Professor Jorma Virtamo

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Andrea Buonerba
Name of the Thesis:	Skype Traffic Detection and Characterization
Date:	September 4, 2007 **Number of pages:** 72 + VII
Department:	Department of Electrical and Communications Engineering
Professorship:	S-38 Networking Technology
Supervisor:	Prof. Jorma Virtamo

Skype is a very popular VoIP software which has recently attracted the attention of the research community and network operators; furthermore Skype uses a proprietary signalling design and its source code is unavailable. This makes its analysis really important since the classification of IP flows becomes increasingly crucial in modern network management platforms. Traditional classification systems based on packet headers are rapidly becoming ineffective. In this work after a general analysis of Skype protocol and traffic in both time and frequency domain, a new classification method is presented. It is based on statical classification of the flow, using only three basic properties of IP packets: their size, interarrival time and order of arrival. The whole process is based on a new quantity called Protocol Fingerprint. Its aim is to express these quantities in an efficient way. An important part in the classification process is taken by a Gaussian filter that smooths the protocol fingerprints avoiding misclassifications caused by any kind of noise generated in the network. Even if this technique is at an early stage of development and requires more work, it is quite promising.

Keywords: Skype, traffic, statistical classification, fingerprint

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

My thanks go to Prof. Jorma Virtamo who has taken the time and trouble to give me advices, comments and remarks and to alert me on errors in the text.

I am indebted to my many student colleagues for providing a stimulating and fun environment in which to learn and grow. I am especially grateful to all "Leppävaara group"; without them my Finnish experience would not have been so great. I expecially thank Jose Victor because with his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make work fun for me. Throughout my thesis-writing period, he provided encouragement, wise advice, good teaching, good company, and lots of good ideas.

I cannot end without thanking my family, on whose constant encouragement and love I have relied throughout my time at the University.

Otaniemi, September 4, 2007

Andrea Buonerba

Contents

List of Figures

Chapter 1

Introduction

Voice over IP (VoIP) has been very popular in the last few years due to its low cost, high quality, and ease of use. These years have witnessed VoIP telephony gaining a tremendous popularity, as also testified by the increasing number of operators that are offering VoIP-based phone services to residential users. Skype is beyond doubt the most amazing example of this new phenomenon: developed by the creators of KaZaa, it recently reached over 10 millions of users, becoming so popular that people indicate Skype IDs in their visiting cards. A number of reasons could explain this success. First, today Internet (capacity, responsiveness, robustness) makes it possible and easy to provide new and demanding services, like real-time applications; second, the user's attitude towards technology has improved in the last years. People are willing to accept a good service for free, even though service continuity and quality may not be guaranteed. Everybody likes acceding from the same terminal and the same application enviroment to a plenty of different communication facilities. Skype is an extremely good piece of software, carefully engineered, user firendly and efficient at the same time. This very popular VoIP software has attracted the attention of the research community and network operators.

It is a famous peer-to-peer application since its launch in 2003. However, none of Skype's algorithms or its protocol specifications are available for a general discussion or inspection, which impedes evaluation from a security perspective. This point is critical, since Skype uses strong encryption mechanisms that make it really difficult to even glimpse its presence from a traffic aggregate. The importance of Skype traffic identification (besides being instrumental to traffic analysis and characterization for network design and provisioning) is clear when considering the interest of network operators, ranging from traffic and performance monitoring, to design or tariff policies and traffic differentiation strategy. Skype is having a serious impact on carrier's revenue streams, but perhaps the most important aspect is that the nature of Skype traffic is raising security concerns, especially for large enterprise networks. Skype uses a unique peer-to-peer technology, as said, making tedious for operators to identify, classify and manage the associated traffic.

It is important to underline that despite the interest and the work recently done by the research community, reliable identification of Skype traffic remains a challenging task, given that the software is propietary and the traffic is obfuscated. In general, effective traffic identification, detection and classification requires three key elements:

- Accuracy: the technique should have low false positive (identifying other protocols as targeted protocol X).
- Scalability: the technique must be able to process large traffic volumes in the order of several thousands to several million connections at a time, with good accuracy, without being computationally too expensive.
- Robustness: traffic measurement in the middle of the network has to deal with the effects of asymmetric routing (two directions of a connection follow different paths), packet losses and reordering.

Of course there are tradeoffs in terms of level of accuracy, scalability and robustness that can be achieved relative to the detection of any given protocol or service. One current classification practice consists of TCP/UDP port number application identification using known TCP/UDP port numbers to identify traffic flows. This method is highly scalable since only the TCP/UDP port numbers must be recorded to identify a particular application. It is also highly robust since a single packet is sufficient to make a succesful identification. Unfortunately port number-based identification is increasingly inaccurate primarily due to the fact that P2P networks (such as that one of Skype) tend to intentionally disguise their generated traffic in order to circumvent filtering firewalls. The majority of P2P networks now operate on top of custom-designed propietary protocols and their clients can easily operate on any port number even HTTP's port 80, making port based detection schemes incapable of accurate and robust classification of Internet protocols. To overcome the issues with port-based detection, a new technique that processes packet payloads for patterns signatures that univocally identify any given protocol can be used. One challenge facing payload-signature techniques on telecom networks is the high speed at which such pattern matching algorithms should operate. It is therefore critical to design algorithms that can efficiently perform pattern matching while simultaneously dealing with memory and CPU limitations.

Another challenge is the lack of knowledge about reliable protocol specifications. This is due to the development history and to the propietary nature of many protocols. For application detection and classification to be reliable and accurate, it is important to identify signatures that span all the variants (or at least the dominant used ones). However, it is common to see new application using 128-bit or 256-bit bit encryption techniques to defend the privacy of the information exchanged between their users. As a consequence, the payload-signature method fails when traffic is encrypted, because the signatures in the packet payload are scrambled by the encryption. Skype offers a combination of challenges that make it difficult to detect with scalable, reliable and accurate algorithms:

- The Skype agent does not run on any standard source port. Skype randomly selects a soure port for the agent to run on, then communicates via either TCP or UDP, or both. The choice of the protocol that Skype uses depends on whether the agent is behind a proxy/NAT or has a public IP addres. The destination IP addresses are not the same every time Skype runs, and the destination port numbers are also not standard.

- All communication via Skype is encrypted. This means that phone numbers called (SkypeOut) or other data are also encrypted. In many cases, there is no direct communication between end users in Skype. All communication passes through intermediate nodes, and these nodes may be different for every call.

- Skype is a peer-to-peer protocol, which means that the peers (IP addresses) to which a Skype agent connects are many and the network is very dynamic, so these peers (and thus their IP addresses) keep changing.

- Skype provides voice, chat, file transfer and viceo services. It appears that all of these services are passed together, making it difficult to separate out voice, from chat, from video, etc.

To accurately detect and classify these unfriendly applications, it is necessary to provide a systematic methodology that overcomes the lack of well-known port numbers or user payload signatures. Instead, any new methodology should analyze flow connections at the transport layer (Layer 4) to extract and profile key features from the packet streams processed. Such a method could be referred to as “classification in the dark”. This kind of analysis can be divided in two different analysis applications: protocol analysis based on packet inspection and based on traffic behavior. Packet inspection analysis, as already said, relies on monitoring traffic passing through the network and inspecting the data payload of the packets according to some previously defined P2P application signatures. These signatures consist of regular expressions matching on the application data, in order to determine whether a special P2P application is being used. Because protocol analysis focuses on the packet payload and raises alerts only on a definite match, any client-side tricks that use non-default or dynamic ports to avoid detection by P2P application will fail. Using this approach, the result is normally more accurate and believable, but it still has some shortcomings:

- P2P applications are evolving continuously, and therefore signatures can change. Static signature based matching requires new signatures to be effective when these occur.
- With more and more P2P identification and control products on the market, P2P developers tend to tunnel around any controls placed in their way. They could easily achieve this by encrypting the traffic, such as by using SSL, making protocol analysis much more difficult.

Any TCP and UDP streams not classified by the Payload-signature application should be forwarded to the Behavioral application. Streams of packets with encrypted payloads, emerging P2P protocols for which a signature is not available, or multimedia applications using proprietary technologies (such as VoIP, Video, Gaming, File Transfer, Chat, etc) fall into this family. The behavioural-signature application profiles the behaviour of hosts at different levels by exploring its social level (hosts that it communicates with), its functional level (server vs client vs peer-node), its application level (transport layer interactions between particular hosts on specific ports) and specific dynamics, with the intent to identify the application of origin.

A different approach is based on a stochastic characterization of Skype traffic in terms of the packet arrival rate and packet length, which are used as features of a decision process based on Naive bayesian Classifiers.

1.1 Goals of this Work

With the continuous increase of Internet bandwidth and rapid advancement of P2P applications, VoIP technology has become a communication alternative for many Internet users. The popular VoIP service Skype uses a propietary signalling and media protocol for voice calling, instant messaging, audio conferencing and file tranfers. Additionally all traffic is encrypted. Thus, it is currently not known what information is exactly transported within the application. This is the reason why a more detailed knowledge about Skype traffic is necessary, since Skype also impedes protocol analysis for potential security holes. In this work traffic is only analysed from a network point of view.

1.2 Thesis Structure

The thesis is organized as follows. Chapter 2 describes key components of Skype software and Skype network. Chapter 3 reviews all the previous work about Skype. In chapter 4 the results of the protocol analysis are shown. These results concern issues both in time and frequency domain. In chapter 5 the concepts of Self-Similarity and Long-range Dependence are introduced. Their applicability in estimating the Hurst Parameter of Internet traffic and their applicability to Skype traffic are tested. In chapter 6 different methods for traffic classification are analysed. Finally in chapter 7 a statistical method for traffic classification is tested. In this last chapter it is explained how it is possible to classify traffic using statistical fingerprints and aspects related the applicability of this technique to Skype traffic are discussed. A summary about the work required to improve the classification technique is also presented. Chapter 8 concludes the work.

Chapter 2

Architecture and Principles of Operation of Skype

As already written, Skype has gained a tremendous popularity during the last few years. Hudson Barton [1] made a study on worlwide Skype usage and from it, by looking at the "concurrent on line" statistic, it is possible to discern trends and demographics in its growth pattern. Skype will keep growing in 2007 and 2008, adding more than 12 million new real users (to its current 21.3 million). This is a faster pace than 2006, demostrating that growth remains exponential. Of course there are regional differentiation, in Asia/Pacific the growth has lagged other regions but may start to catch up the rest of the world. The problem could have been the lack of physical infrastructure and political bans (such as in China, where VoIP is banned by the government). There are also enterprise and consumer differentiation but anyway regarding Skype profitability, price increases will not hinder the growth and Skype will struggle less than its competitors in 2007 and 2008: It is becoming a sort of social, collaborative, and borderless network, supporting concepts of groupware and Web 2.0.

2.1 Network Structure

The first, deep analysis of Skype was done in 2004 [2]. Since then many Skype versions have been released, but all of them regarded the terminal layout and features, the core network remained of course the same. Skype is a peer-to-peer system based on clients developed by KaZaa, that gives the user the possibility of instant messaging, calling and video-calling other users belonging to Skype network. There is also the option of video conferencing. In the end, the features offered make it quite similar to the MSN and Yahoo IM applications, however the protocol and the techniques used underlying are different. Like almost all other different peer-to-peer networks, also Skype is formed by different nodes (machine on which Skype client is running) that communicate with each others. There are two different Skype node types: an ordinary node and a supernode. A super node is an ordinary node with a public address and a sufficient CPU, memory and network bandwidth. This approach clearly favors the overall availability of the system. It can been shown [3] that the population of the supernodes selected by Skype, apparently based on reachability and spare bandwidth, tends to be relatively stable. Skype, therefore, represents an interesting point in the P2P design space where heterogeneity is leveraged to control churn, not just to cope with it.

Typically, supernodes mantain an overlay network among themselves, while ordinary nodes pick one (or a small number of) supernodes to associate with; supernodes also function as ordinary nodes. Ordinary nodes issue queries through the supernode(s) they are associated with. An ordinary node must connect to a supernode and must register with the Skype Login Server for a succesful login. Skype Login Server is not a node itself but it is fundamental for the functioning of the network. Usernames and passwords are stored there, and it checks that these ones are unique. User authentication at login is also done there. This is the only central server present in the network. But online and offline user information is always spread around the network, making a research faster.

Another important aspect of Skype is how it deals with NAT and firewall traversals. It is believed that Skype uses a variant of STUN protocol in order to

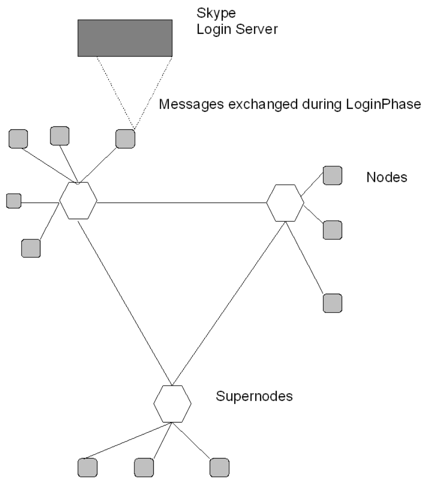

Figure 2.1: *Skype Network. There are three main entities : supernode, ordinary nodes, and the login server*

determine in which kind of network it is used. It is also believed [2] that there is no server containing this info, because no exchange of this kind of information has been found in any experiment.

The existence of supernodes is important in the login phase; every ordinary node should keep an updated list of reachable nodes. This list is called "host cache" and it contains IP address and port numbers of super nodes. In [4] a good ports analysis is done. Skype uses for communacation both UDP and TCP connections, also together. It has the ability of using also TCP ports 443 (HTTPS) and 80 (HTTP), in this order. Regarding UDP traffic, it uses arbitrarily set port. The user can change the used port for UDP (and incoming TCP) traffic. UDP is the preferred type of connection, but it is not mandatory. It can be shown the a client is able to login even with only TCP traffic allowed.

2.2 Codecs

So far, VoIP has predominantly been hyped as a mean of cost saving, and this is a legitimate pitch. But VoIP also lays the groundwork for a revolution in the quality of voice we communicate with which was somehow blocked by the fixed rate nature of PSTN. Telephones have remained unchanged for so long that most people have no idea what limitations they have lived with. This has begun to change with the recent rise of wideband technologies for the masses such as Skype. Now people are asking everywhere, “why does Skype sound good?” The answer is that Skype is capturing double the spectrum of voice frequencies that are captured by standard telephones.

Standard PSTN, and the overwhelming majority of VoIP telecom codecs, capture at 8 kHz. Skype can capture at 16 kHz. The added fidelity in the high frequencies adds substantially to the realism of the reproduction, makes it possible to distinguish between otherwise difficult phonemes of “s” “f” “c” “e” “d” and so on, and in an unmeasurable way, creates much more a sense of “being there” with the other person. The comparison is analagous to that of AM and FM broadcast fidelity. The bitrate this data is transmitted is a separate topic from the sample rate of the initial capture. A great codec may minimize the amount of data needed to be transmitted per second (bitrate), but it can never capture it any better than the original sampling (sample rate). This is the “buy low, sell high” of communications. We want the lowest bitrate possible for the highest sample rate possible... low cost, high quality. There are many codecs available for this compression / decompression process, but they are almost all tuned for an 8 kHz process. The simple reason for this is that regardless of how high fidelity a VoIP signal is, as soon as it passes into the PSTN, anything beyond 8 kHz is discarded. Furthermore, a 16 kHz signal that is downsampled to 8 kHz typically doesn’t sound as good as one that started at 8 kHz from the start.

Increasing the bandwidth of sound signals from the telephone bandwidth of

200-3400 Hz to the wider bandwidth of 50-7000 Hz results in increased intelligibility and naturalness of speech and gives a feeling of transparent communication. The emerging end-to-end digital communication systems enable the use of wideband speech coding in a wide area of applications. Recognizing the need of high quality wideband speech codecs, several standardization activities have been recently conducted, resulting in the selection of a new wideband speech codec, AMR-WB, at bit rates from 6.6 to 23.85 kbit/s by both 3GPP and ITU-T. The adoption of AMR-WB by the two bodies is of significant importance since for the first time the same codec is adopted for wireless as well as wireline services. This will eliminate the need for transcoding, and ease the implementation of wideband voice applications and services across a wide range of communication systems and platforms.

