Department of Economics and IT

Degree Project
T08-117

Evaluation of Statistical Distributions for VoIP Traffic Modelling

Fredrik Gustafson
Marcus Lindahl

DEGREE PROJECT

Evaluation of Statistical Distributions for VoIP Traffic Modelling

Fredrik Gustafson
Marcus Lindahl

Abstract

Statistical distributions are used to model behaviour of real VoIP traffic. We investigate call holding and inter-arrival times as well as speech patterns. The consequences of using an inappropriate model for network dimensioning are briefly discussed. Visual examination is used to compare well known distributions with empirical data. Our results support the general opinion that the Exponential distribution is not appropriate for modelling call holding time. We find that the distribution of talkspurt periods is well modelled by the Lognormal distribution and the silence periods by the generalized Pareto distribution. It is also observed that the call inter-arrival times tend to follow a heavy tailed distribution.

Publisher:	University West, Department of Economics and IT, Box 957, S-461 29 Trollhättan, SWEDEN Phone: + 46 520 47 50 00 Fax: + 46 520 47 50 99 Web: www.hv.se		
Examiner:	Stanislav Belenki		
Advisor:	Stanislav Belenki		
Subject:	Computer Science	**Language**:	English
Level:	Degree project	**Credits**:	10 Swedish, 15 ECTS credits
Number:	T08-117	**Date**:	May 27, 2008
Keywords	statistical model, voip, call holding, speech pattern, call interarrival, telephony, probability distribution		

Preface

We would like to thank Stanislav Belenki for his guidance and for suggesting this degree project. We also thank Mats Lejon for helping us with the data capturing setup.

Most work has been carried out by both authors in unison. However, some topics have been studied in more detail individually.

Marcus Lindahl

- Distribution fitting in Matlab
- Construction of post processing scripts

Fredrik Gustafson

- VoIP background study
- Construction of scripts for automated data capturing

Table of contents

Abstract i
Preface ii
Nomenclature iv
1 Introduction 1
2 Voice over Internet Protocol 2
2.1 Data stream 2
2.2 Call control 3
2.3 Codecs 6
2.4 Telephone system architectures 7
3 Goal 11
3.1 Objectives 11
4 Delimitations 11
5 Method 11
5.1 Data capturing setup 12
5.2 Post processing 15
6 Result 17
6.1 Call holding time 18
6.2 Talkspurt and silent duration 22
6.3 Call inter-arrival 29
7 Discussion 30
8 Conclusion 32
References 33

Appendix

A Random variables
B Script for call length calculation
C Script to find the busy hour
D Script to calculate the inter-arrival time
E Script to calculate silence
F Script to calculate talkspurt
G Basic statistics
H AWK-filter for time conversion

Nomenclature

ATM – Asynchronous Transfer Mode

IPX – Internetwork Packet Exchange

VoIP – Voice over Internet Protocol.

RTP – Real-time Transport Protocol

PSTN – Public Switched Telephone Network

MGCP – Media Gateway Control Protocol

ISDN – Integrated Services Digital Networks

PRI – Primary Rate Interface

VAD – Voice Activity Detection

PBX – Private Branch Exchange

PSTN – Public Switched Telephone Network

SPAN - Switched Port Analyzer

PDF – Probability Density Function

Heavy-tailed distribution – Statistical distributions whose tail are not exponentially bounded

Talkspurt – An uninterrupted burst or 'spurt' of speech as interpreted by VAD

Silence – A period of silence during conversation as interpreted by VAD

Softphone – An IP telephone implemented in software

1 Introduction

Voice over IP (VoIP), IP-telephony or broadband telephony—they are all terms describing the same thing, namely transmission of the human voice over a data network, using the Internet Protocol (IP).

In contrast to the ordinary telephone system (PSTN), where every call has a dedicated connection between the callers, VoIP calls are sharing capacity not only among themselves, but often with other data as well.

As a direct consequence—the statistical properties of VoIP traffic needs to be known to successfully calculate the capacity necessary to achieve a certain quality of service.

With the introduction of VoIP, calls from a VoIP telephone to another is often not charged by the minute and therefore the call holding distribution is aught to change[1],[2]. Several research papers states that in the past, the call holding time distribution of any telephony voice service was thought to be Exponential [1,3,4]. However, later research rejects this belief and rather suggests that it follows a Lognormal distribution, or is a mix of two Lognormal distributions [3,4,5,6,7], while investigations made in [8] arrive at the conclusion that the generalized Pareto distribution provides the best fit for the call holding times.

Call inter-arrival is assumed to follow the exponential distribution [8],[9]. Trang Dinh Dang *et al* investigate this, and their finding supports the popular assumption that call inter-arrival times follows the exponential distribution [8].

Early researches on speech pattern generation, as that performed by P.T Brady, assumed that the talkspurt and silence periods in a telephone call follow an exponential distribution [10]. But in later research, performed by Trang Dinh Dang *et al*, the speech pattern is found to be rather following the generalized Pareto distribution [8]. Other distributions are suggested like in [10], where the Gamma- and Weibull distribution is proposed as a fit for the talkspurt and silence periods in speech, respectively. In [8] the Lognormal distribution is presented as the fit for modelling the speech pattern, and ITU-T propose the speech pattern to be modelled by their P.59 distribution [11]. However, Padungkrit Pragtong *et al* finds that it can not be used to model talkspurt, and for silence it fails to describe the longer tail of their collected data[12].

Regarding speech pattern, there are few—if any—studies where observations are based on Swedish as the spoken language.

In this report we use empirical data collected from the VoIP telephone system at University West and compare them to a set of well known statistical models to see which model best describes the speech pattern and call duration. We also investigate if the Exponential distribution fits the inter-arrival time of VoIP traffic. The matched statistical models are used both in VoIP, mobile telephony and in PSTN environments.

We also present a general introduction to VoIP.

This report is organized as follows: After the introduction section, the background and basic operation of VoIP is presented in section 2. Next, our goals and objectives are stated along with the delimitations of the study in section 3 and 4. This is followed by a detailed description of the methods used in our study in section 5, followed by a presentation of our findings in section 6. In section 7 and 8 we discuss and summarize our results. In appendix, we provide our scripts for data capturing and post processing along with the basic statistics of our results. Since call statistics is described as random variables, readers unfamiliar with this subject may want to read a short introduction presented in appendix A.

2 Voice over Internet Protocol

The story of VoIP started back in 1995 when a small Israeli company called Vocaltec released what is believed to be the first VoIP softphone, the 'Internet Phone'. The software ran on a PC much like the soft phones used today and utilized the H.323 control protocol for call signalling. The success of the Internet Phone was however brief due to the lack of broadband availability at that time which resulted in poor sound quality and interrupted calls [13].

By 1998, VoIP traffic represented approximately 1% of the overall voice traffic in the USA and a number of implementations were introduced such as PC-to-phone and phone-to-phone solutions. The technology growth was further helped by networking companies such as Cisco, Lucent and Nortel introducing equipment capable of routing and switching the digitized voice data [13].

The concept of 'Voice over Internet Protocol' consists of several components that together create a function optimized for transmission of voice through a packet switched network. VoIP carry voice data as digital audio typically using speech data compression and- or header compression to minimize bandwidth consumption.

2.1 Data stream

Basically a VoIP call consists of three parts; data streams, call control and codecs. The data is sent by RTP encapsulated in a data packet stream over UDP/IP. VoIP uses RTP/UDP for its data stream although UDP is a connectionless, best effort protocol since TCP with its retransmission and congestion control will introduce to much delay to the data stream. In addition to that – in any real-time application one would rather miss one packet than having the whole transmission halting to wait for a resent packet to arrive.

2.1.1 RTP

RTP is an Internet standard protocol, described in RFC 3550[14], designed for transporting real-time data such as video and audio over an internet. It is used in both video-on-demand and voice over IP applications. RTP consists of a data part, that supports timing reconstruction, security, loss detection and content identification for real-time applications, and a control part that adds support for source identification, quality of service feedback and synchronisation of different media streams [15]. Although RTP is often used with UDP/IP, it is designed to be platform independent and is intended to work with other protocols such as IPX and ATM. The standard RTP packet header (figure 1) is 4 bytes long.

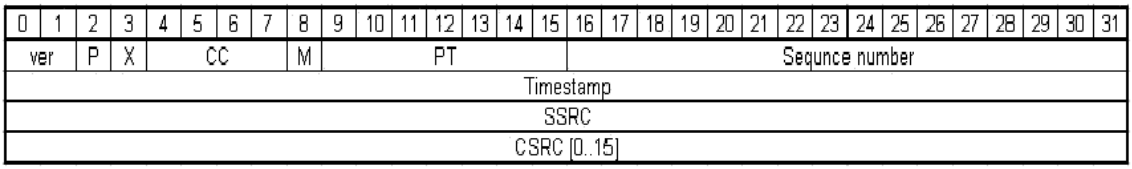

Figure 1: RTP header [16]

The 'ver' field is the RTP version number. The P and X fields are padding and header extension respectively. If the P field is set, it indicates that one or more bytes of padding are added to the packet. Padding is sometimes needed by encryption algorithms. The X field indicates that the packet header is followed by a header extension. The CC field contains the number of CSRC identifiers in the CSRC field and the meaning of the M field is defined by the profile used. A profile defines the default mapping of payload type codes to payload formats. The PT field identifies the payload type such as PCMA, G721 or GSM.