Wideband VoIP only works when VoIP is sent from end to end. With the prevalence of existing PSTN infrastructure in our lives, pure end-to-end VoIP links still make up only a small percentage of communications traffic. It seems to be prevalent folklore that Skype uses what is commonly referred to as “Wideband iLBC”. To be more accurate, Skype has licensed the VoiceEngine product, which is a comprehensive solution that includes all of these codecs, as well as a jitter buffer, error concealment, and echo cancellation technology. The GIPS suite of codecs includes, among others, iLBC which is a fixed rate codec. But, as indicated on the Skype website [5], Skype varies its bit rate which could not happen with iLBC. But in the suite there is also the iSAC codec. According to a GIPS engineer, The iLBC [6] and iSAC [7] algorithms are pretty much unrelated. iLBC is a narrowband fixed rate codec operating at 13.3 kbps or 15.2 kbps. The algorithm is available from IETF (RFC 3951 and 3952). ISAC is an unrelated wideband variable rate codec, which can adapt its operating rate between 10 kbps and 32 kbps. The codec is proprietary and the algorithm is unavailable. So, though at times Skype may indeed utilize iLBC, it must at others be using iSAC. Since

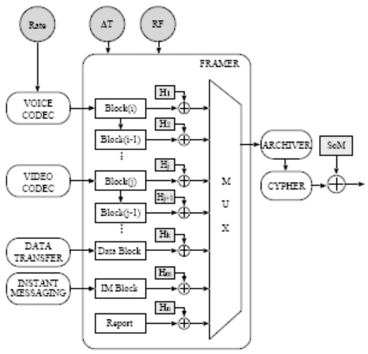

Figure 2.2: *Schematic diagram representing the Skype message building process*

Skype has licensed both (as well as iPCM-wb, another 16 kHz codec).

2.3 Encryption

In [5] it can be read that Skype uses AES (Advanced Encryption Standard), the same used by the U.S. Government to protect sensitive information. Skype uses 256-bit encryption, that means there are 1.1 x 10^{77} possible keys for encrypting data in a call or in a message. Skype uses from 1536 to 2048 bit RSA to negotiate symmetric AES keys. User public keys are of course certified by Skype central server at login.

2.4 Framing and Scheduling

Real-time voice communication requires frames to be encapsulated in IP packets and sent periodically to receivers. A short packet period shortens the mouth-to-ear delay, but may incur higher loss rates and therefore lower quality.

Considering the voice source, the voice encoder used during a call, for example the "Voice Coder", creates blocks of encoded voice. The framer is then addressed to create Skype frames, by multiplexing into a single frame one or more blocks (in order, for example, to cope with the potential link loss of the previous frame, or to generate the message rate). Possibly, video, audio, chat, report data is multiplexed together, adding only headers that will specify it. Once created the frame, it is compressed by the "Archiver" and encrypted by the "Cypher". Sometimes and additional non-chiphered header may be present, usually it is called "Start of Message", (SoM). In this process there are important parameters, like the Rate (the bit-rate used by the Codec), the Interval-Time (meaning the framing time) and the Redundancy Factor, meaning the number of past block that need to be included in the current frame. Of course these parameters change during the connection, following the network connection state.

As written in [8] Skype sends voice packets by two rates. In fact it uses voice activation detection (VAD), it detects an active period while the caller is speaking and packs frames of 30 ms iLBC (for example) of 50 bytes each into an UDP packet and transmits the packets every 60 ms. During the silence period, it trasmits 25 bytes in each UDP packet every 100 ms. No silence suppression is supported in Skype. It has been observed [2] that packets flow also during silence period. Sending these packets is useful for two reasons. First, it helps mantaining UDP bindings at NAT, and second, they are useful for playing some background noise at the peer. Even if a TCP connection is used, these packets are necessary for avoiding the connection time out, and the congestion window size drop, which usually would take few RTTs to reach the maximum level again.

Jitter buffers are used to smooth out variations in packet arrivals. For real-time applications, the size of the jitter buffers must be chosen properly in order to balance between the number of arrivals considered late for play-back and the mouth-to-ear delay. This size could, of course, adaptively change either during silence periods, or not. Skype uses a simple FIFO approach to handle arrivals and play-outs. It smoothly plays packets arriving within 60 ms with respect to the expected arrival time. It also buffers different packets arriving ahead of time and stores them for playback later. If a packet arrives out of order, it would be processed following the FIFO rule. The fixed jitter length in Skype is also

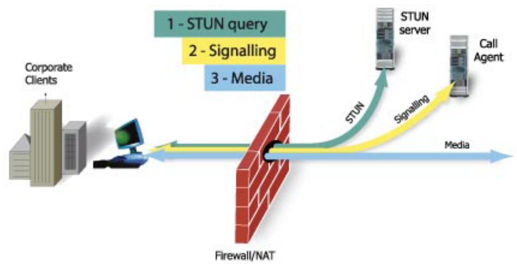

Figure 2.3: *Simple Traversal of UDP Through Network Address Translators (STUN)*

inadequate, as there is a moderate amount of jitter over 60 ms with respect to the mean delay for some connection. As we know, loss rate and behaviour in an Internet path is characterized by being asymmetric, non-stationary and variable. It follows that a trade-off between the network overhead and the accuracy of the feed-back must be found. Skype sends/receives feedback every 2 seconds by encapsulating 4 or 8 bytes of information.

2.5 NAT and Firewall Determination

During some experiments, confirmed also in [2], it has been noticed that Skype is able to recognize in what kind of network it is situated (e.g. behind what kind of NAT and what kind of firewall) during the login phase. In fact if the network situation is changed after the login (for example with a new entry in the firewall rules list) Skype is not anymore able to connect to any other peer, soon after the change. It is possible (but not proven, since messages are encrypted) that Skype refreshes this kind of information, since after a while Skype is able to connect again. One possible way to get this information is to exchange data with the Supernode and possibly with more ordinary nodes (after it has made a TCP connection with the supernode) while logging in, for example using a version of STUN protocol (Fig. 2.3); after determined, it stores the info.

2.6 Call Placement

In this work only calls between peers already present in the buddy-list have been analyzed and studied. Of course, one call to a peer that is not present in a buddy-list is just like a normal call, preceded by the "user search" mechanism. It is important to note that call signalling is always exchanged in a TCP connection. If between the caller and callee there is neither a NAT nor a firewall, the caller establishes a direct TCP connection to the callee. Thereafter the caller sends messages, inside a challenge-respone mechanism, to all the Skype nodes he knows. In fact, Skype keeps track of online nodes in the Skype network so that it can connect to one of them if its Supernode becomes unavailable. This table of online nodes is called "Alternate Node Table". After signalling has been exchanged, the peers start the media transfer over a direct UDP flow.

If one of the two peers is behind a NAT, the TCP connection is placed with a third node, which will forward the packets to the callee through another TCP connection. But if no firewall is present, the media will also flow through this node, but over UDP. If also a UDP-restricted firewall is present, the TCP connections (caller-third node, third node-callee) will carry also the media flow.

Chapter 3

Overview of Previous Research Work

As the use of Skype spreads and increases in importance, there is a need for tools and techniques to analyze Skype traffic; it is also becoming a problem for network administrators, since Skype´s algorithms are not available for public inspection, which impedes evaluation from a security perspective. Skype operations from network point of view and Skype protocol reversing [4] analysis is one of the various studies done over Skype.

3.1 Protocol Reversing

From these analyses it has been developed traffic signatures that allow a third party monitoring entity to detect the usage of the Skype application. A signature needs to meet some criteria in order to be useful.

- It should be as compact as possible. A complex signature complicates the monitoring unit which results in decreased performance with multiple concurrent traffic patterns to monitor.
- The signature should consider all possible cases how traffic could be injected in the network; otherwise Skype sessions might evade detection.

- The number of detected false positives, i.e. the detection of a Skype session if in reality no Skype session was established, should optimally be zero.

By analyzing UDP packet flows it has been possible to identify some patterns. For example a sort of ID; a 16-bits long section that uniquely identifies the message. It seems to be randomly selected, put in the sender query and copied in the receiver reply. Another part of the message is used for underlining the payload type. It is a 5-bits long field obfuscated into a byte (byte 3). Three random bits can be easily removed by considering a 0x8f bitmask. The rest of the message is the frame, that contains a multiplexed sequence of information (that has been previously ciphered) and voice blocks. The ID and payload identification sequence are part of the so-called SoM header (Start of message). This part of the message can only be obfuscated and not ciphered. In fact Skype uses both UDP and TCP as transport protocol. While TCP implements a connection-oriented reliable transport protocol and the application is guaranteed to receive all data segments, and furthermore in-same-sequence they have been inserted in the network. UDP no longer guarantees all in-sequence data delivery. Therefore Skype receiver must extract from the application layer header some additional information to detect and eventually deal with some losses; as said, it comes that the information cannot be protected by means of a stream cypher, but can only be obfuscated by means of some function based only on a single packet payload.

3.2 Skype Traffic

Voice over IP has become more and more popular in the wired network, but since all the operators are starting offering large data rates in UMTS, it can also make it a suitable environment. Anyway, the success of any new application/service is strictly connected with the user satisfaction, and in the Skype-specific case with voice-quality that is perceived. It is interesting to analyze the achievable and the actual quality of IP-based telephony using Skype [9].The result shows if Skype is able to keep pace with the existing mobile telephony systems and how it reacts to different network characteristics.

The work comprises the “Perceptual Evaluation of Speech Quality” (PESQ) in order to evaluate voice quality, packet loss, inter-packet delay and the throughput. It has been emulated the UMTS rate control mechanisms by restricting the link data rate with a traffic-shaping router. In this case, Skype does not react to packet loss, for example increasing the throughput. The measurement in a public UMTS environment revealed that capacity offered by UMTS is sufficient to make mobile VoIP call possible. Anyway, due to network jitter and the use of different codecs used by Skype, sometimes the PESQ were better in the simulation environment than in the real one. The important aspect is that, Skype, after influencing and changing the fixed telephony world, would be suitable also for a big expansion in the mobile one. It is already possible to buy handsets that are Skype-software capable, and when in the future they will become more and more common, and the network data rate higher, an application like Skype will start to “scare” also mobile operators.

3.3 Skype Relayed Traffic

Networked applications are using network resources in increasingly ingenious ways in order to achieve high scalability and circumvent security limitations. One example is the emergence of relay nodes -application running the same application. As already written, the use of relays allows two nodes that could not otherwise communicate (due to firewall or NAT) to do so, and can improve the quality of the communication between two nodes by avoiding congested or faulty paths. Of course this design has some disadvantages from the perspective of users and network operators. A node chosen to be and to act as a relay must, of course, bear more traffic, his own and the relaying one. The cost is evident to users in the form of slower communication (in fact, part of the bandwidth would be used by relayed traffic) and financial cost (if the payment system is based upon the quantity of traffic exchanged). Network operators and small ISPs see only drawbacks, as relayed traffic increases the amount of traffic exchanged in the network. It easy to check that a Skype application could relay more than 1 Gigabyte of voice traffic in 15 days. These are all valid points for explaining a characterization of the nature of relayed traffic and a detection of its presence in the network. Network operators would like to know if and which traffic is being relayed through hosts that belong to their network.

In [10] relayed traffic created by Skype is characterized, and a methodology for detecting it is proposed. Several metrics have been proposed to characterize the nature of multimedia relayed traffic. Detection is performed by setting up thresholds for the metrics taken into consideration. By tuning the thresholds, the desired balance between true positives and true negatives can be obtained. After this a detection methodology for a large aggregate of traffic traces can be applied. Usually a good classification method relied over port number, but as explained, this information is not useful for Skype, since the application is more flexible and diverse than older applications. In [10] neither this nor protocol-specific information is taken into consideration. Only the fact that Skype trasmits multimedia traffic is taken into consideration. Another problem is also how to identify relayed traffic inside the aggregate one.

In this paper a flow is defined as a sequence of packets with the same 5-tuple (source and destination IP addresses, source and destination port numbers, protocol number). But this “definition” needs some additions, as for example the direction of the flow. Another requirement is a maximum amount of time elapsed between two adjacent packets in a flow. It is also important to add the notion of burst of packets, that can characterize relayed traffic. A burst of packets is a contiguous piece of a flow and is defined as a sequence of packets that has a minimum average data rate (for example 10 kbps) and a minimum duration (for example at least 30 seconds). For detecting the start and the end of a burst it is possible to use many algorithms, such as EWMA (Exponential Weighted Moving Average):

$$A_i = (1 - \alpha)A_{i-1} + \alpha I_i \tag{3.1}$$

where I_i represents the average data rate of the *i-th* second of the flow and A_0 and I_0 are set to zero. Usually it is updated each second and when A_i goes beyond a certain threshold a flow is considered started. If A_i goes below that threshold the flow is considered finished. Different metrics are used:

- the difference between the start times of bursts i and j
- the difference between the end times of bursts i and j
- the ratio between the number of bytes carried by bursts i and j
- the maximum cross correlation between time series corresponding to the total number of bytes sent

These metrics should be used together in order to detect relayed traffic; but it is shown that the gain is not much higher than using them separately. This occurs because somehow the metrics are correlated. It should be noted that these metrics do not take in consideration Skype protocol, in fact in some experiments traffic generated by applications other than Skype are misclassified as being Skype-relayed traffic.

Chapter 4

Time and Spectral analysis

In this study Skype traffic traces have been analyzed. These traces have been collected inside the faculty's campus network at the Politecnico di Milano. The networking infrastructure, which consists of thousands of workstations equipped with a variety of operating systems, is composed of several 1000Based-TX layer-2 segments routed through Linux-based processor boxes. Connectivity is provided to end users by either 100BaseT links or 802.11b/g Access Points.

All experiments were performed using Skype version 3.1.0.152 on Windows machines and Skype version 1.3.0.53_API version on Linux machines. Each machine had a 10/100 Mb/s Ethernet card. The portion of the network used as testbed was composed of six machines with Linux as Operating System and one machine with Windows as Operating System, as shown in figure 4.1.

Different experiments were performed under three different network setups. In the first setup, both Skype users were on machines that had a direct connection. In the second setup, one firewall was enabled between the two hosts, caller and callee, forcing Skype to route the call through a third node, acting as a relay-node, in the external network. In the third setup all UDP traffic has been blocked, so that a TCP connection was used for transmitting the media between caller and callee. During the tests different kinds of media were sent through the System:

1) Music

2) Voice

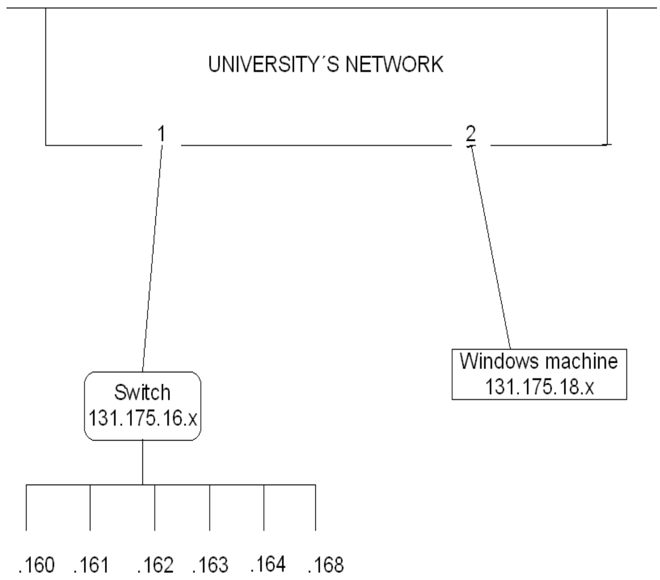

Figure 4.1: *Part of the Network used for the experiments*

3) Silence

It is important to underline that no differences have been found in the analyzed traces depending on the media that was delivered. Both analyses in time domain and frequency domain are independent on the delivered media. Even the Power Spectral Densities do not reflect any changes. This can be explained by the fact that in packets building process, after the codec, there is a multiplexer, an archiver and a cypher, as in figure 2.4. It means that the final traffic shape and the characteristics are not only given by the codec, but by all the different elements concurring in building the packets. Another point is that there are no differences between a direct call and a relayed one. The important factor is the transport layer used, UDP or TCP.

4.1 Time Domain: UDP calls

The first test that was performed consisted in a direct call between caller and callee, using UDP as transport layer. The trace was collected and analyzed

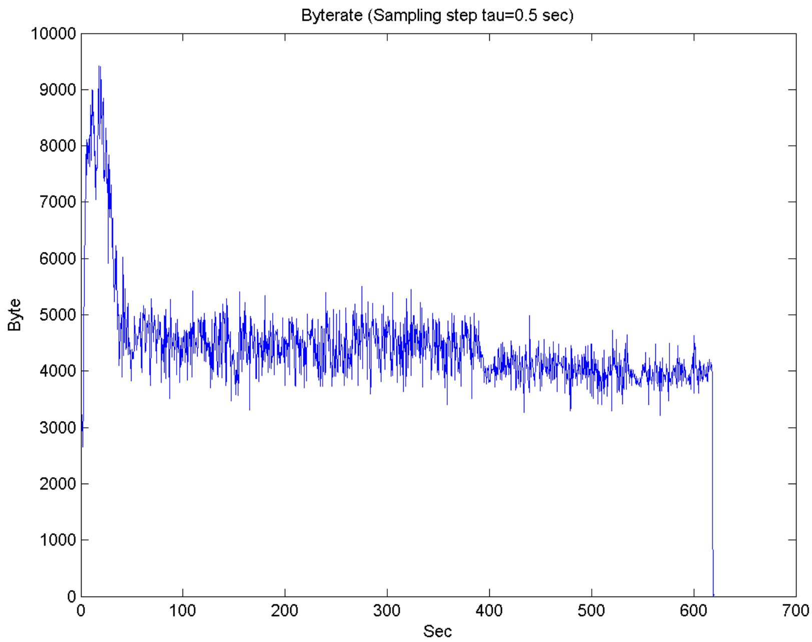

Figure 4.2: *Byterate of a direct call, UDP flow, sampling rate = 0.5 s*

through Tcpdump, the information required (timestamps and packet) gathered through "bash" commands on a Linux machine. The first thing that was checked was the Byterate. It was computed with a sampling interval of 0.5 sec; as it can be seen from figure 4.2 at the beginning of the call the byterate is almost twice than in the rest of the call.

It can be noticed from figure 4.3 and 4.4 that this is caused not by a higher packetrate (that keeps being constant) but by greater packets lengths. This, can be explained either by a sort of "training" period of the codec, or, more realistically, by the switch to a different codec, or framing policy more suitable to the network situation. This "transition" perdiod lasts at least 30 seconds each call, so quite a long time, during which anyway, caller and callee do not experience any kind of problems. As it can be seen from figure 4.5 the majority of the packets have a length between 100 and 150 bytes. It is also interesting to underline the presence of many packets that are 36 bytes-long, sent almost each 20 seconds

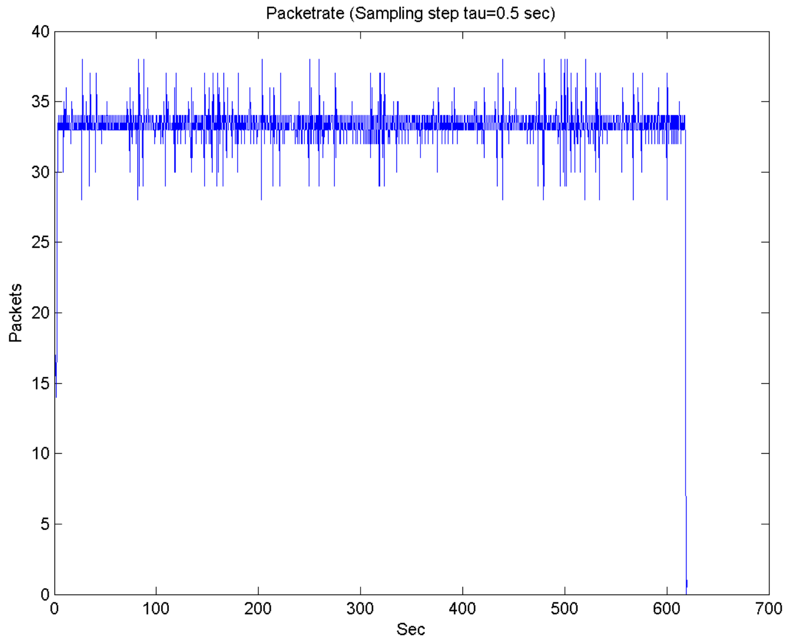

Figure 4.3: *Packetrate of a direct call, UDP flow, sampling rate = 0.5 s*

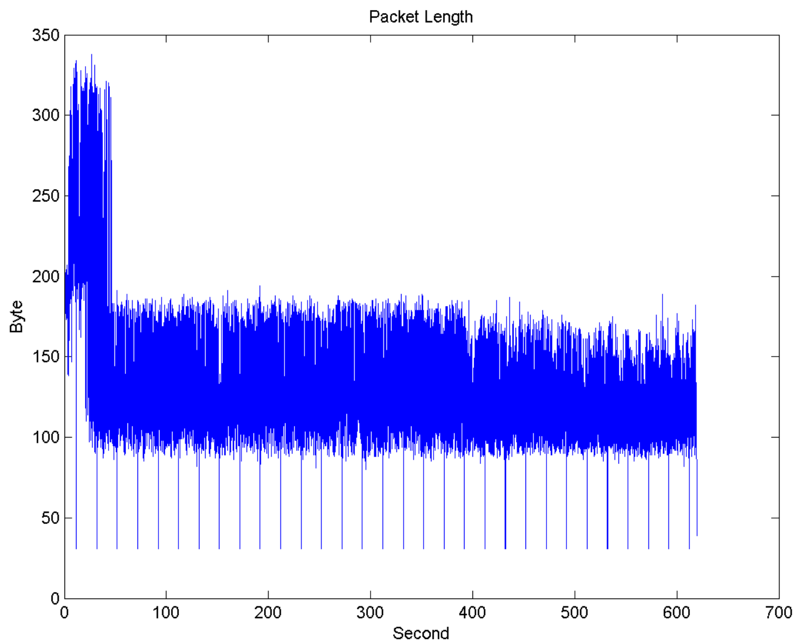

Figure 4.4: *Packet-Lenghts of a direct call, UDP flow*

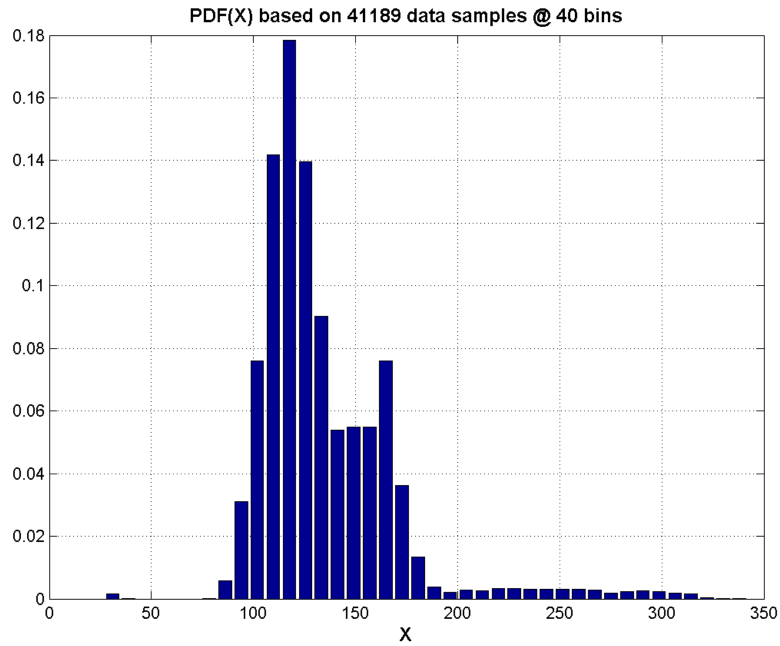

Figure 4.5: *Probability Density Function of packets length of a direct call, UDP flow*

(figure 4.4). It seems that they carry some kind of signalling, in addiction to that one carried by the TCP connection. In fact, even if the media is on a UDP flow, there is always a TCP connection between the two hosts (with characteristic ports used by Skype, that could be also chosen by the user) that exchanges a few packets at regular interval times.

Regarding the interarrival time almost all the packets arrive in 30 milliseconds after the previous one (figure 4.7). If the call is routed through Internet using a third host as relay, on average the same results are obtained.

4.2 Time Domain: TCP calls

As already written, the second test consisted in analyzing a direct call between caller and callee for which Skype was forced to choose TCP transport layer.

In order to force Skype to choose a TCP connection, a firewall blocking all the packets between caller and callee has been enabled. In this case the byterate

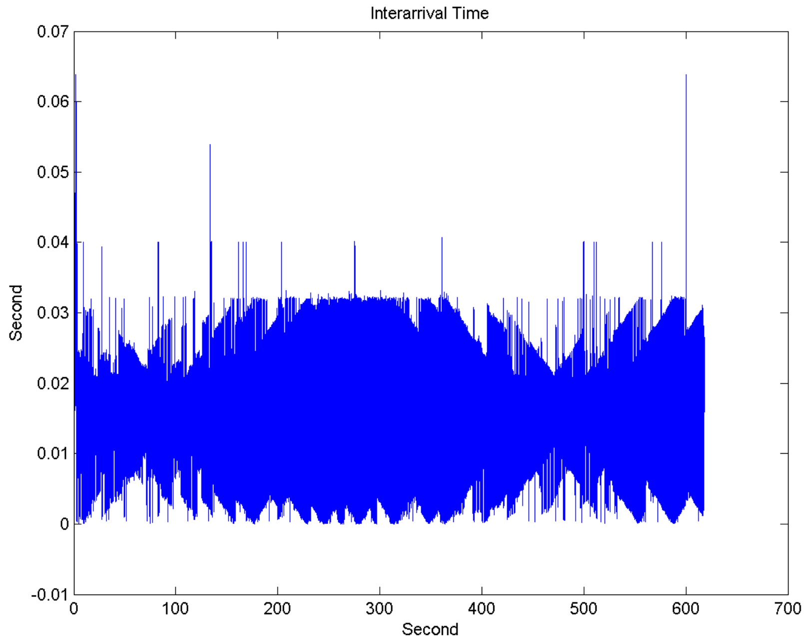

Figure 4.6: *Interarrival time of a direct call, UDP flow*

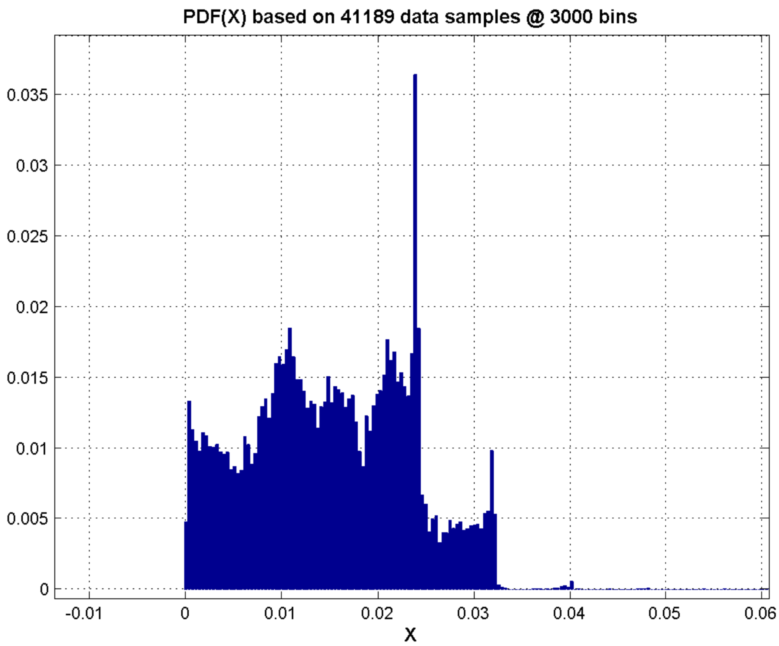

Figure 4.7: *Probability Density Function of Interarrival time of a direct call, UDP flow*

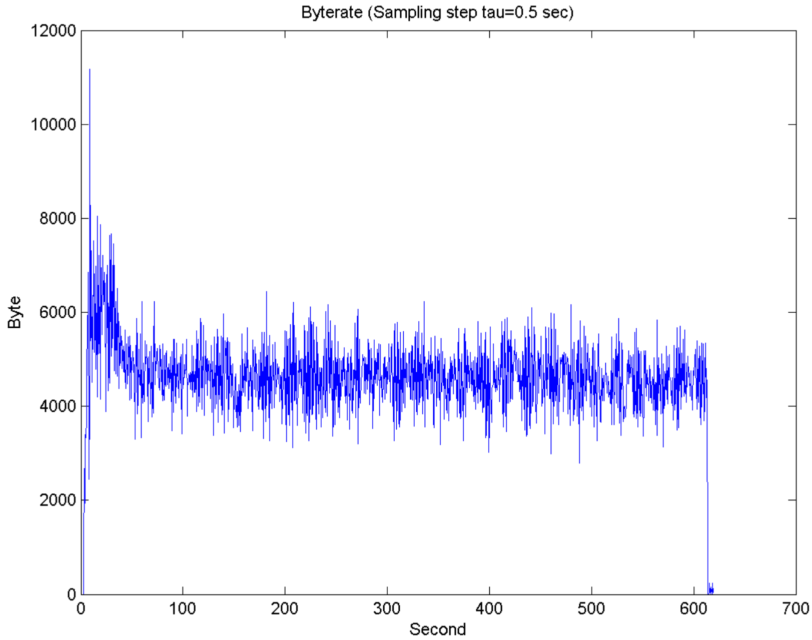

Figure 4.8: *Byterate of a direct call, TCP flow, sampling rate = 0.5 s*

at the beginning is also higher than in the rest of the call (figure 4.8), but the difference is lower than in the UDP case. This is because of the way in which the TCP packet is composed, and because of the features that distinguish TCP from UDP: ordered data transfer, retransmission of lost packets, discarding duplicate packets, error-free data transfer, congestion/flow control.