The value of the sequence number field is incremented for each packet in a data stream and used for packet loss detection and restoring packet sequence at the receiving application. The time stamp field is 4 bytes long and could be used for e.g. calculating jitter and delay. The SSRC field contains a unique identifier for the RTP session. If there are more than one source creating/generating the content of the payload their SSRC numbers are presented in the CSRC field in individual elements. In case of e.g. a voice conference the CSRC elements allows for correct talker indication at the receiver [16].

2.2 Call control

Call control, or signalling, is handled by a separate protocol like SIP, MGCP or H.323. These protocols differ in many ways but their main task is the same, to act as a director of call setup and teardown, to find a route to the called party and to setup the RTP data streams. Some of these protocols monitor the RTP streams during the whole conversation whilst others just handle the initiation and cancelling of the session. Below

is a short description of some common signalling protocols.

2.2.1 SIP

Development of the Session Initiation Protocol, SIP, started in 1999 by the IETF SIP group [17],[18]. For call control, SIP uses the Session Description Protocol [19], SDP, to describe the call details such as packet size, audio or video stream and codec type. It uses an URI, Uniform Resource Identifier to point to a logical destination rather than to an IP address. SIP operates according to the client-server model in that aspect that each SIP enabled device has a server called User Agent Server and a client called User Agent Client. These functional entities are able to send responses and requests respectively.

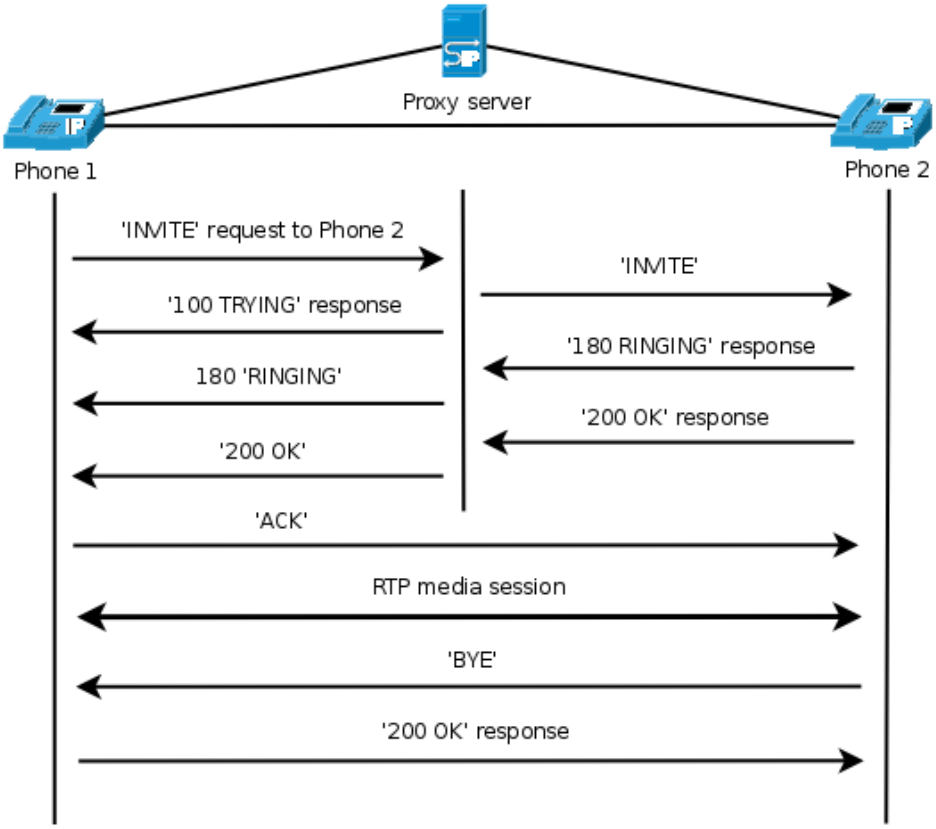

Figure 2: SIP call setup [20]

Above a simple call setup with SIP is described (figure 2). The setup consists of two SIP phones using a SIP proxy server to route calls to unknown destinations. Phone 1 starts by sending an 'INVITE' request to phone 2 including SDP data for the requested session type. Since it does not know the IP address of phone 2 it sends the signal to the proxy server that sends a '100 TRYING' response back to phone 1 letting it know that the proxy server is trying to route the call. The proxy sends the 'INVITE' request from phone 1 to phone 2 that starts to ring, notifying the user of an incoming call.

Phone 2 sends a '180 RINGING' response back to phone 1 through the proxy server and phone 1 also starts to ring, indicating that the call is being set up. When the user of phone 2 lifts the handset and answers the call the phone sends a '200 OK' response to phone 1 via the proxy. The '200 OK' message contains the SDP media description that tells phone 1 what type of media session that phone 2 is able to handle for the call. Recall that in the initial 'INVITE' from phone 1 the session type request was sent thus the '200 OK' SDP media description completes a two-way negotiation of the capabilities that are to be used in the call.

Phone 1 sends an 'ACK' message directly to phone 2 and the bidirectional RTP media stream is set up enabling phone 1 and 2's users to talk to each other. When the conversation is ended by any of the participants the terminating party's phone sends a 'BYE' request to the other part which responds with a '200 OK' message that terminates the RTP stream and the call is terminated[20].

2.2.2 H.323

H.323 is an ITU-T recommendation protocol suite that has evolved out of a video telephony standard in response to an industry call for an IP telephony standard [20]. H.323 consists of a set of entities – for VoIP the entities are gateways, terminals and gatekeepers. Gateways provide protocol translation and media transcoding between H.323 and non-H.323 end devices such as one terminal connected to a VoIP network and one connected to PSTN.

Terminals, or end devices, includes a System Control Unit using H.245 and H.225.0 for call control and an audio codec that transcode and compress speech. H.225.0 also performs logical framing and error correction of incoming data streams.

Gatekeepers perform admission control and address translation. Gatekeepers use two channels for administrative messaging – one of them is the RAS (Registration Admission Status) channel that carries gatekeeper messages used in endpoint registrations that serves to associate an endpoint's alias with its IP address and port number that should be used for call signalling. An alias could be a regular PSTN telephone number. The RAS channel is also used for admission, status and bandwidth change messages. The other channel used is the Call Signalling Channel, used for reliable transmission of H.225.0 call control messages over TCP [20].

2.2.3 Voice Activation Detection

A typical voice conversation contains from 35% up to 50% of silence [21]. Voice Activation Detection (VAD) is a method to save bandwidth by not sending packets over the network when there is no speech to transmit. This method can reduce bandwidth consumption with up to 35%, however, by using VAD the audio quality gets slightly degraded [22]. In order to not having the callers mistaking a silence period for a disconnected call a comfort noise is generated locally by the phones, making the call

appearing to be still in progress.

2.2.4 MGCP

MGCP, Media Gateway Control Protocol, is a protocol for controlling gateways from a centralized unit called Call Agent and is defined in RFC 2705 [22],[23]. The MGCP gateway handles audio translation between a circuit-switched and a packet-switched network, i.e. from a PSTN to an WAN or LAN, or the other way round. An MGCP gateway could be e.g. an access or residential gateway. In MGCP, endpoints represent the point where the packet-switched and circuit-switched networks interconnect. The set up of an end-to-end call with MGCP is performed as follows: When establishing a call the Call Agent tells the two gateways associated with the endpoints to create a connection to the each others endpoints. The calling endpoints gateway sends back its call session parameters, which are then forwarded to the gateway of the called endpoints. In that way, both endpoints gateway have the parameters needed to set up the RTP sessions between the caller and the callee. The connections associated with this call will have the same call ID using the same media stream. When an interlocutor terminates the call, the Call Agent sends a 'delete connection' request.

2.3 Codecs

Codecs are used to compress and encode voice data in order to optimize bandwidth utilization. The codecs reside in the IP telephones or softphones. There are several levels of encoding, each with increasing effects on end device or gateway CPU load, voice quality and delay. Clearly, there need to be an engineering trade-off between these factors. In addition, the payload size generated by encoding is a matter of trade-off in terms of payload efficiency. A standard IPv4 RTP/UDP/IP header is 40 bytes and a payload of 40 bytes would mean an overhead per packet of 50%. To reduce the large overhead, header compression can be used to compress the headers to a size of ~2-7 bytes (figure 3). The time it takes to compress the data induces a delay, that adds to the overall round-trip time[20].

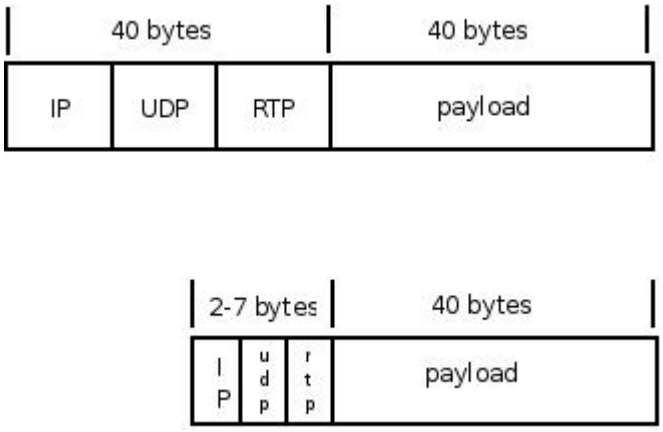

Figure 3: Header compression

2.4 Telephone system architectures

2.4.1 Circuit switched telephone system

In a circuit-switched telephone system set up, as in figure 4, all phones are connected to a PBX. The PBX in turn is connected to the service provider through a trunk.

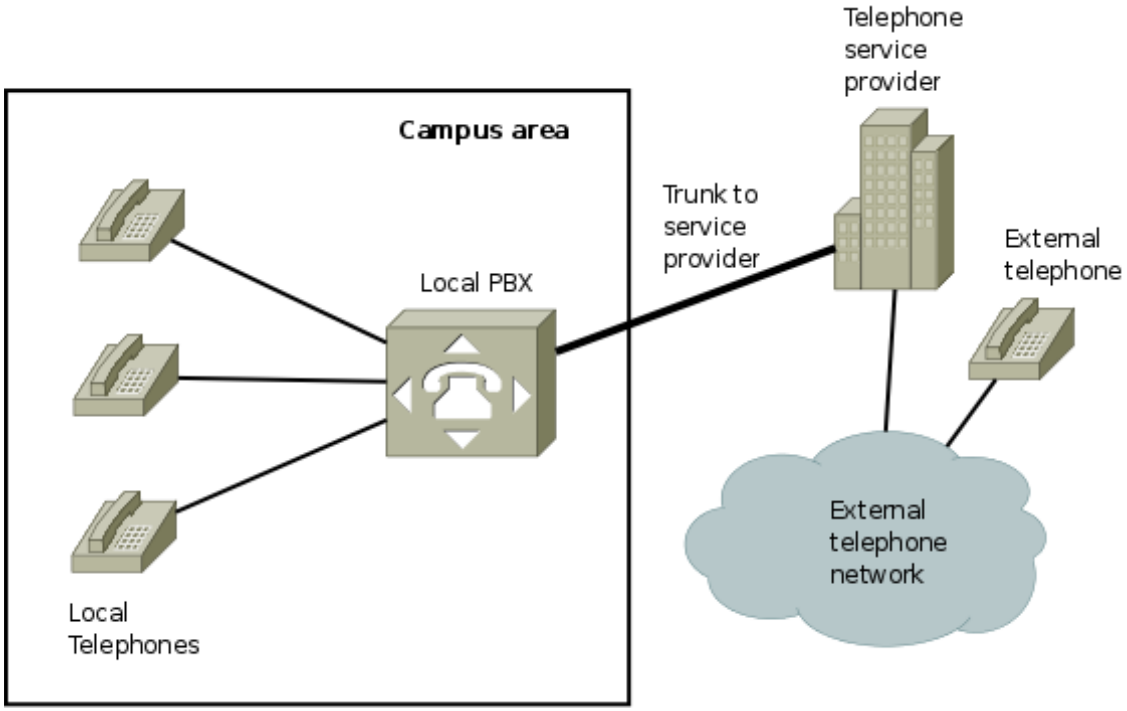

Figure 4: Basic PSTN example

A PBX is used to switch calls within the local area of the campus. It is similar to the switch used by a telephony service provider, but it is a smaller privately owned version. By connecting all campus phones to the local PBX there is only need for a single connection—called a trunk—to the service provider. This trunk is shared by all outgoing and incoming calls to the campus. This is a very cost effective solution since every phone on campus otherwise would have needed a separate connection to the telephone company [22].

The basic principle for establishing an end-to-end call in this set up is illustrated in figure 5.

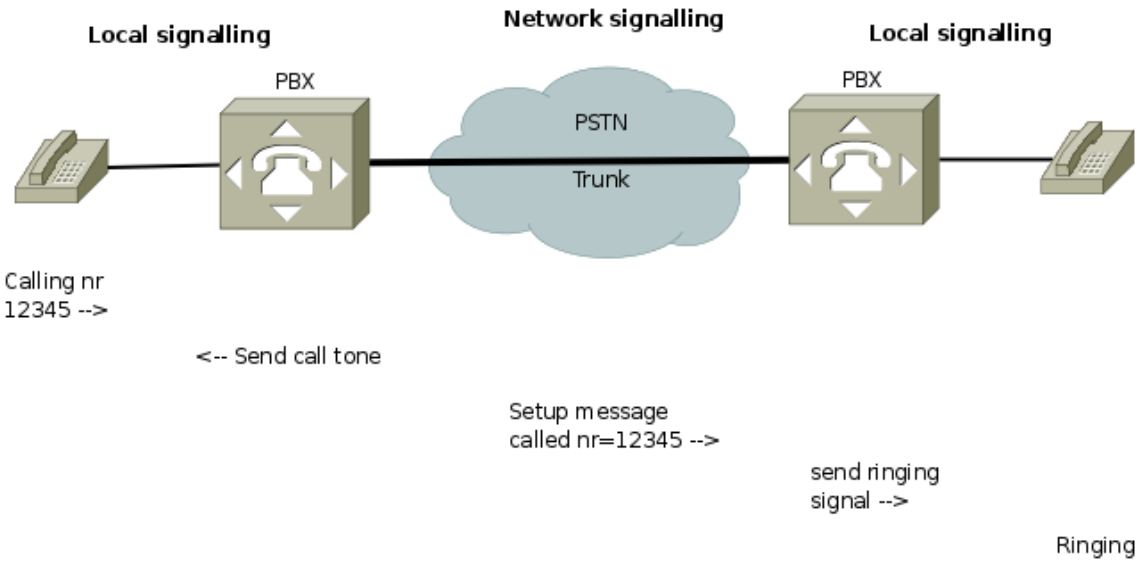

Figure 5: Basic PBX call setup

In the initial state the calling phones handset are lying in their cradles and the circuit is open, hence no current flows between the phone and the PBX. This state is called 'On Hook'. When the calling party is preparing to make a call the handset is lifted of it's cradle and the circuit between the switch and the phone is closed, allowing a current to flow through the wire. This is called the 'Off Hook' state.

The current notifies the telephone company that a request for call is about to be made and the telephone network provides a dial tone, indicating that it is ready to receive the request. When the calling party dials a number, the telephone company sends a voltage to the ringer of the called party's phone that is always in an idle state waiting for the voltage telling it to ring, notifying the called party of an incoming call.

The telephone company also sends a ring-back tone back to the caller to notify him that the call setup is in progress. Once the called party lifts his handset from the cradle, a dedicated circuit is set up between the two phones [22]. The signalling between the PBX's over the telephone network is called network signalling, and the signalling taking place between the local PBX and the end user phone is called local signalling.

2.4.2 Hybrid telephone network

As mentioned earlier, VoIP is all about sending digitized voice in packets, and the telephones used for making the calls can be either analogue or digital. The phones are able to digitize, encode and packetize the data [20]. Figure 4 describes a standard telephone set up with a PBX on each site to handle local calls. In figure 6 the set up has been extended to contain an IP gateway on each of the two sites, connected to a corporate WAN or the Internet.