In fact, even if we can notice a sort of training period at the beginning of the call, fig 4.10, regarding packets length trend, shows a deep difference with the UDP case. It is much more costant on average, but with a greater variance around it. Same thing for the packet rate (figure 4.9), on average packets are longer (since error-free data transfer and congestion-flow control features each packet carries more information than the necessary) as it can be seen also from figure 4.11. In this situation it is also difficult to recognize the regular pattern of the "signalling" packets 36 bytes long from the figure 4.10, but from figure 4.11 it can be noticed the presence of these packets. The features offered by TCP can be easily noticed

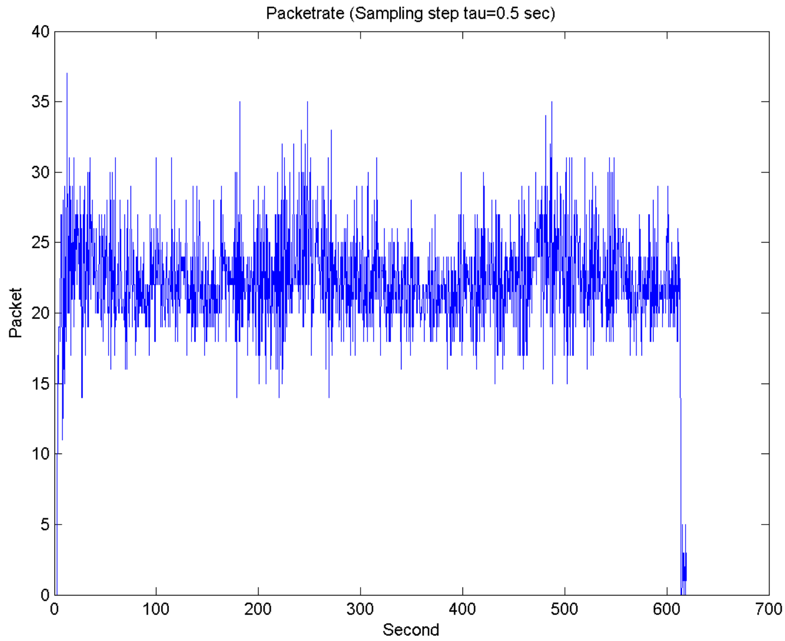

Figure 4.9: *Packetrate of a direct call, TCP flow, sampling rate = 0.5 s*

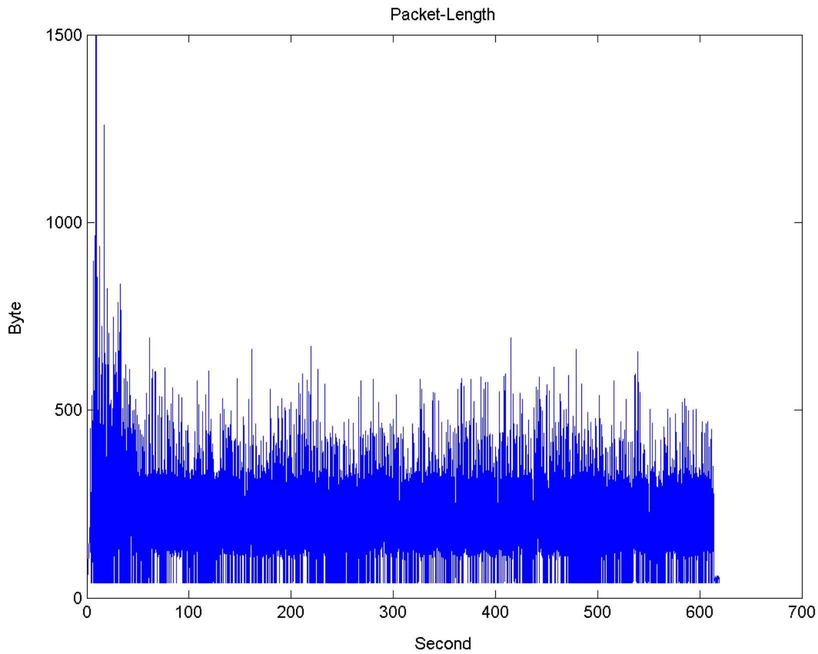

Figure 4.10: *Packet-Lenght of a direct call, TCP flow*

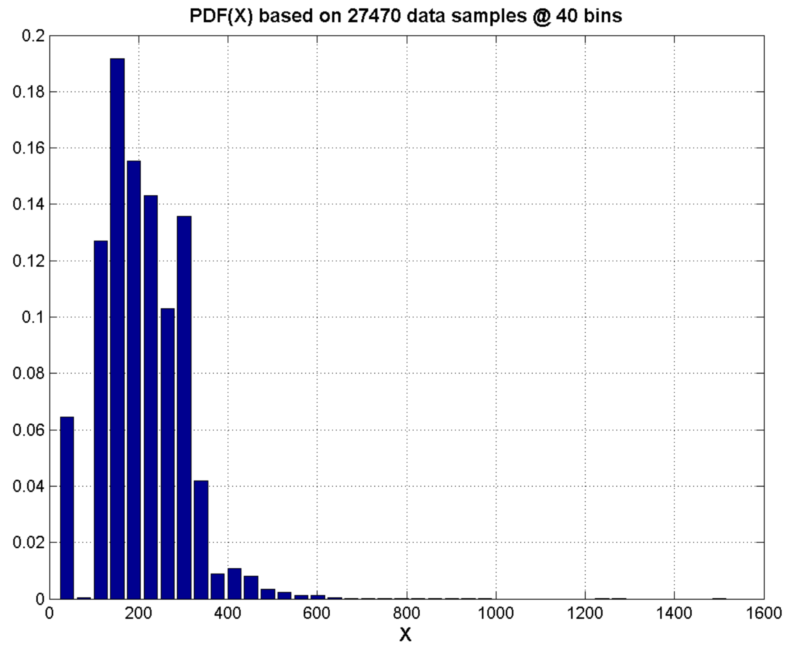

Figure 4.11: *Probability Density Function of packet length of a direct call, TCP flow*

also from figure 4.12 and 4.13 which strongly differ from the relative ones of the first type of tests. It is also interesting to mark the fact that the packet rate is no longer constant, but in two point of the trace it is a bit higher. If we compare this graphic with figure 4.12, it is possible to notice that these points correspond to two spikes in the second graph. In fact in figure 4.12 there are two high spikes, caused by some kind of congestion quickly solved by TCP, and the interarrival times are spread over a larger interval.

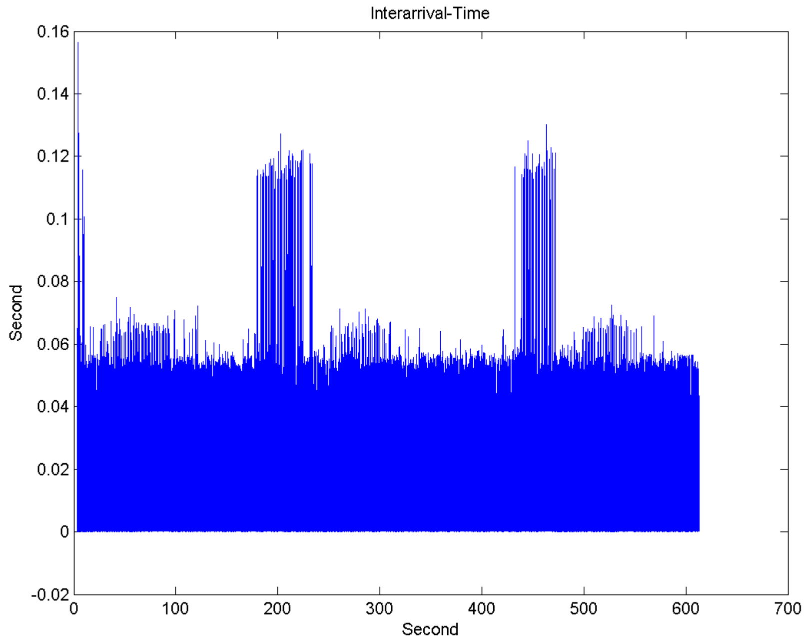

Figure 4.12: *Interarrival time of a direct call, TCP flow*

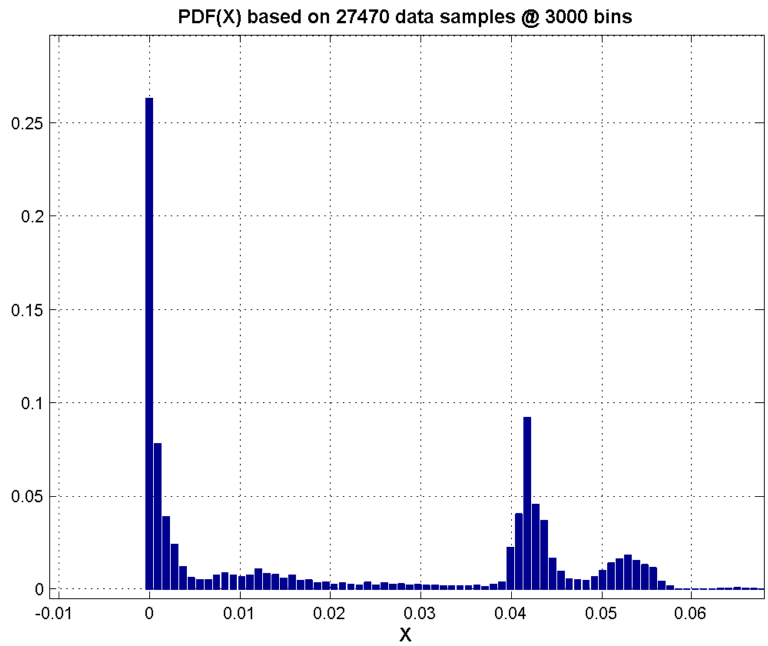

Figure 4.13: *Probability Density Function of Interarrival time of a direct call, TCP flow*

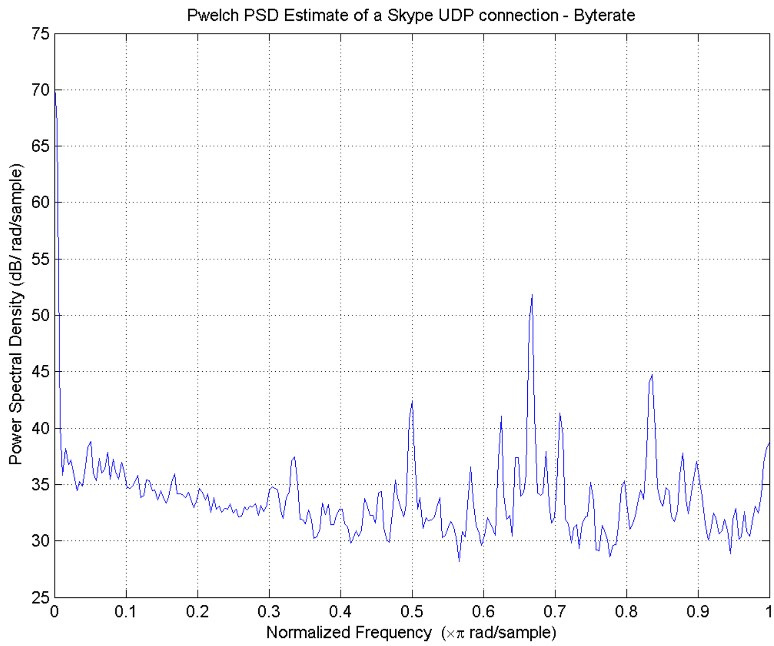

Figure 4.14: *PSD computed with Pwelch method of a UDP connection. Byterate, sampling period 0.5 s*

4.3 Frequency Domain

During this work the Power Spectral Densities (PSD) of the characteristics analyzed in the time domain have also been studied (from figure 4.14 to figure 4.19). As already reported no differences have been found between calls with different type of media delivered by the streams. The PSD of Byterate in the TCP case seems to exhibit a sort of periodicity around the RTT time. This periodicity leads to the presence of two spikes in figure 4.19 that represents the PSD of the Interarrival Time function.

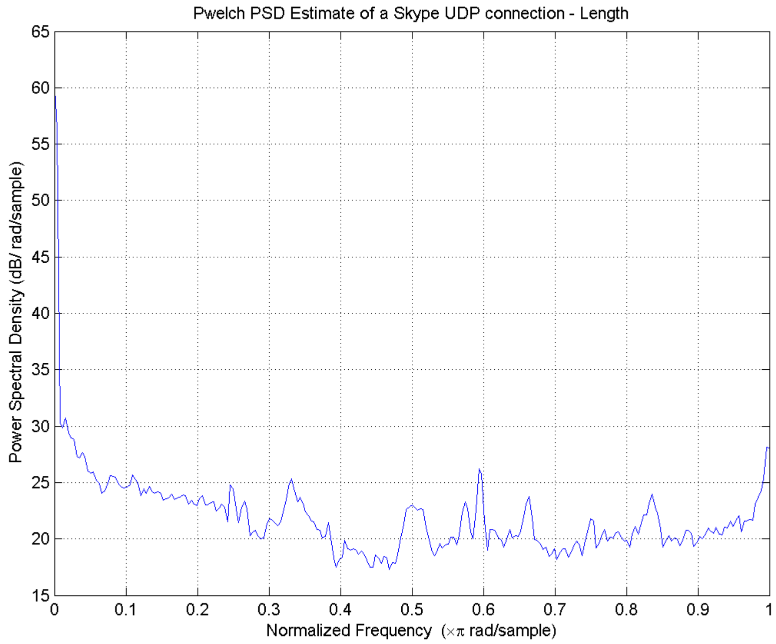

Figure 4.15: *PSD computed with Pwelch method of a UDP connection. Length*

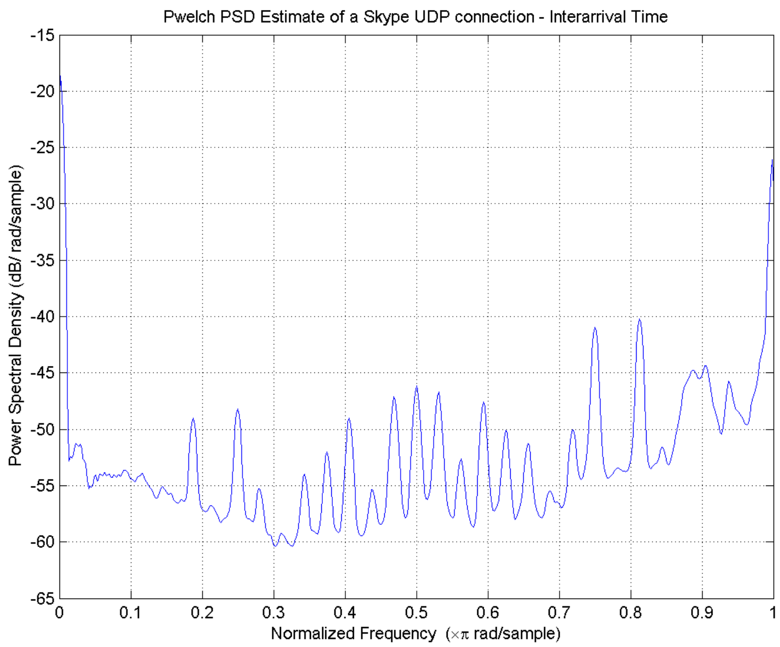

Figure 4.16: *PSD computed with Pwelch method of a UDP connection. Interarrival Time*

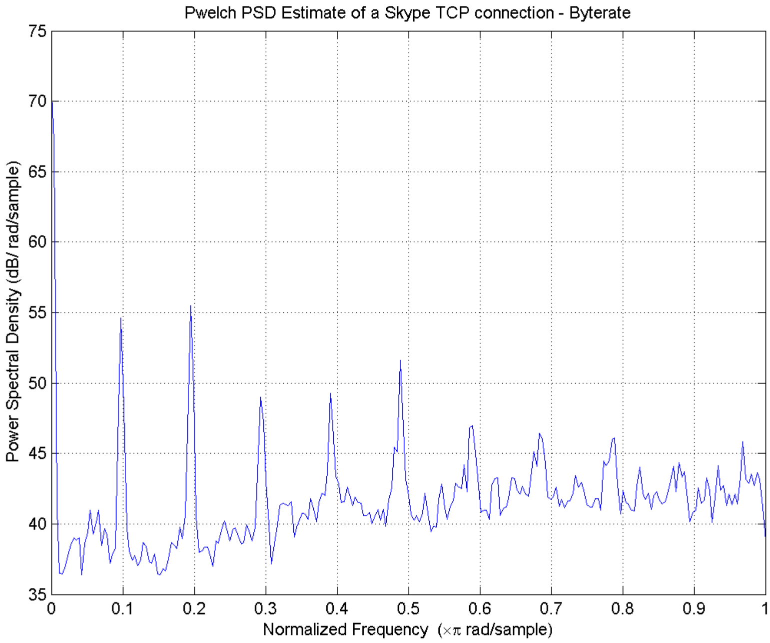

Figure 4.17: *PSD computed with Pwelch method of a TCP connection. Byterate, sampling period 0.5 s*