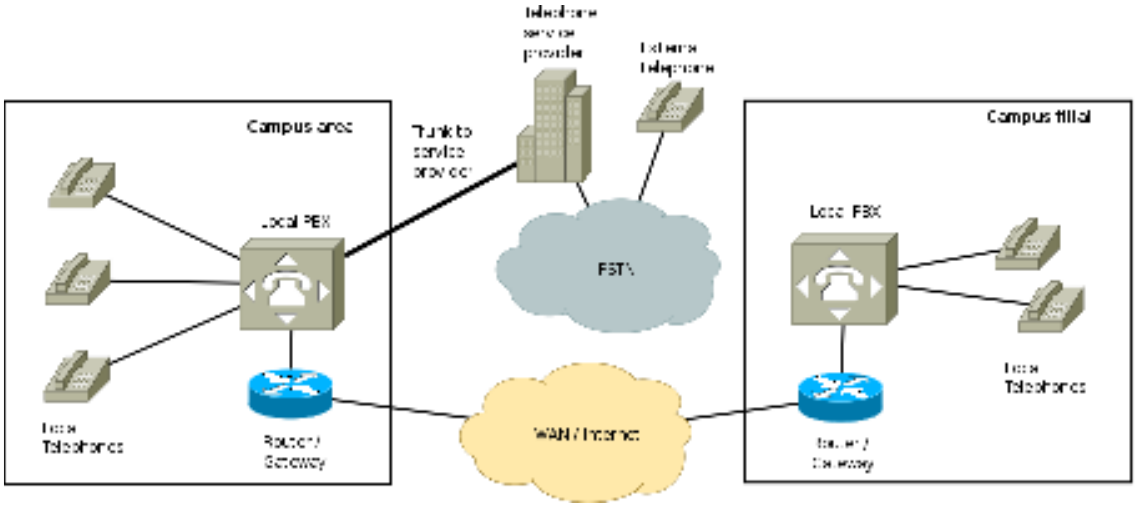

Figure 6: PBX / VoIP example setup

The PBX's on each site connects to the gateways enabling calls to the campus filial to be sent over the packet switched network, allowing low cost long distance calls to be made between the campus areas. At the same time the trunk to the telephone service provider is kept for calls destined to external phones not part of the corporate WAN or not able to receive IP telephony calls. The choice of which technology to use for outbound calls can be determined by algorithms based on e.g. lowest cost. Local calls within the campus area or filial are still switched by the local PBX.

2.4.3 VoIP network

The technology that will probably dominate the future IP telephone networks is shown in figure 7. Here the PBX is removed and the telephones are replaced with digital IP phones, or soft-phones running on standard PC's, all connected to a network switch

forming a LAN. The IP phones are capable of digitalisation and packetization of voice

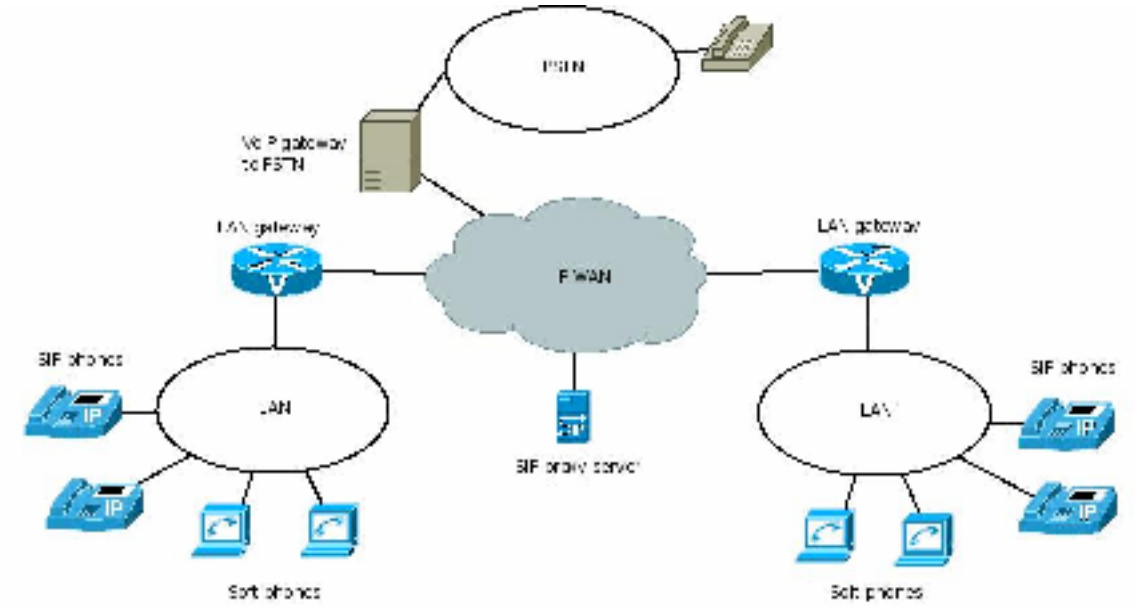

Figure 7: VoIP WAN / LAN setup

data before sending it out on the link.

The IP phone LAN is attached to a router or gateway capable of routing VoIP calls. Like the PBX this router handles calls within the local area (LAN) but also routes outbound calls to the appropriate destination. Calls between sites are carried over the WAN link or tunnelled through the Internet and calls to PSTN are routed to a PSTN gateway. The advantage of this set up is that it can be implemented on an existing LAN and nearly any new network infrastructure could be used to carry VoIP traffic i.e. there is no need for a separate physical phone network.

In figure 7, SIP is used for call control. IP Phones register with a SIP proxy server that also takes care of the call signalling. By acting as a middleman for call setup and teardown the proxy acts in some ways much like a PBX. If someone wants to make a call outside the LAN the SIP proxy handles the 'INVITE' message from the calling party, notifies the called IP phone and sets up the RTP data stream used for transmitting the digitized voice. The SIP proxy set up in figure 7 could be altered with a proxy on each of the sites or LAN's. Then the two proxies would relay signalling messages between the phones.

The task of providing reliable, high quality voice communication over a network designed for data transmission is complex. There is a clear advantage in being able to simulate traffic behaviour prior to implementation. Accurate simulation tools however need to be based on appropriate models.

3 Goal

The goal of this project is to find what well-known models that accurately describe the observed speech patterns, call holding- and call inter-arrival times.

3.1 Objectives

- Develop methods to capture data
- Capture voice traffic data on University West network
- Develop methods to derive relevant statistics from captured data
- Match statistics to chosen models
- Investigate consequences on network dimensioning

4 Delimitations

Concerning call holding- and call inter-arrival times, this study limits itself to analyse VoIP traffic to and from University West, the internal traffic flow is not examined. The campus working day spans from 8.00 am to 4.30 pm [24]. Therefore, the capturing is limited to these hours with one additional hour added before and after to capture stray calls.

5 Method

This project aims at validating several statistical models against empirical data. The data used in this study has been captured during 14 working days.

The analysis is based upon visual comparison of graphs to see the grade of correspondence with the empirical data. We also draw conclusions on the distribution fitting based on probability plots for the candidate distributions [25]. The analysis is performed by importing the data derived from the post processing into Matlab where the distribution fitting toolbox is used to find the parameters of the matched distributions [26].

In order to collect and analyse data, several methods where considered.

The VoIP installation at University West consists exclusively of equipment from Cisco systems, where two Cisco callmanagers (one for redundancy) are controlling all VoIP telephones. The callmanager logs a vast quantity of statistical data in Call Detail Records (CDR), regarding calls performed over time. Call inter-arrival and call holding can be extracted from these CDRs. Using CDRs as source for the statistics has the advantage that the time series truly reflects the actual traffic conditions, and that post processing of data would be fairly easy. Sadly, the CDRs could not be accessed within the time frame of this study. Instead, statistics are based on data packets captured in the

live network.

In order to obtain the speech pattern of a voice data stream, Voice Activation Detection (VAD) needs to be enabled on the network. Unfortunately, VAD is not used on the campus network and the speech pattern analysis cannot be carried out on data captured from it. Instead, an experiment is performed and the data from it is used for analysing the speech patterns.

There are many commercial hardware and software solutions on the market that can derive voice data statistics, either by capturing and processing data in real-time, or by post processing captured data [27],[28]. However, commercial solutions all have in common that they cost money, and this study is on a zero budget, making this an unviable option. Instead, a combination of freely available tools is used to capture data and derive the wanted statistics.

5.1 Data capturing setup

5.1.1 Call holding time and inter-arrival times

Because the campus network is a switched Ethernet environment, there is no single point that all voice traffic traverses. Therefore, as advised by the system administrator at University West, the data used for studying the call holding time and inter-arrival is captured in the network right behind the media gateways that forwards and receives calls from campus to the PSTN. The reason for capturing data at this location is its aptness to receive a lot of traffic.

Figure 8 shows the capturing setup in the university network. There are about 300 IP-Telephones connected to the Campus VoIP LAN. The Media gateways are connected to the PSTN via two ISDN PRI connections, each supporting 30 telephone calls, hence giving a total capacity of 60 concurrent calls. The Call Agent is responsible for setting up and tearing down telephone calls between the gateways and the telephones. The outgoing traffic is load balanced between the gateways.

A standard PC with a 100Mbit network card is used for data capturing, it is attached to the SPAN enabled switch. The data capturing is automated by a script that run Tcpdump [29] between 07.00 and 18.00 hours Monday to Friday, thus giving a specific time range for the captured data.

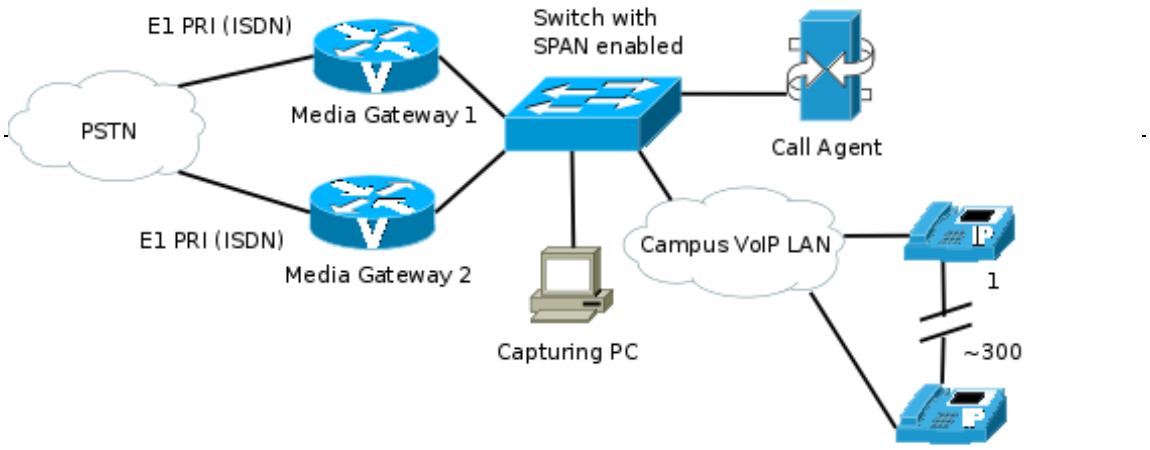

Figure 8: Data capturing setup on University West

5.1.2 Speech pattern

Because of the absence of VAD in the University West network, a different approach is taken to extract the talkspurt and silence periods from a VoIP conversation. Figure 9 shows the procedure used for generating the data. Using the Sound Recorder [30] in Ubuntu/Gnome for capturing microphone input, a total of 12 hours of speech sessions over a popular softphone was recorded. Since both sides of the conversation were captured, the result is two files of 12 hours length. These files were then merged into a 24 hour wave file and converted to PCM 16 bit sample size and 8kHz sample rate using SoX[31], thus the file contains 24 hours of unidirectional speech. As found in [32] both side of a conversation has the same characteristics, hence both sides are equally suited to be modelled by the same model. Therefore, it is viable to consider the two merged speech recordings as a 24-hour unidirectional conversation.

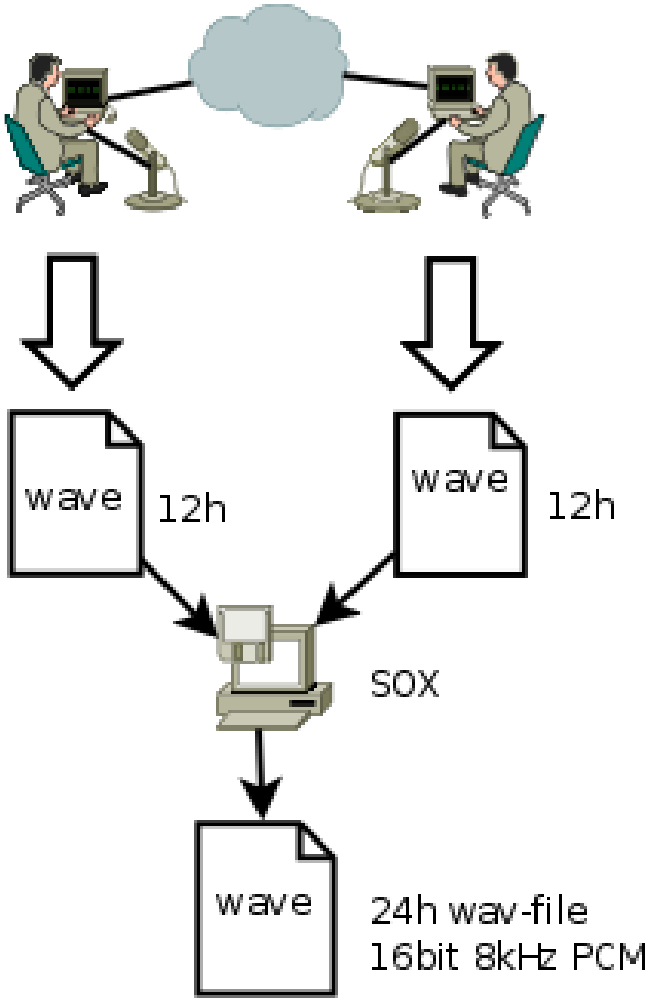

Figure 9: Voice session recording setup

The files are then replayed and encoded from node 1 to node 2 (figure 10) using OpenH323 [33] and Pwlib [33] as done by Arkadiusz Biernacki in [34]. The calling computer, node 1, uses simph323 [35], which is a simple IP soft phone able to make calls to another softphone. Node 2 uses oh323tut [36] that is started with the wave-file as an argument. When node 1 places the call, the call is setup with openH323, and oh323tut answers by playing the wave-file. Thus the 24 hours of recorded speech is replayed. This approach is taken based on the setup in [34]. A third computer, connected to a SPAN enabled switch port, captures the traffic between the other two with tcpdump capturing only UDP traffic on port 5000 which is the default port that openH323 uses for the RTP data stream.

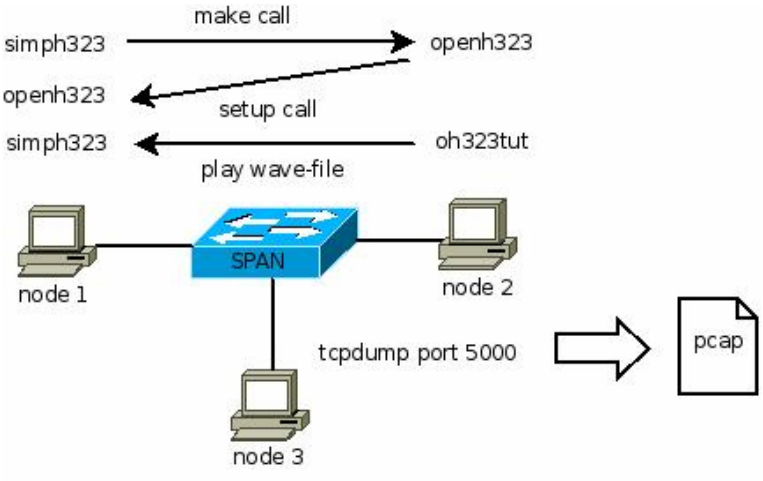

Figure 10: Speech pattern capturing setup

5.2 Post processing

The post processing procedure handles the task of deriving the relevant statistics out of the collected raw data files. There are three categories of statistics that is to be collected including call holding times, speech patterns - both talkspurt and silence measurements - and inter-arrival times. This is achieved by using a number of open source tools and the respective process is described below.

5.2.1 Call holding time

Initially Wireshark was the candidate program to derive the call holding time by processing the captured data [38]. By using the statistics function in Wireshark it is possible to detect the call holding times, based upon the MGCP signalling.

Unfortunately, Wireshark quickly runs out of memory and terminates when handling large capture files [38]. The typical file size for a day of capture is about 9.5 GB, and Wireshark—as of version 0.99.6—cannot handle this, but by using Tcpdump to extract only the MGCP signalling from the large capture file, the file size can be reduced (~10Mb) to Wiresharks ability. However, Wireshark lacks export functionality for the call holding times, so it cannot be used for this task either way.

Due to the shortcomings of Wireshark, Rtpbreak is used instead to derive the call length [39]. It does so by measuring the length of the RTP-streams that are present in the capture file. Rtpbreak works independent of the signalling protocol, and can therefore detect any RTP-stream regardless of the protocol that set up the stream.

A normal VoIP-call consists of two RTP-streams, one carrying the voice from the caller

to the interlocutor and one doing the opposite. By coherently filtering out only one direction of all RTP-streams detected—and measuring their length—the call duration is acquired.

Rtpbreak is instructed to only consider RTP-streams that originate from the Media Gateways that are the rendezvous points where all calls leave and enter the campus network.

The scripts used for processing call-holding data can be found in appendix B.

5.2.2 Call inter-arrival time

When VAD is not implemented—which is the case for VoIP traffic at University West—the marker bit is set to one on the first RTP-packet in the stream—that is, representing the start of a call—and to zero for the rest of the packets[40]. Consequently, by collecting all packets with the marker bit set, and calculating the time difference between these packets; the call inter-arrival times is obtained.

The packets with the marker bit set is extracted from the capturing file and Tshark is then used to derive statistics over the number of calls on a per hour basis, in order to find the busy period[41].

When the busy period is found, it is extracted with Tcpslice[42]. By using Tshark and examining the time differences between the packets, the call inter-arrival times are obtained.

The scripts used to derive the busy hour and the call inter-arrival times are found in appendix C and D.

5.2.3 Speech pattern

The beginning of a talkspurt period (figure 11) is identified by the marker bit (Bt1) that is set to one in the RTP-header, in all other packets, during the same period, the marker bit is set to zero[40]. When the talkspurt period is over (Et) there is no audio data to send, and the datastream to the receiver is halted.