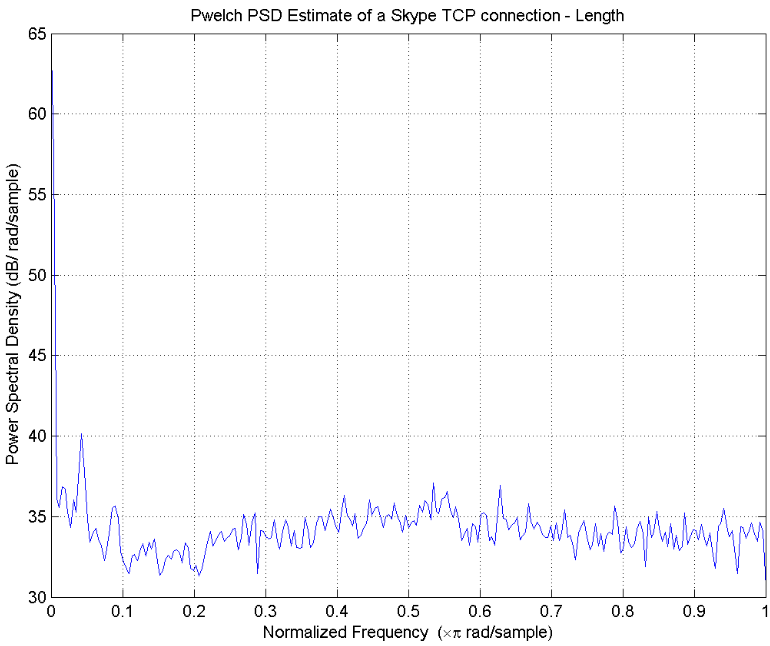

Figure 4.18: *PSD computed with Pwelch method of a TCP connection. Length*

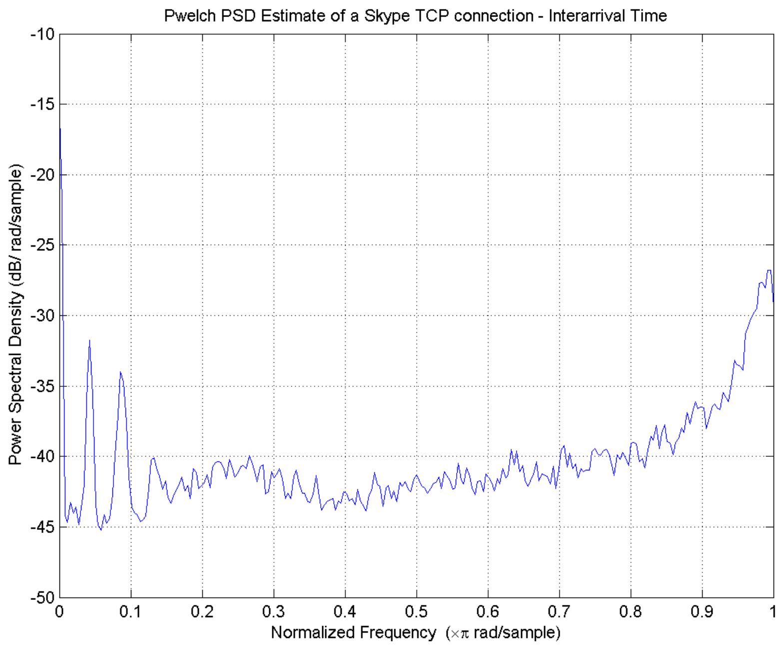

Figure 4.19: *PSD computed with Pwelch method of a TCP connection. Interarrival Time*

Chapter 5

Time-Domain Analysis Tool for Estimating the Hurst Parameter of Internet Traffic and its Applicability over Skype Traffic

In this chapter concepts like Long-Range Dependence and Self-Similarity are introduced and discussed; their use in Internet traffic analysis it is also addressed.

5.1 Self Similarity

Self-similarity and Fractal properties are concepts introduced by Benoit Mandelbrot [11] that describe the property for which a certain object characteristic, for example some statistical property of a temporal serie, keeps being constant respect to dilatation in both axes. If an object has self-similar characteristics, once

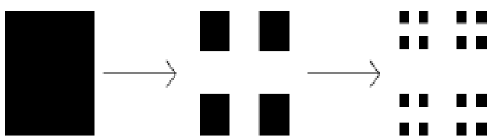

Figure 5.1: *Cantor-2D set, obtaining dilating the original object for a third of its size at each iteration*

dilated these characteristics are the same and indiscernible. A classic example is the Cantor set (figure 5.1), obtained from a single square. It can be noticed that, zooming a particular, it keeps being the same than the original one. Talking about internet traffic, for example number of TCP connections, packets arrival processes or others, the definition of self-similarity is not enough, but it is important to add the concept of stocastical self-similarity.

For example, in a temporal sequence it can be noticed that after a dilatation some characteristics are exactly the same. Given a temporal sequence X(t), if (X(t_1), X(t_2),......, X(t_n)) and (X(t_1+k), X(t_2+k),......, X(t_n+k)) have the same joint distribution, the process is said stationary. Called X_k the process shifted by k, this property could be summarized as:

$$X \stackrel{d}{=} X_k \tag{5.1}$$

Usually it is really hard to find a stationary process in a strict sense, so it is more common to use a "weaker" definition of stationarity, called stationarity of second order and it states that in a stationary process the mean of the autocovariance is the same even if the time interval is shifted. Defined:

$$\gamma(r, s) = E\left[(X(r) - \mu)(X(s - \mu))\right] \tag{5.2}$$

the autocovariance function of the process, it should be valid that:

$$\gamma(r, s) = \gamma(r + k, s + k) \tag{5.3}$$

for each value of r, s and k. If a process is self-similar then $\gamma(r, s) = \gamma(r - s, 0) = \gamma(k)$. It is possible to statistically define also the aggregate process, that is the process obtained from the dilatation in the time-domain. Defined:

$$X^{(m)} = \frac{\sum_{t=m(i-1)+1}^{mi} X(t)}{m} \tag{5.4}$$

the aggregate process at level m, obtained partitioning the $X(t)$ process in different and non-overlapping intervals, it is possible to associate the autocovariance function $\gamma^{(m)}$ defined by 5.3.

The process $X(t)$ is called self-similar of the second order with Hurst parameter H, with $(0 < H < 1)$ if

$$\gamma(k, H) = \gamma^{(m)}(k, H) \tag{5.5}$$

that means the autocovariance properties of the original process are equal to those of the aggregate one. It is possible to have a similar definition starting from the aggregate process $Y(t)$, connected to $X(t)$ by

$$X(t) = Y(t) - Y(t-1) \tag{5.6}$$

The process $Y(t)$ can be defined self-similar with Hurst parameter $(0 < H < 1)$ and it is called H-ss if, for each $a > 0$ and $t > 0$:

$$Y(t) \stackrel{d}{=} a^{-H} Y(at) \tag{5.7}$$

The process $Y(t)$ so defined is not stationary since the presence of the factor a^{-H}, and it has not the stationarity property of the second order. There are anyway some processes with stationary increments [12]. In this case $X(t)$ has the stationary property and so it is self-similar of the second order. These processes are defined H-sssi (H-self similar stationary increments). One important case of H-sssi process it is represented by Fractionary Brownian movement (FBM), that is the only Gaussian self-similar process with stationary increments, so it is really important.

With a H-sssi process and $t = 1$, $a = t$, we have

$$Y(t) \stackrel{d}{=} t^H Y(1) \tag{5.8}$$

from it, it follows

$$E\left[Y^2(t)\right] = \sigma^2 t^{2H} \tag{5.9}$$

This result, that shows the non-stationarity property of the process $Y(t)$, is used for calculating the autocovariance $\gamma(k)$. It is important to understand how this self-similarity property of $Y(t)$ described by 5.7 is connected to that one defined for the process $X(t)$. The process $X^{(m)}(t)$ can be seen as a result of an average process

$$X^{(m)}(t) = \frac{\sum_{t=1}^{m} X(t)}{m} = m^{-1}(Y(m)-Y(0)) = m^{-1}m^{H}(Y(1)-Y(0)) = m^{H-1}X(t) \tag{5.10}$$

so, if $Y(t)$ is a H-sssi process, its incremental process $X(t)$ accomplishes

$$X(t) \stackrel{d}{=} m^{1-H}X^{(m)}(t) \tag{5.11}$$

The 5.11 shows that $X^{(m)}$(t) and $X(t)$ are connected by a simple dilatation function subject to the Hurst parameter H.

This parameter has a really important role in defining the object properties [12]. It is clear when calculating the variance of $X^{(m)}$(t). Knowing that the mean $(\bar{Z})$ of a random variable Z accomplies

$$Var(X^{(m)}) = \frac{\sigma_Z^2}{m} \tag{5.12}$$

where m is the number of points, the variance of the aggregate process can be obtained from

$$Var(X^{(m)}) = \sigma^2 m^{2H-2} \tag{5.13}$$

Hurst parameter is important in determining the variability properties of the aggregate process.

The most important case is when $0.5 \leq H \geq 1$, where

$$Var(X^{(m)}) = \sigma^2 m^{-\beta} \tag{5.14}$$

with $0 \leq \beta \leq 1$ (and $H = 1 - \frac{\beta}{2}$). The 5.14 shows that the variance of the aggregate process hyperbolically converges to 0, so in a slower way than when the variance is computed following 5.12. This characteristic implies the presence of a long range memory property in the process.

5.2 Long-range Dependence

Long-range dependence (LRD) is a property that marks the presence, inside a process, of an important kind of memory, and the fact that the auto-correlation function doesn't converge. In particular, being $r(k) = \gamma(k)/\sigma^2$ the auto-correlation function of a $X(t)$ self-similar process, it can be computed that for $H \neq 0.5$

$$r(k) \sim H(2H-1)k^{2H-2} \qquad\qquad k \to \infty \tag{5.15}$$

Equation 5.15 marks the importance of parameter H in determining the asymptotic trend of the auto-correlation function [12]. For $0.5 \leq H \leq 1$:

$$r(k) \to ck^{-\beta} \qquad\qquad \beta = 2 - 2H \tag{5.16}$$

that implies

$$\sum_{k=-\infty}^{\infty} r(k) = \infty \tag{5.17}$$

The autocorrelation function has an hyperbolic trend, causing $r(k)$ not to converge. When 5.17 is true, the $X(t)$ process is said long-range dependent, if not, it is said short-range dependent. A similar definition of LRD can be also given in the spectral domain. In fact the PSD of the process can be expressed for a long-range dependent process as

$$\Gamma(\nu) \sim c|\nu|^{-\alpha} \qquad\qquad \alpha > 0 \tag{5.18}$$

It is important to underline that there are LRD processes that do not have the self-similarity property and there are self-similar processes that are not LRD. Only when $0.5 < H < 1$ a H-sssi process has the long-range dependancy.

5.3 Power-law Model

The LRD property is detectable, as said, because of the particular shape of the power spectral density of the process $X(t)$. The PSD is described by a power law model, this is why it is indicated as *power-law*. The one-sided power spectrum of $X(t)$ can be espressed as

$$(f) = \begin{cases} \frac{1}{(2\pi)^2} \sum_{\alpha=-4}^{0} h_{\alpha+2} f^{\alpha} & 0 \leq f \leq f_h, \\ 0 & f > f_h \end{cases} \tag{5.19}$$

where f_h is a cut-off frequency for limiting the spectrum. The five power-low noise types with integer α are: White Phase Modulation (WPM), for $\alpha = 0$, Flicker Phase Modulation (FPM), for $\alpha = -1$, White Frequency Modulation (WFM), for $\alpha = -2$, Flicker Frequency Modulation (FFM), for $\alpha = -3$ and Random Walk Frequency Modulation (RWFM) for $\alpha = -4$. It is also possible that α is not an integer; in that case it is common talking about Fractional Noise [12].

5.4 Modified Allan Variance

In time and frequency measurement theory, a well-known tool in the time domain for stability characterization of precision oscillators is the Allan Variance.

This variance was proposed in 1981 by modifying the definition of the two sample variance recommended by IEEE in 1971 for characterization of frequency stability [13], following the pioneering work of D.W. Allan in 1966 [14] Allan Variance is a particular case in a family of variances, said *M-Sample Variance*, defined for characterizing power-law signals given a finite number M of samples. These variances are defined as

$$\sigma_y^2(M, T, \tau) = \frac{1}{M-1} \sum_{i=1}^{M} \left(\bar{y}_i - \frac{1}{M} \sum_{j=1}^{M} \bar{y}_j \right)^2 \tag{5.20}$$

where τ is, as usual, the period where the mean of $y(t)$ is computed and T is the period between two different samples. This quantity is itself a stocastic variable of which it is possible to compute the mean.

Compared to the poor performances of this Variance against white and flicker phase noise, the Modified Allan Variance (MAVAR) discriminates all power-law noise types recognized so far. Given an infinite sequence x_k of samples evenly spaced in time with sampling period τ_0, the MAVAR is defined as

$$Mod\ \sigma_y^2(n\tau_0) = \frac{1}{2n^2\tau_0^2}\left\langle\left[\frac{1}{n}\sum_{j=1}^{n}(x_{j+2n}-2x_{j+n}+x_j)\right]^2\right\rangle \tag{5.21}$$

where the observation interval is τ=nτ_0. In practical measurements, given a finite set of N samples x_k, spaced with sampling period τ_0, an estimate of MAVAR can be computed using the ITU-T standard estimator

$$Mod\ \sigma_y^2(\tau) = \frac{1}{2n^4\tau_0^2(N-3n+1)}\sum_{j=1}^{N-3n+1}\left[\sum_{i=j}^{n+j-1}(x_{i+2n}-2x_{i+n}+x_i)\right]^2 \tag{5.22}$$

with n=1,2,...,$\lfloor N/3 \rfloor$. A recursive algorithm for fast computation of this estimator exists, which cuts down the number of operations to $\sim N$ instead of $\sim N^2$.

5.5 Application of MAVAR

The behaviour of the MAVAR on real IP traffic traces was finally tested. Traces have been obtained thanks to Tcpdump used on off-the-shelf hardware in the same environment of the previous tests. At first a TCP connection was analyzed, then the attention was moved to a direct UDP call. A third experiment was run with a firewall between caller and callee, in order to force Skype to route the call through a relay peer. Once traces were obtained, some characteristics of the flows were analyzed. These are packets-length, byterate and interarrival times between two following packets. These characteristics are series of samples, acquired with a certain sampling period from the tcpdump traces (e.g., for the byterate, τ=5 ms) over a measurement interval T of about 10 minutes. In the following tests the beginning and the end of the traces have not been analyzed in order to avoid transitory effects that would have affected the results. No nonstationary trends, such as steps, are evident. The Hurst parameter of a LRD sample realization x_k can be computed following this procedure:

1. Compute MAVAR (τ) with the estimator 5.22, based on the data sequence x_k;

2. Estimate its average slope μ in a log-log plot, at least in some intervals of values of τ, by best fitting a straight line to the curve (e.g., by least square error);

3. If -3 $\leq \mu \leq$-2 (i.e., -1$\leq \alpha \leq$0, 0 $\leq \gamma \leq$1) get the estimate of the Hurst parameter as

$$H = \frac{\mu}{2} + 2 \tag{5.23}$$

It is worthwhile noticing that the stimate of MAVAR computed from a finite number of samples is a random variable itself. Its variance can be computed and used to assess the uncertainty of the estimate of H.

In the next section some log-log plots of the MAVAR are shown. As it can be seen, these plots do not give too much information about the streams, meaning that MAVAR is not useful in characterizing these traces.

MAVAR plots In the next pages some log-log plots of the MAVAR are shown. As it can be seen, these plots do not give exhaustive information about the streams, meaning that MAVAR is not useful in characterizing these kind of traces. Some plots show characteristics that can be easily misclassified as white noise, making this analysis useless. In some other plots the unceirtainty of the measurement is too high to infer meaningful results. It can easily noted that in many plots the graph is not composed by straight lines, making a traffic characterization almost impossible.