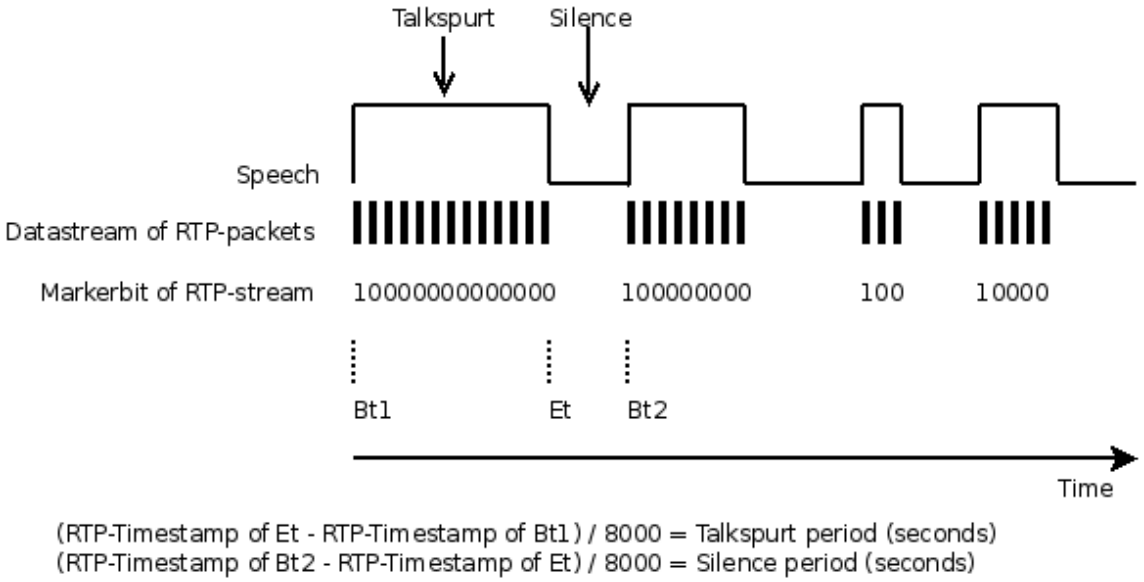

Figure 11: Identifying the talkspurt and silence periods in an RTP-stream

When a new talkspurt period begins (Bt2), the datastream resumes and the marker bit is once again set to one, the previous scenario then repeats itself for every talkspurt period in the call.

The calculation of the talkspurt length (in samples) is done by subtracting the RTP-timestamp from the first packet (Bt1) in the talkspurt period from the RTP-timestamp in the last packet (Et) in the talkspurt period.

The length of the silence period is calculated by subtracting the RTP-timestamp of the last RTP-packet seen from the sender (Et) from the RTP-timestamp in the next talkspurt period (Bt2).

By extracting the RTP-timestamp and marker bit from every RTP-packet in the capture file produced in the experiment (described in section 2.1.2), the length of every talkspurt and silence event can be derived.

The timestamps—that are in the format of samplepoints—can be converted into seconds by dividing with the samplerate (8 kHz).

The scripts used to derive the statistics from the experiment are found in appendix E and F.

6 Result

This section presents the empirical data compared to the chosen statistical models.

Matlab is used to find the appropriate parameters for the models and produce the graphs. The results are displayed by category with a probability density graph and the distributions that visually fit the empirical data best are treated as candidate distributions and are further examined in a probability plot.

The basic statistics for the collected data is available in appendix G.

6.1 Call holding time

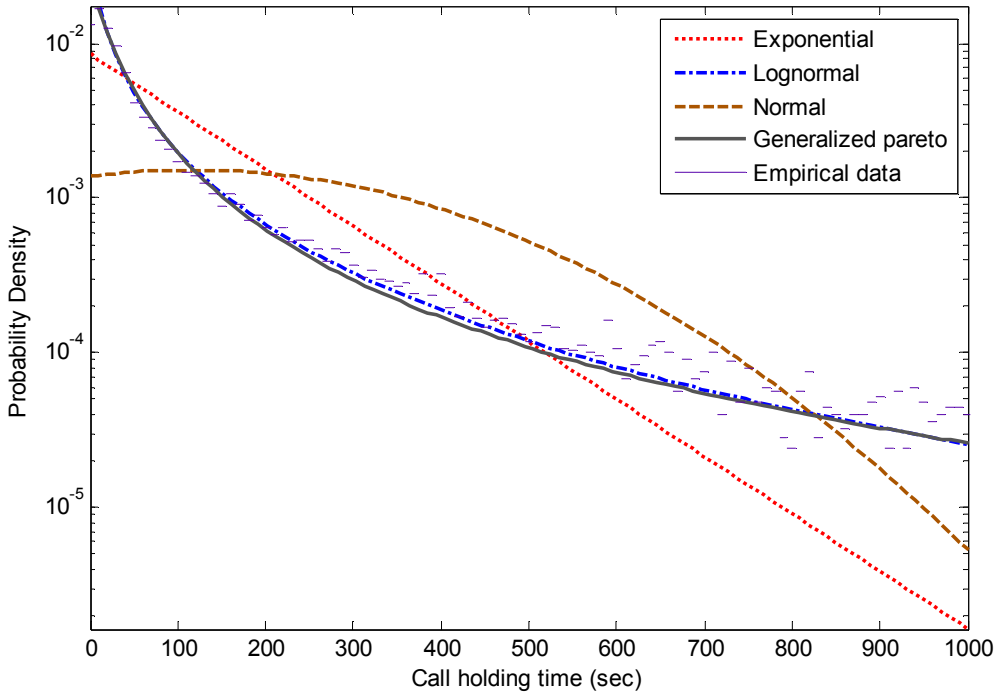

Figure 12: Call holding time PDF

The results of the matched distributions (figure 12) are shown with a logarithmic Y-axis and a truncated linear X-axis spanning to 1000 seconds. We motivate the truncation with the fact that there are few calls longer than 1000 seconds, in fact 99% are shorter than 1200 seconds and 98% are shorter than ~920 seconds. As observed in the PDF graph (figure 12) the Lognormal and generalized Pareto distribution visually fit the observed data best. These will be selected as the candidate distributions for call holding. For comparison reasons we also submit the Exponential distribution since it is classically used for modelling call holding times.

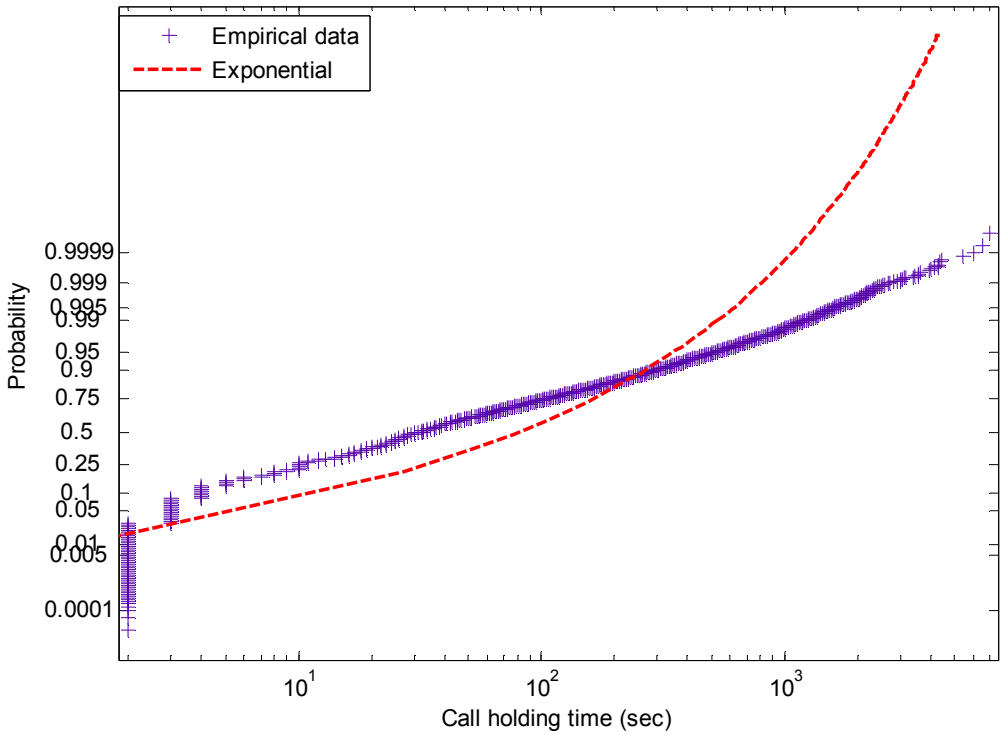

Figure 13: Call holding time: Exponential distribution probability plot

The probability plot (figure 13) for the Exponential distribution shows that it fails to accurately model call holding times, as seen in the PDF (figure 12) it decays to early and does not follow the heavy tailed characteristics of the actual call holding time, thereby underestimating the probability of longer calls.

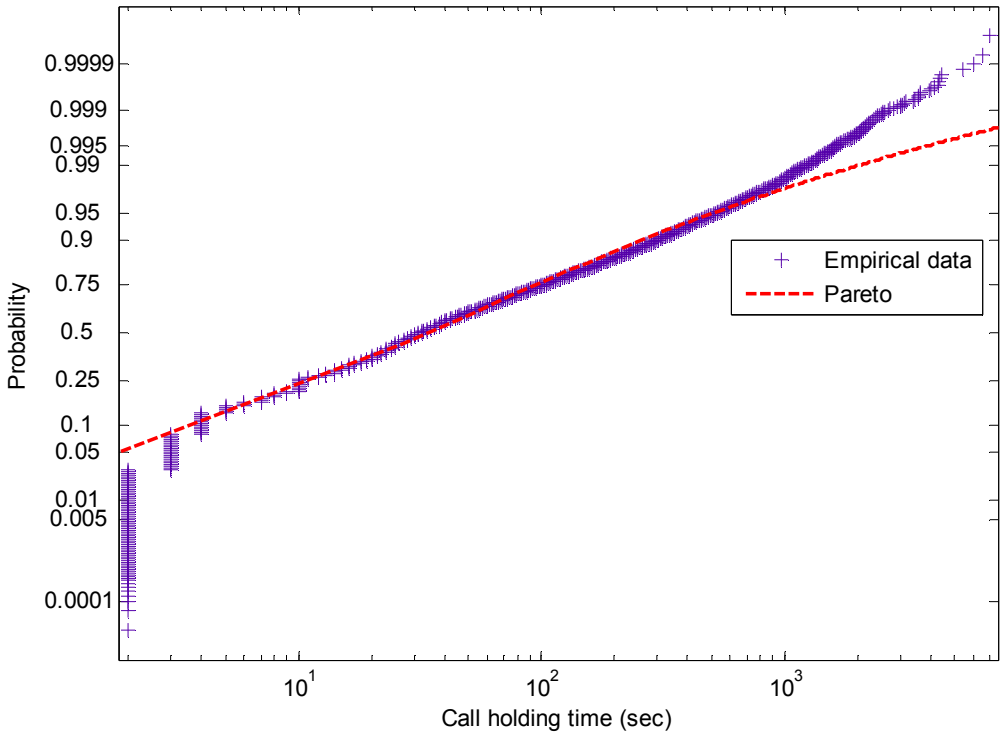

Figure 14: Call holding time: Pareto distribution probability plot

As the probability plot on the Pareto distribution (figure 14) shows, the Pareto is a good fit up to ~95% although it overestimates the number of very short call holding times.

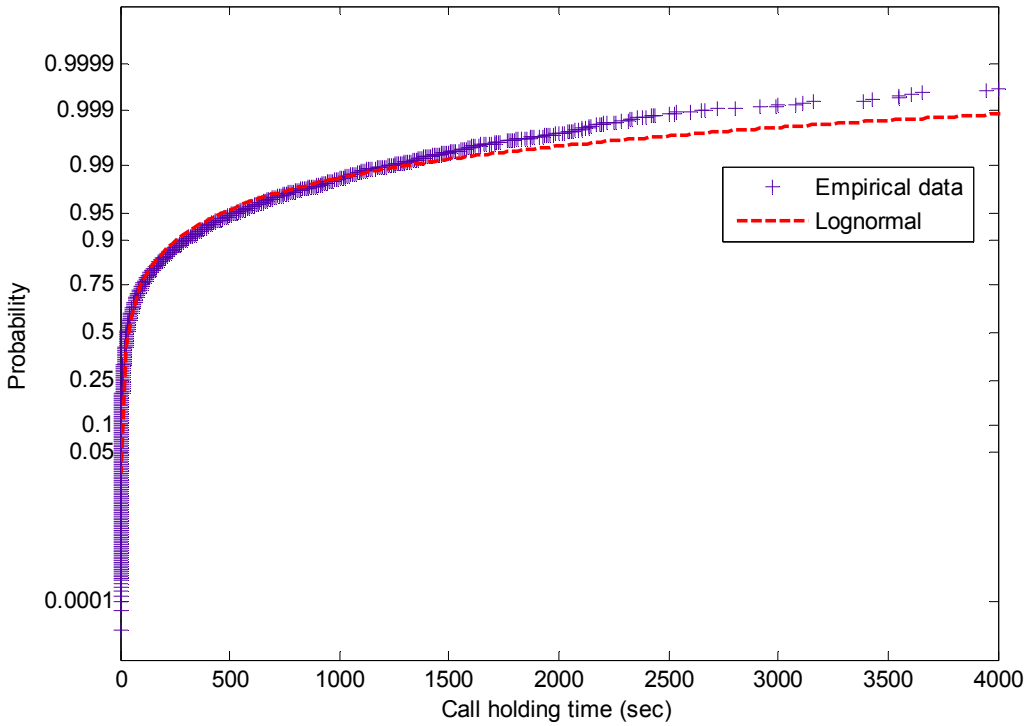

Figure 15: Call holding time: Lognormal distribution probability plot

The Lognormal distribution probability plot (figure 15) fits very well up ~99% and only deviates slightly for the call holding times longer than 1800 seconds (30 minutes). Observe that in this plot the X-axis is linear and truncated to 4000 seconds. In our traces there are only 10 calls out of the total calls collected (see appendix G) that are longer than 4000 seconds thus motivating the limitation of considered call holding times.

6.2 Talkspurt and silent duration

6.2.1 Talkspurt

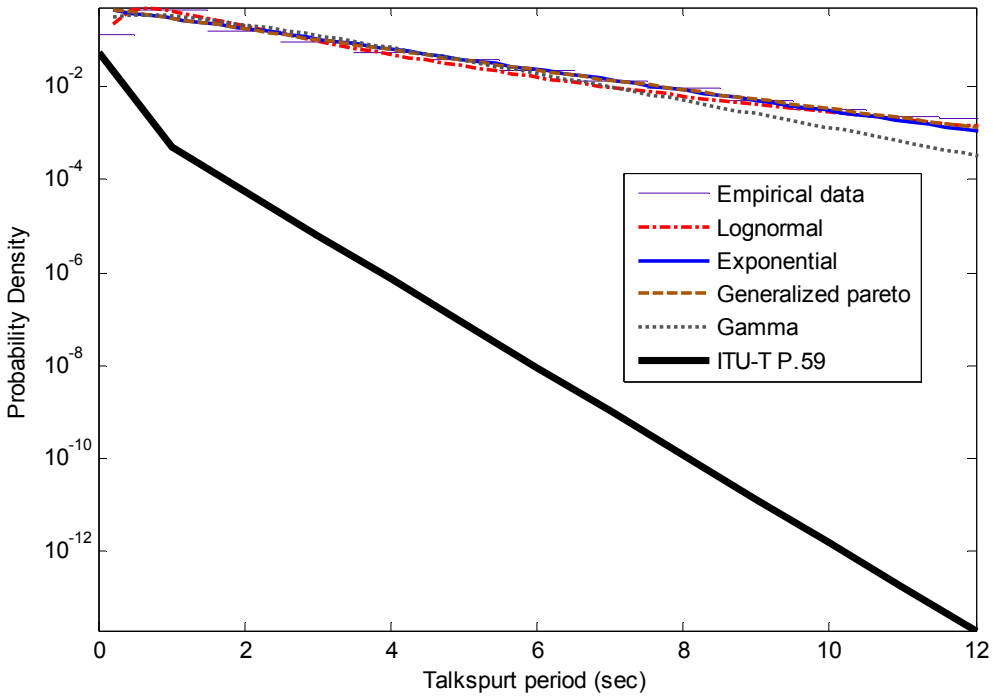

Figure 16: Talkspurt period PDF

In the talkspurt PDF (figure 16) the linear X-axis has been truncated to include only periods shorter than 12 seconds. This is motivated by the fact that 99% of all talkspurts are shorter than 10 seconds.

The ITU-T distribution is matched because it is a standard recommendation for generating artificial speech [11]. In this case, the ITU-T P.59 consists of two weighted geometric PDFs. The PDF of the matched distributions for talkspurts (figure 16) shows that the ITU-T P.59 distribution does not follow the actual talkspurt data at all. The Gamma distribution referred to in [10] cannot model the longer periods of talkspurt but the generalized Pareto, Exponential and Lognormal distributions seem to fit quite well. These are selected as our candidate distributions.