Beside the Byterate-case, where we can assume a sort of LRD property, these graphs show us how in this kind of traffic there is property of LRD or Self-Similarity. In Byterate-case of a direct UDP call (figure 5.3) the slope of the line is α=-3, which corresponds to White Noise. This means that either the

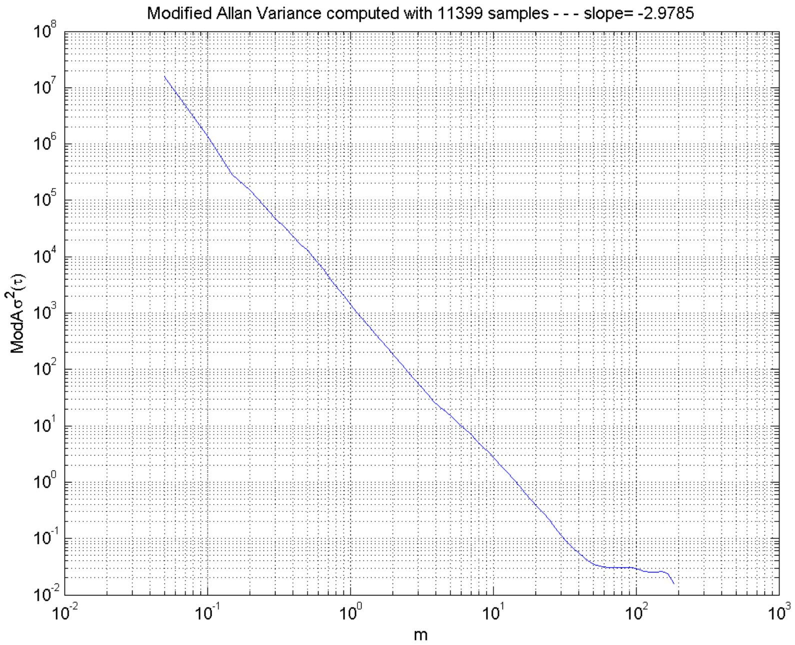

Figure 5.2: *MAVAR of Skype traffic trace for a direct UDP call. Samples are Byterate*

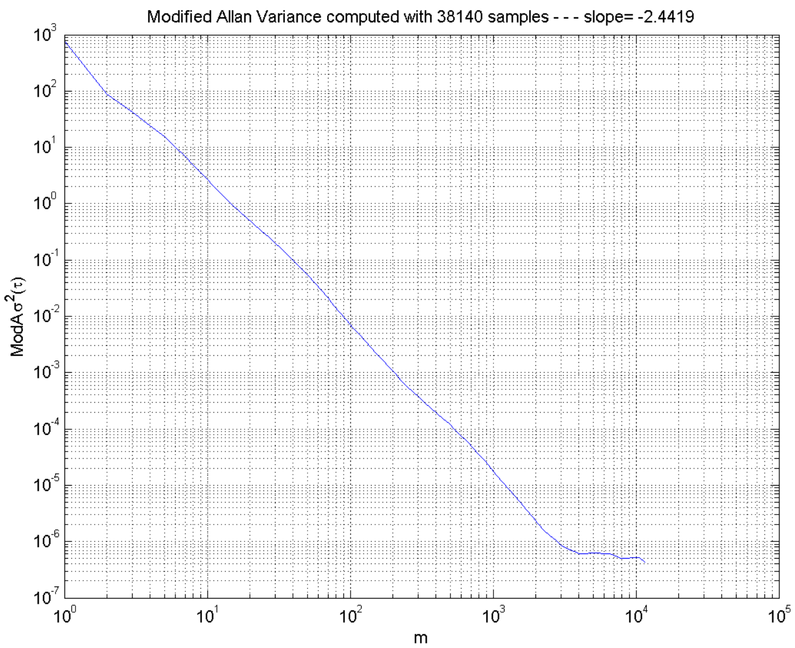

Figure 5.3: *MAVAR of Skype traffic trace for a direct UDP call. Samples are Length*

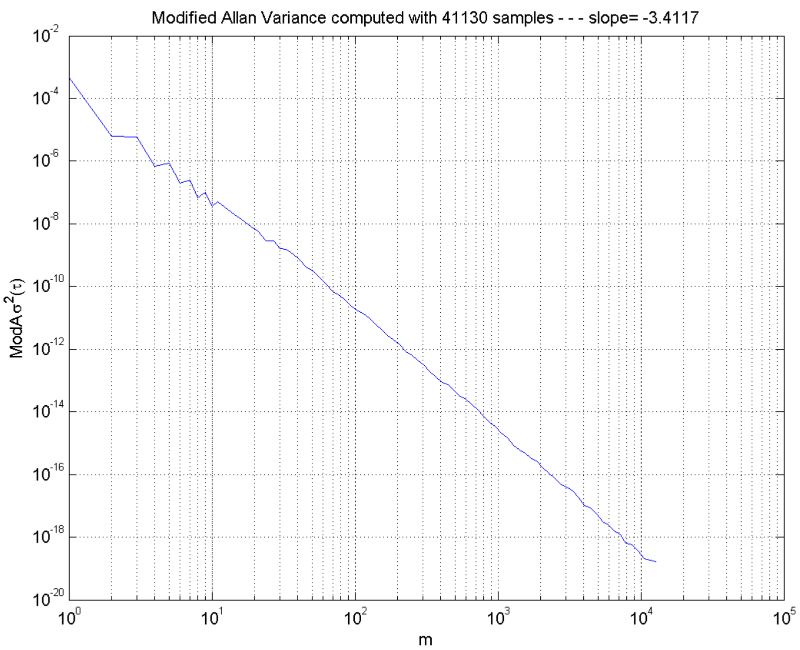

Figure 5.4: *MAVAR of Skype traffic trace for a direct UDP call. Samples are Interarrival*

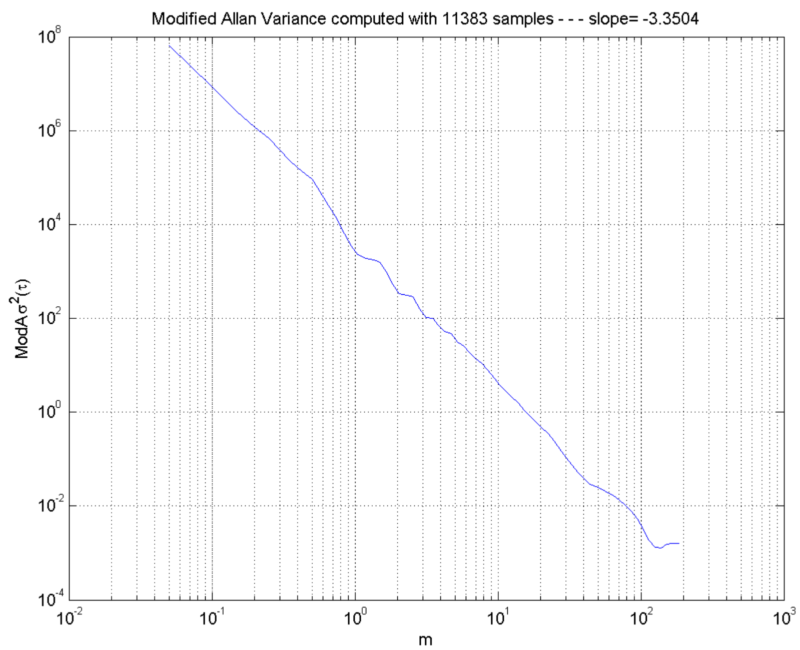

Figure 5.5: *MAVAR of Skype traffic trace for a direct TCP call. Samples are byterate*

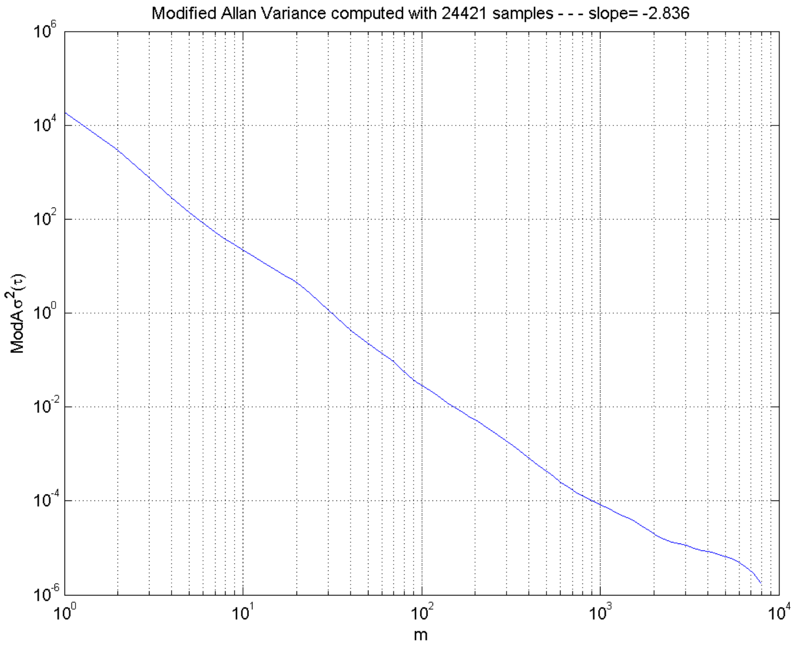

Figure 5.6: *MAVAR of Skype traffic trace for a direct TCP call. Samples are length*

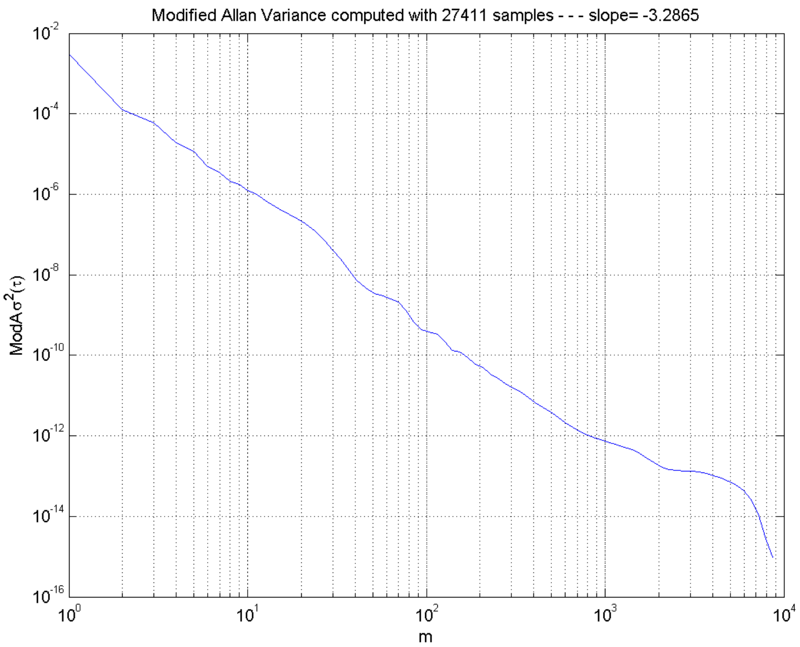

Figure 5.7: *MAVAR of Skype traffic trace for a direct TCP call. Samples are Interarrival*

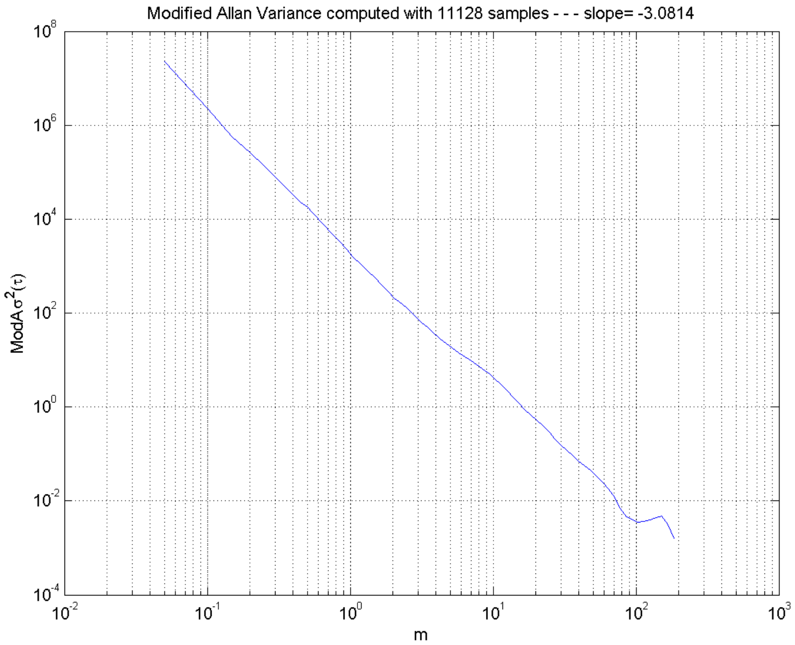

Figure 5.8: *MAVAR of Skype traffic trace for a rely-routed UDP call. Samples are byterate*

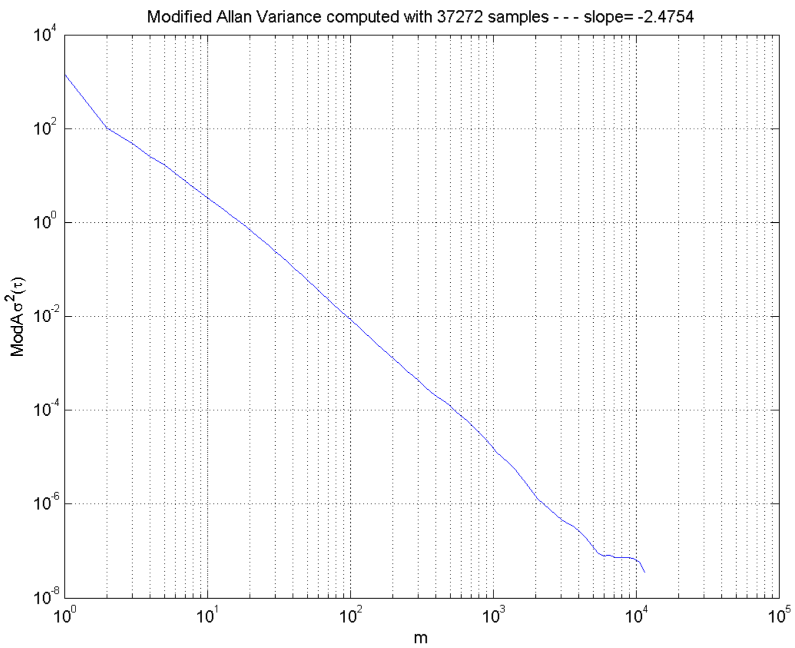

Figure 5.9: *MAVAR of Skype traffic trace for a rely-routed UDP call. Samples are length*

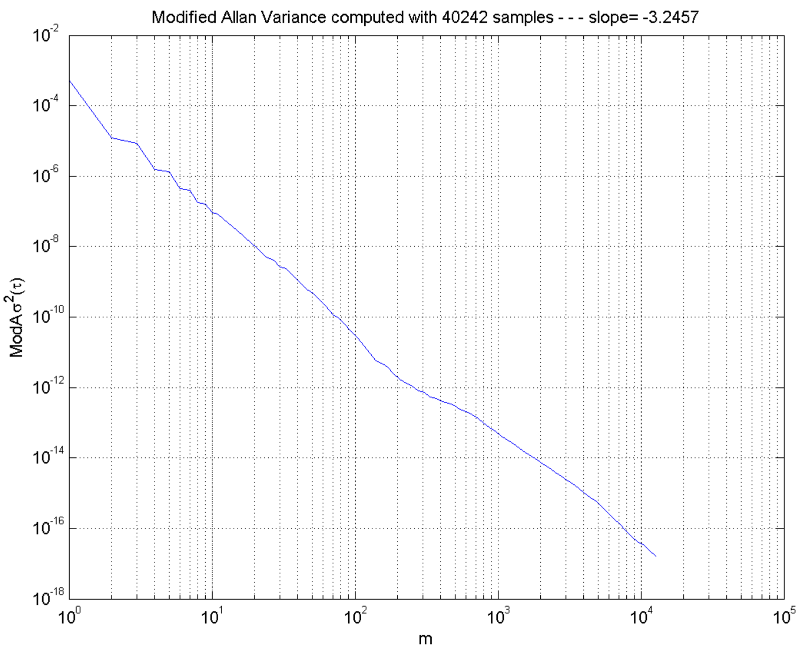

Figure 5.10: *MAVAR of Skype traffic trace for a rely-routed UDP call. Samples are Interarrival*

traffic presents a LRD property with Hurst parameter $H = 0.5$ or the trace is affected by so much noise making useless this kind of analysis.

One reason is that this kind of property can be more realitically found in aggregate traffic and not in a single traces as this work were trying to show.

Chapter 6

Statistical Traffic Classification

The motivation for the need of real-time streams classification, expecially in case of Skype, has already been exposed. As written, traffic classification is fundamental in planning and controlling a network.

Four different classification methods have been presented:

- Header-based methods focus on classification relying on information that can be found directly inside the header of the protocol at the transport layer used for the streams.
- Payload-based methods focus on classification on data insided the payload of the packets forming the stream.
- Statistical classification algorithms use statistical parameters computed over different characteristics, easily and directly obtained by the stream.
- Hybrid algorithms use the previous methods together, trying to obtain a better quality of the classifier, but usually increasing the computational complexity of the classification algorithm.

Knowing the origin of a traffic stream allows network administrators to apply different policies for each attempt of accessing to the network. For example it is possible to reserve resources to particular applications in order to give better quality service to critical traffic, or particularily demanding in capacity, delays, or jitter. Another possibility is to deny the access from potentially dangerous activities, or generically not required or authorized.

At the same time, many network planning related problems as capacity planning, routing planning, work-load modellization and characterization could find benefits from an accurate stream classification. In fact, if during the projecting step it is possible to know which links will be used by one or more particular types of traffic, it could be possible to plan in a better way resources in order to reach better quality with similar costs.

In order to reach these benefits, it is important to have a realtime identification process, therefore the classification must be done in the shortest time possible in order to apply the necessary policy. This point sets principally two limits to the algorithm that will be implemented by the classifier:

- It must use information that could be gathered during the first seconds of the stream, not using parameters, like connection length, that need the knowledge of the entire stream.

- It must be computationally not hard, even if more complex algorithms could give better results, they require higher computational times.

Some of these methods need information that can be easily obtained from the packet header, while others need the computation of more complex statistics, such as the average packets length, or others that need statistical correlation quantities among different streams and connection requests.

Progressive complications added in the classification algorithms had not only the aim of improving performances, but were often necessary in order to by-pass problems and particular situations created by the evolution of the network.

The incredible spread of peer-to-peer applications and the still on-going growth of audio-video services added new problems and difficulties, making quite useless previous methods.

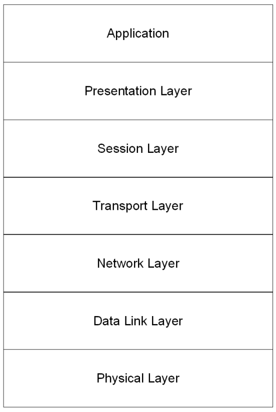

Figure 6.1: *ISO-OSI stack*

6.1 Header-based Method

One of the first methods that has been thought for identifying easily and quickly different applications was based on using transport layer port numbers. It is known that according to OSI standard (Open System Interconnection) introduced in 1978 by ISO, it is possible to divide network protocols in a stack composed of seven layers, according to tasks they perform. Figure 6.1 represent the ISO-OSI stack.

This model has been introduced for simplifying and regulating the functioning of a networking communication system. Each layer implements different services and it passes them to the upper layer, through standard interfaces called PoS (Point of Service). In this way it is possible to modify algorithms at a certain layer withouth affecting other layers.

Layer four is the transport layer, and its task is to safely and reliably deliver data. The most common protocols are, as known, TCP (Transmission Control Protocol) and UDP(User Datagram Protocol). The difference is that TCP can assure a reliable, in-order packets delivery, suitable for applications that need high reliability. Different kinds of applications that can afford lower reliability, but faster delivery usually rely on UDP.

Both protocols use port numbers for identifying applications that transmit and

Port No.	Protocol	Service Name	Aliases	Comment
7	TCP	echo		Echo
7	UDP	echo		Echo
9	TCP	discard	sink null	Discard
9	UDP	discard	sink null	Discard
13	TCP	daytime		Daytime
13	UDP	daytime		Daytime
17	TCP	qotd	quote	Quote of the day
17	UDP	qotd	quote	Quote of the day
19	TCP	chargen	ttytst source	Character generator
19	UDP	chargen	ttytst source	Character generator
20	TCP	ftp-data		File Transfer
21	TCP	ftp		FTP Control
23	TCP	telnet		Telnet
25	TCP	smtp	mail	Simple Mail Transfer
37	TCP	time		Time
37	UDP	time		Time
39	UDP	rlp	resource	Resource Location Protocol
42	TCP	nameserver	name	Host Name Server
42	UDP	nameserver	name	Host Name Server
43	TCP	nicname	whois	Who Is
53	TCP	domain		Domain Name
53	UDP	domain		Domain Name Server
67	UDP	bootps	dhcps	Bootstrap Protocol Server
68	UDP	bootpc	dhcpc	Bootstrap Protocol Client
69	UDP	tftp		Trivial File Transfer
70	TCP	gopher		Gopher
79	TCP	finger		Finger
80	TCP	http	www, http	World Wide Web
88	TCP	kerberos	krb5	Kerberos
88	UDP	kerberos	krb5	Kerberos
101	TCP	hostname	hostnames	NIC Host Name Server
102	TCP	iso-tsap		ISO-TSAP Class 0
107	TCP	rtelnet		Remote Telnet Service
109	TCP	pop2	postoffice	Post Office Protocol - Version 2

Figure 6.2: *Some well-known ports*

receive data on the end-points. Each packet is marked as belonging to a particular stream through destination port number and source port number present inside the packet header of layer-4 protocol.

TCP/UDP port number could be divided into three groups:

- Well-known ports (0-1023), to which applications are statically referred. (figure 6.2)
- Registered ports (1024-49151)
- Dynamic or private ports (49152-65535)

Theoretically it would be possible to identify an application simply looking at the used port number.

Anyway this technique is nowadays able to give only a little information, since many applications use not-standard ports for communicating.

Many of the famous peer-to-peer file-sharing or audio-video conference applications, try to avoid fixed ports, using dinamically chosen ports. Some examples are applications like Skype or BitTorren. They can change ports, but they can also use well-known port, like port 80, usually left for web-access.

6.2 Payload-based Method

A second method that can be used is the analysis of the packets payload.

This technique works at level seven of ISO/OSI stack, and consists in searching inside the payloads a special bytes sequence, called signature, that is characteristic of a particular software. Many classification softwares implement this strategy, usually with IP addresses, or port numbers analysis, since it is a really good and precise method to use, once a reliable signature has been found.

One example of these softwares is SNORT [15], that through rules written following a particular format, is able to identify different streams and to apply different policies depending on the application found.

A fundamental drawback of these systems is that it is necessary to store and update the signatures in order to respond to the developments and upgrades of the softwares. In addition, finding a rule, strict enough to not affect other streams, could be hard.

Another point is that this type of analysis is time consuming and computationally heavvy, expecially in high speed links, where the number of packets is really high.

In the end, this method, even if theoretically valid, is not practical. If we add the fact that nowadays more and more applications crypt data, it is useless.

6.3 Statistical Classification

As already mentioned there is a different classification method. This method relies on statistical characteristics of the streams in order to reach an application classification. The principal benefit of this kind of methods is that they can be used on each type of traffic, also crypted, since they do not need any kind of payload inspection.

The most used characteristics are the distributions of packets length and interarrival times that can be easily obtained from the analysis of a stream with any kind of origin. These characteristics often contain information that can be used for classification.

In general, a parameter is considered not good if:

- It does not add any kind of information about the knowledge if the stream.
- It is correlated with the info given by some other parameter.

In fact, a parameter that adds no information is totally useless and so there is no necessity for computing it. On the other hand, a parameter that gives an information already gained in some other way can be even harmful.

In order to gather the best possible parameters, two methods have been thought.

- Methods based on filtering elaborate information from the training set for determining the "quality" of the parameters.
- Wrapper methods use the same classification algorithms for determining the "quality" of a parameter, making the classifiers using different ones, and taking those who give better results.

In [16] the authors show how, using a k-NN algorithm (Nearest Neighbour) it is possible to classify streams belonging to seven different traffic classes, with an error rate lower than 10%. Statistical parameters used are:

- Average packet length
- Byterate

- Packetrate
- Mean Square value of packet length

while the seven different traffic classes are:

- DNS
- FTP (data)
- HTTPS
- Kazaa
- realmedia
- telnet
- WWW

As can be noticed, the statistical parameters refer to length and numbers of stream packets, while traffic classes are heterogeneous, able to cover a huge variety of applications.

In [17] the authors use a more complex classifier, called Bayesian, based upon Bayes theorem and prior probabilities.

The traffic classes considered by the authors are:

- WWW
- mail (POP3, SMTP)
- ftp (data)
- SSH-telnet
- Database (sqlnet, etc)
- DNS

- Peer-to-peer
- Games (Half life)
- Attack (virus, works, etc)
- Multimedia (realmedia)

Also in this case, the set is really wide and heterogeneous, covering many possible applications.

The algorithm could also be expressed mathematically. If c_j is the traffic class *j-th* and y the stream to be classified, it is possible to write, by the Bayes theorem, that:

$$p(c_j \mid y) = \frac{p(c_j)f(y \mid c_j)}{\sum_k p(c_k)f(y \mid c_k)} \tag{6.1}$$

where $f(y \mid c_j)$ is the distribution probability function of y, given c_j, while the denominator works as a normalizing constant.

The algorithm needs a training period for determining the prior probabilities $p(c_j)$ and the distribution probability function $f(y \mid c_j)$ for each class.

A simple method for computing $f(y \mid c_j)$ is to presume that it has a Gaussian probability distribution with mean and variance computable from the stream.

In the article, actually, it is demostrated as the Gaussian assumption is too far from the reality for many kind of streams, and the probability error for this classifier is around 35 %.

Anyway it is possible to improve the classification simply considering different distribution for different application. In this case 93,50 % of the streams were correctly classified.

Besides the statistics related to lengths and interarrival times, it is also shown in [18] that other characteristics, such as TCP flag, can help in traffic classification.

6.4 Hybrid Algorithms

A different method for traffic classification uses more techniques at the same time, in order to by-pass problems caused by one method and to take advantage of the benefits given by another one.

The method proposed in [19] aims to analyze the host behaviour at three different levels:

- Social: the host behaviour is analyzed respect to the interaction with other hosts in the network, for example monitoring the number of hosts in contact, or checking the presence of hosts community inside the network.
- Functional: the host is analyzed with respect to the role it assumes inside the network (client, server or peer). A way is to check whether it uses few or many ports for communicating.
- Applicative: transport layer interactions are checked, comparing stream characteristics with a pattern of reference, based on various statistics.

This algorithm is of course more complex than others, because in addiction to the statistics of each stream, it is necessary to keep trace of the behaviour of the hosts, meaning collecting information about number and type of connections, way of connection etc.

On average this kind of classifier is able to correctly classify 90 % of streams. It is important to notice that this mechanism is not able to distinguish different applications belonging to the same traffic class. It means that it is able to differentiate a HTTP stream from a peer-to-peer one, but not eMule traffic from GNUtella traffic. It has also been shown that with crypted traffic, expecially where the transport layer header is also crypted, there could be misclassifications.

Chapter 7

Statistical Fingerprinting for Skype Classification

The approach that has been used through this study tries to classify network traffic relying exclusively on the statistical properties of the flows. The key idea, as already shown, is that some statistical properties of basic elements of each network flow, for example the size of the IP packets, their inter-arrival time and the order in which they are seen at the classifier should be sufficient to determine which application has generated the traffic. In this work, the definition of "protocol fingerprints", which express the three properties above mentioned is explained in a compact and efficient way.

Using the definition of anomaly score, the protocol fingerprints used, allow the measurement of how "far" an unknown protocol is from the basic characteristics of a given protocol. Then a classification algorithm, based on the protocol fingerprints, is used for the final classification. The algorithm can classify flows dynamically as packets pass through the classifier, deciding if a flow belongs to a given application layer protocol, or if it was generated by an "unknown" (that means not-fingerprinted) protocol.

The important differences from all the existing approaches are the application of a smoothing filter on the Probability Density Functions (PDF) of the quantities expressed in the protocol fingerprints, in order to take into consideration the effect of noisy factors, and the use of packet arrival order as one of the main elements in the definition of protocol fingerprint. The proposed algorithm is computationally simple: the comparison of a flow against a known fingerprint requires only the algebraic sum of a few terms obtained by looking up values in PDF tables. Moreover, the low false positive ratio in the detection of not-fingerprinted traffic (obtained in the experimental phase and explained later) indicates the effectiveness of fingerprints for protocol characterization and a certain degree of robustness of the classifier against the emergence of new application layer protocols can be reasonably expected. This technique is at a very early stage of development. It has been tested by Salgarelli in [20] and then a modified version has been used through this work.

7.1 Protocol Fingerprints

The classification of IP flows produced by Skype protocol through UDP and TCP flows has been tested. The definition of flow F is a bidirectional, ordered sequence of IP packets produced either by one peer towards the other, or the way around during an application layer session. At the IP layer the flow F can be characterized as an ordered sequence of N pairs $P_i{=}(s_i, \Delta t_i)$, with $1 \leq i \leq N$ and N is the total number of packets in the flow, where s_i represents the size of the packet i and Δt_i represents the inter-arrival time between Pkt_{i-1} and Pkt_i. All the work is based on the assumption that the statistical information contained in an appropriate amount of flows generated according to the same application layer protocol rules should be enough to decide whether an unknown flow is in agreement with such protocol or not. The information given by these statistical properties is what has been defined "protocol fingerprint".

The idea behind the use of packet size, inter-arrival times and arrival order (of packets) for the classification of network flows lies in the observation that at least during the beginning stage of each layer-4 connection, the statistics related

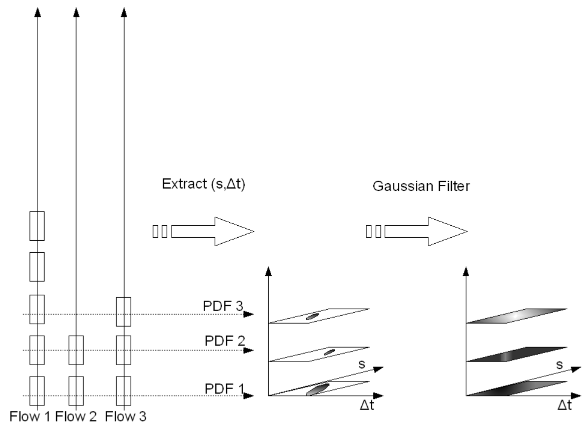

Figure 7.1: *Summary fo the procedure used to derive the protocol mask, necessary for the classification algorithm*

to each of these quantities depend mostly on the application-layer state machine that has generated the flow.

The generation of a given application layer protocols fingerprint starts from the evaluation of a set of T Probability Density Functions (PDF_i) estimated from a set of flows (a training set) generated by the same application and captured by a monitoring device. The i-th PDF_i is built on all the i-th pairs P_i belonging to those flows that are at least *i+1* packets long. The objective of the PDF_i is to describe the behaviour of the i-th packet on the three-dimensional plane (probability, packet-size s, inter-arrival time Δt) for a certain protocol. In this way the packets' arrival order is also taken into consideration, since only all the i-th packets of the flows in the training set take part to the computation of PDF_i.

Variable s is discrete and assumes values in a range dimension according to the minimum and maximum size of an IP packet in the network interface used to collect the traces. The plane that has been chosen for computing the PDFs contains sizes within the range 30-1490 (bytes). Variable Δt is, instead, sampled with resolution coherent with the speed of the network interface used to capture the traffic traces and with the clock resolution of the capture device (tcpdump), and binned accordingly. It has been decided to bin the values along the (log_{10})

Δt-axis from 10^{-9} to 10^{1} (seconds), with step 0.01. So each resulting PDF_i matrix is 1461x1001. At the end of the training period, chosen the number T of packets that should be analyzed, the machine prepares a vector of PDFs, $\vec{PDF}$, which length is T.

In order to classify a flow passing inside the classifier, it is necessary to control if the behaviour of the unknown flow is statistically compatible with the description given by $\vec{PDF}$ vector.

7.2 Anomaly Score

The way for doing it is to define an *anomaly score* S that is able to describe the "statistical distance" between the training set (and so $\vec{PDF}$) and the unknown flow F. The value should give the correlation between the unknown flow's i-th packet and the application protocol layer described by $\vec{PDF}$, that means the i-th component of the vector. It can be done looking at the value that the i-th component of $\vec{PDF}$ assumes in P_i, with P_i being the i-th pair of the unknown flow: the higher the probability value, the higher the possibility that the unknown flow has been generated by the same application of $\vec{PDF}$. Anyway it is important to consider the fact that these values are affected by forms of "noise" (variability in round trip times caused by congestion, differences in the local implementation of each protocol's state machine and others). The idea regarding how to take these problems into account is to consider not only the value in P_i, but also in a region close to it. This is the exact idea behind the concept of protocol mask $\vec{M}$ as the basic component of a protocol fingerprint. $\vec{M}$ is defined as a vector of T matrices resulting from the application of a Gaussian smoothing filter to the components of $\vec{PDF}$. After the application of the filter each matrix is rescaled in order to normalize the sum to 1, according to the general property of a PDF.

Figures 7.2 and 7.3 represent the distribution of a Probability Density Function before and after the application of the Gaussian Filter.

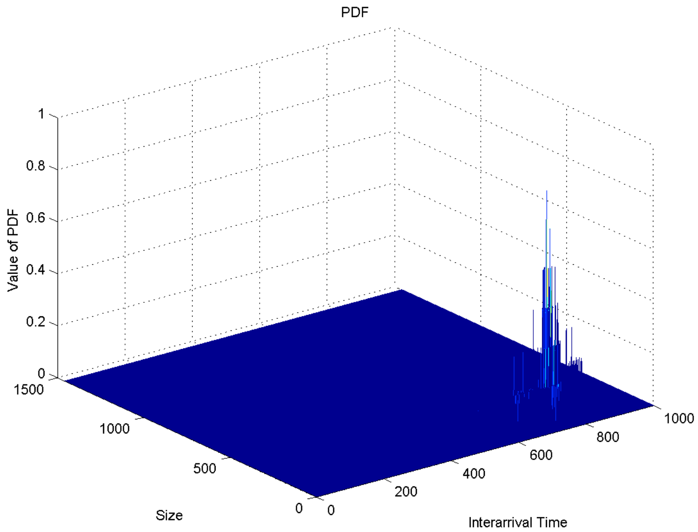

Figure 7.2: *Probability Density Function*

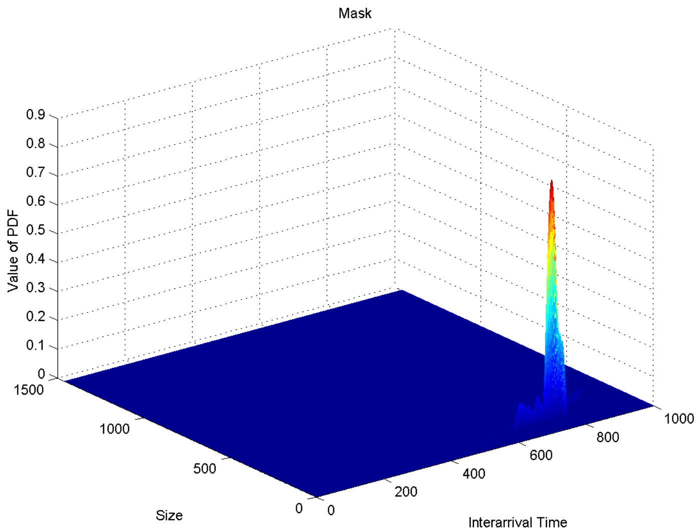

Figure 7.3: *Mask obtained smoothing the PDF with a Gaussian Filter*

The Gaussian smoothing operator is a 2-D convolution operator that is used to "blur" images and remove details and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian ("bell-shaped") hump. The idea of Gaussian smoothing is to use this 2-D distribution as a "point-spread" function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we could truncate the kernel at this point.

As already explained the i-th component of the anomaly score is a function of the i-th component of the value of $M_i(P_i)$, and it is defined as follows:

$$A_i(P_i, M_i) = \frac{1}{max(\epsilon, M_i(P_i))} \tag{7.1}$$

where ϵ is a small positive quantity. Its function is to let the score be always finite, even when M_i is zero in P_i.

Starting from this definition of the anomaly score vector $\vec{A}$ it is possible to introduce the concept of the anomaly score S of F versus $\vec{M}$:

$$S_n(F, \vec{M}) = \frac{[\sum_{i=1}^{n} A_i(P_i, M_i)/n] - A_{min}}{A_{max} - A_{min}} \tag{7.2}$$

where n represents the number of packets considered for the classification, and A_{min}, A_{max} are the extreme values of A (respectively, 1 and ϵ^{-1}). This implies that $0 \leq S_n(F, \vec{M}) \leq 1$. S is a function of n, the number of packets of the flow that the classifier "sees" during the analysis before taking a decision. All these vectors take part in the definition of protocol's fingerprint.

The classification decision is built on protocol's fingerprint and anomaly score. The choice of the threshold is important because if the value of the anomaly score is below the threshold, then the flow is considered belonging to the group that originated the fingerprint.

7.3 Using the Technique in Practice

The application of this technique is pretty simple.

There is the need of collecting pre-classified traces of Skype calls in order to form a training set. This can be easily done with Tcpdump or any other traffic-capture mechanism available.

Once having the training set it is possible to build the protocol fingerprints based on the pre-classified traces. After that, it is possible to start the classification machine, that would use the protocol fingerprints for deciding the origin of an unknown flow, extracted from live traffic.

It is important to periodically update the fingerprints, in order to prevent any misclassification caused by different versions of the applications.

The way the algorithm has been thought allows its application in a dynamic environment: as the classifier sees more packet of a flow, it can increase the chances of correctly classifying it.

The validity of the classification algorithm has been tested in the faculty's network of Politecnico di Milano. Two thousands calls have been collected at the beginning of each test in order to build the training set. All the calls were 30 seconds long, collected waiting 1 second between each one. Transported media has always been music (but as already shown there is no difference for this kind of tests between silence, music and spoken language). The experimental analysis was run by following pretty closely what a network administrator would have to do to implement this classification technique.

All the results were obtained tuning the classifier with the parameters that seemed to be the best for achieving optimal results. Each fingerprint was obtained applying a 20×20 window size for the Gaussian Smoother and the standard

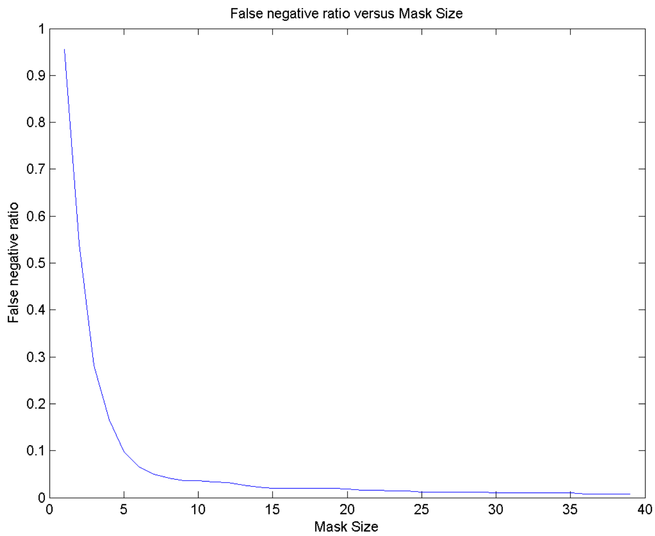

Figure 7.4: *False-negative Ratio versus mask size*

deviation of the function was set equal to 4.55.

The classifier took its decision after the 10th packet of the flow under analysis. It has been chosen to analyse only 10 packets because in this way a real time use of the classifier is possible without affecting the quality of the decision. In figure 7.4 it is shown how the mask size affects the false-negative ratio and so the quality of the classifier. It should be added that the bigger the size, the more complex the operation of building the Gaussian filter is. Anyway this operation needs to be done only once, after collecting the traces and then it is enough to save only the mask vector, so it does not affect the real-time operation of the classifier.

As said, another parameter that was chosen is the standard deviation of the Gaussian function. In figure 7.5 it is shown how it affects the decisions of the classifier. It can be easily noted that any value bigger than 1.5 leads to the same results, it does not affect the decision of the classifier. This is why in all our tests, the standard deviation was set at 4.55.

The classifier performs quite well as it can be seen in figure 7.6. From the figure it is possible to realize the performance of the classifier in different tests.

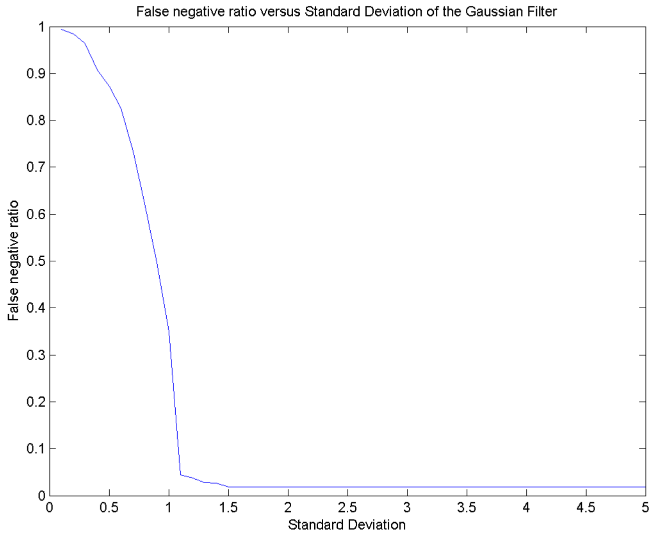

Figure 7.5: *False-negative Ration versus Gaussian function standard deviation*

Test 1 consisted of collecting traces from calls between hosts inside the same network. This was possible thanks to a Java application that automatized the calls. Then each trace was analyzed by Tcpdump thanks to a Unix bash shell script. In this way the vectors necessary to the computation of the $\overrightarrow{PDF}$ are built and then passed to the classifier. In test 2 the caller was in a different network than the callee. Of course for both of these tests it has been necessary to build two vectors, the training (with pre-classified traces) and the evaluating one. In test 3 the training vector from test 1 has been used but the evaluating vector was the one from test 2, vice versa in test 4. Test 5 and 6 use vectors obtained mixing traces from test 1 and test 2. In test 7, but also in other tests in which same results were obtained, the false positive ratio was checked. In fact, in these tests the evaluating traces did not belong to Skype traffic, but to different Internet protocols, like FTP or HTTP or media delivering protocol like RTP and RTCP. In all the tests the False Positive Ratio was equal to 0.

These results should be taken carefully since the study is still at an early stage and the effective applicability should be studied more. The pre-classification phase

Test	False Negative Ratio
Test 1	1.7%
Test 2	1.3%
Test 3	2.6%
Test 4	2.57%
Test 5	0.99%
Test 6	0.01%
	False Positive Ratio
Test 7	0%

Figure 7.6: *Test 1 has been performed inside the campus network, Test 2 has been perfomed with calls between one host inside the campus of Politecnico di Milano and one host outside it. Test 3 and 4 have been performed using Mask from test 3 for valuating data from test 4 and the way around. In test 5 and 6 the mask has been created mixing data from test 1 and 2. Test 7 has been performed in order to evaluate false positive ratio (traces gathered not from Skype traffic).*

of the flows for the creation of the training set is still an issue because having pre-classified traces, expecially in peer-to-peer traffic, is quite demanding and hard to achieve in an automatic way.

Another important factor is the "transportability" of the fingerprints. It is not sure that a fingerprint taken at one site could be used for classification at a different site, even if tests 3 and 4 show low false negative ratios. this needs more investigations.

Aside from the fingerprints the way the classification algorithm has been thought implies that the engine could be implemented directly in hardware. This device would require memory in order to save the number of fingerprints necessary to classify all the wanted or unwanted traffic in the network.

It should be also important to analyse the precision of the device. It is known that *pcap* (and so Tcpdump) is not precise but with this level of precision it seems to be quite accurate in dumping the traces in the environment used for the tests. There could be problems in networks with higher data-rate but in that case it could be sufficient to use a proper dedicated hardware for the classification machine.

Tests 5 and 6 present better results and this can be motivated by the fact that the training set has been created with twice as many traces as in all the other tests. In fact, a bigger number of traces belonging to the training set ensures higher classification quality because of more detailed PDF vectors. It would be interesting to analyse how deeply this factor influences the overall process. Sometimes it is not so easy to obtain, store and process large amount of data considering also the fact that this should be done for each of the known protocols that should be monitored. This is therefore another open issues that should be taken into consideration in future works.

Chapter 8

Conclusion and Outlook

In this work an analysis of Skype traffic has been proposed. At the beginning of the work a general analysis of the Skype protocol has been introduced. Skype has several features making Skype usage detection hard at network level, which are unavailable protocol specification, complete encryption of all messages, clandestine port number usage and several different operation modes. Detecting Skype traffic is not an easy task. Because of Skype's P2P character, the security operator has to install monitoring systems at all network egress points to detect all user operations. Additionally, exact detection of Skype usage in high traffic scenarios requires powerful monitoring hardware. After a general overview of Skype network protocol, an analysis of Skype traffic in both time and frequency domain has been proposed. It has been already written that Skype protocol hides the type of traffic transported (file transfers or voice/music/silence media) and that a "training" period of the call possibly caused by the codec can be observed in the byterate graph (figure 4.2). A further attempt to characterize the protocol has been tried with the MAVAR method. In this case the Modified Allan Variance is not useful and its application over this type of traffic does not give any additional information.

A new classification method based on statistical fingerprints has been introduced and tested. Protocol fingerprints are quantities that are at the basis of the training phase, and that express in a efficient way the main statistical properties used to classify and characterize each protocol: interarrival time and size of the packets and the arrival order in which they are seen at the classifier. The first results have shown good possibilities for this kind of characterization with a small error ratio. The analyses showed also that the use of the Gaussian filter is fundamental in helping the classifier to overcome the effects caused by jitter and unexpected changes in packet sizes.

This algorithm is still at an early stage of development and many issues should be analysed more deeply, as discussed in Chapter 7. It would be interesting to better understand the "transportability" of the fingerprints, the importance of the Gaussian filter (and moreover if any other Smoothing filter could be used with better results) and a real and effective usage in a public environment. Another important issue is the validity of the fingerprints since tests suggest that after a certain time Skype changes traffic characteristics. In fact, we have found that if the fingerprints are older than one month, the classification algorithm is no longer effective. Anyway, the good results so far achieved and all these open issues give motivation to keep working on this technique.

Bibliography

[1] Hudson Barton. Skype growth: Analysis and forecast for 2007. *http://homepage.mac.com/hhbc/blog/skypegrowth/skypegrowth.html*, 2006.

[2] Salman A. Baset and Henning Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. 2004.

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

[4] Sandrine Petgang Sven Ehlert. Analysis and signature of skype voip session traffic. *Fraunhofer FOKUS Technical report NGNI-SKYPE-06b*, 2006.

[5] Skype. *www.skype.com.*

[6] Global IPSound. ilbc codec. *http://www.globalipsound.com/pdf/gips iLBC.pdf.*

[7] Global IPSound. isac codec. *http://www.globalipsound.com/pdf/gips iSAC.pdf.*

[8] Batu Sat Benjamin W. Wah. Analysis and evaluation of the skype and google-talk voip systems. *IEEE INT'L Conference on Multimedia and Expo*, 2006.

[9] Kurt Tutschku Markus Fiedler Tobias Hoß feld Andreas Binzenhöfer. Measurement and analysis of skype voip traffic in 3g umts systems. 2006.

[10] Kyounfwon Suh Daniel R. Figureido Jim Kurose Don Towsley. Characterizing and detecting relayed traffic: A case study using skype. *UMass Computer Science Technical Report 2005-50*, 2005.

[11] B. Mandelbrot. The fractal geometry of nature. *Ed. W.H. Freeman, New York*, 1982.

[12] Luca Jmoda. Analisi nel dominio del tempo di traffico ip per mezzo di varianze di hadamard e grandezze derivate. *Master Thesis, Politecnico di Milano*, 2004.

[13] J.A. Barnes A.R. Chi L.S. Cutler D.J. Healey D.B. Leeson T.E. McGunigal J.A. Mullen Jr. W.L. Smith R.L. Syndor R.F.C.Vessot and J.A.Winkler. Characterization of frequency stability. *IEEE Trans. on Instr. and Meas., vol. IM-20,no.2*, May,1971.

[14] D.W. Allan. Statistics of atomic frequency standards. *Proceedings of the IEEE, vol 54, no.2*, July,1996.

[15] Snort. www.snort.org.

[16] O. Spatscheck M. Roughan S.Sen and N. Duffield. Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification. *IMC*, 2004.

[17] D. Zuev A. Moore. Internet traffic classification using bayesian analysis. *Sigmetrics*, 2005.

[18] C. E. Brodley. C. Rosenberg J.P. Early. Behavioral authentication of server flows. *ACSAC*, 2004.

[19] K. Papagiannaki M. Faloutsos T. Karagiannis. Blinc:multilevel traffic classification in the dark. *ACM SIGCOMM, pages 229-240*, 2005.

[20] Manuel Crotti Maurizio Dusi Francesco Gringoli Luca Salgarelli. Traffic classification through simple statistical fingerprinting. 2006.

[21] Codecs. Global ip sound. *http://www.globalipsound.com/partners*.