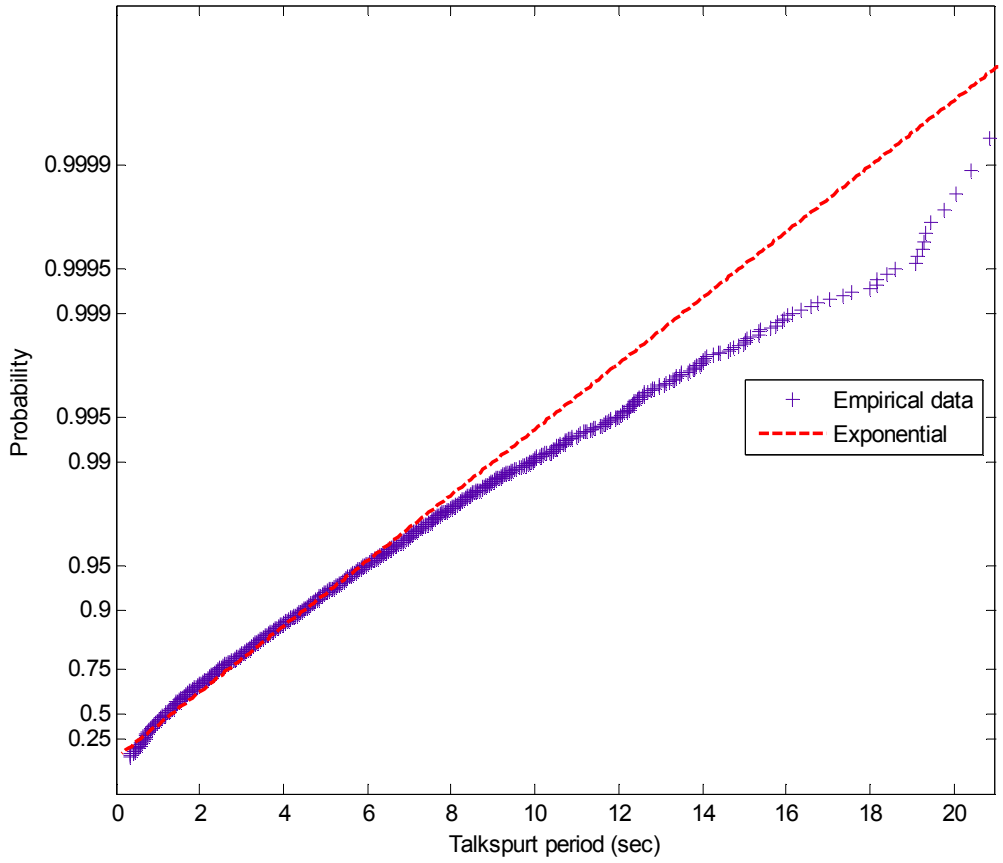

Figure 17: Talkspurt period: Exponential distribution probability plot

The probability plot for Exponential talkspurt distribution (figure 17) shows that for periods shorter than ~7 seconds the Exponential model fits quite well. As previously stated, 99% of all talkspurt periods derived from our data is 10 seconds or shorter. Most of the tail of the trace data—with some exception—would thereby be located below this limit. With this in mind, we see that the Exponential model fails to estimate the heavier tail of the empirical data distribution in an interval that can be of importance.

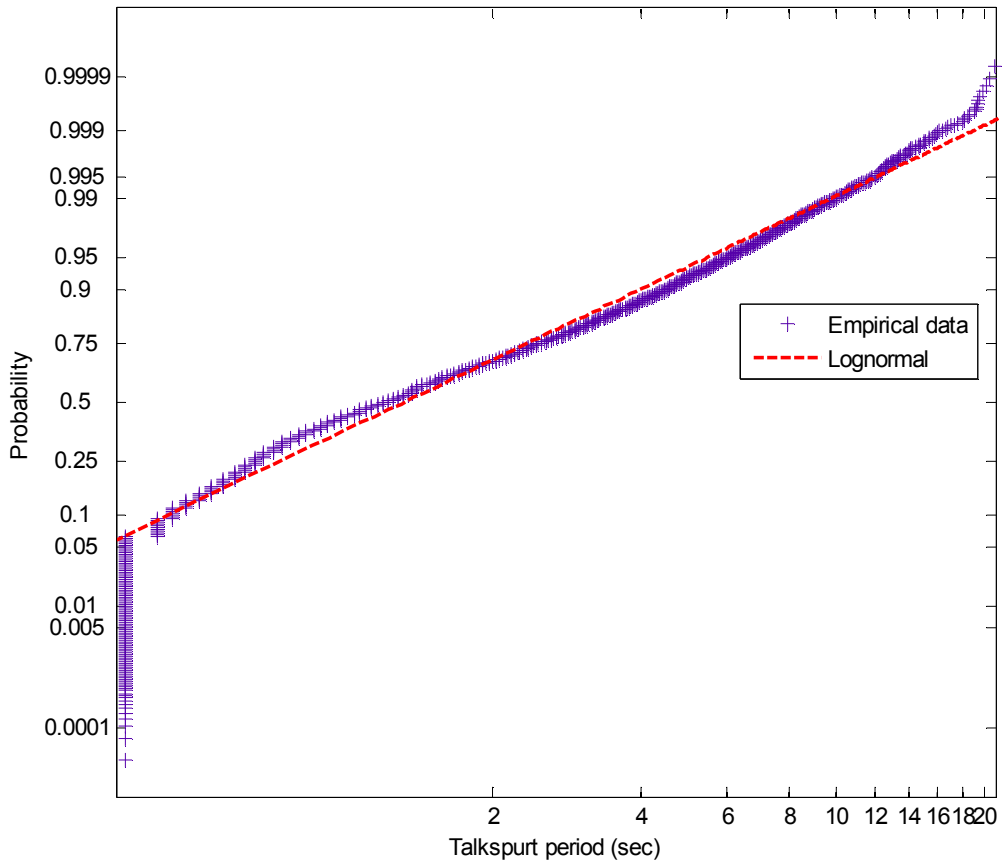

Figure 18: Talkspurt period: Lognormal distribution probability plot

Studying the Lognormal distribution probability plot (figure 18) we find that the Lognormal distribution fits very well even for talkspurts as long as ~15 seconds. Thus, it succeeds to model the significant part of our talkspurt data.

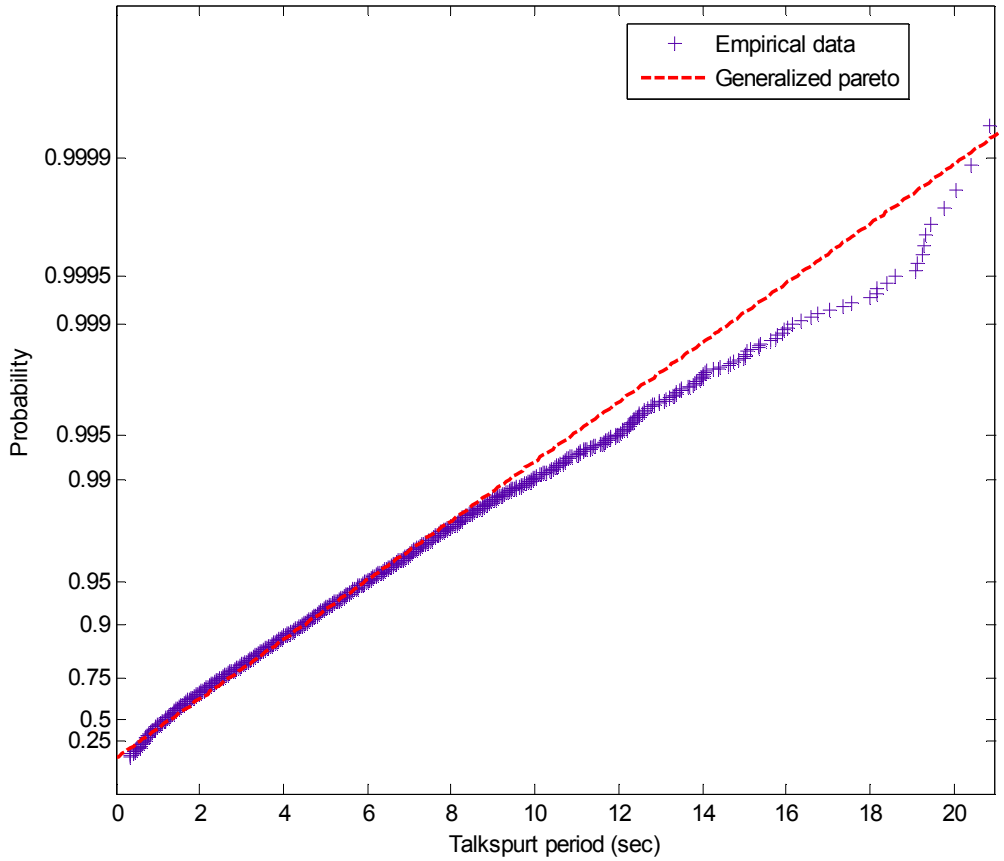

Figure 19: Talkspurt period: Pareto distribution probability plot

The probability plot for the Pareto distribution (figure 19) matches the trace data up to periods of ~9 seconds and then deviates much like the Exponential model.

6.2.2 Silence

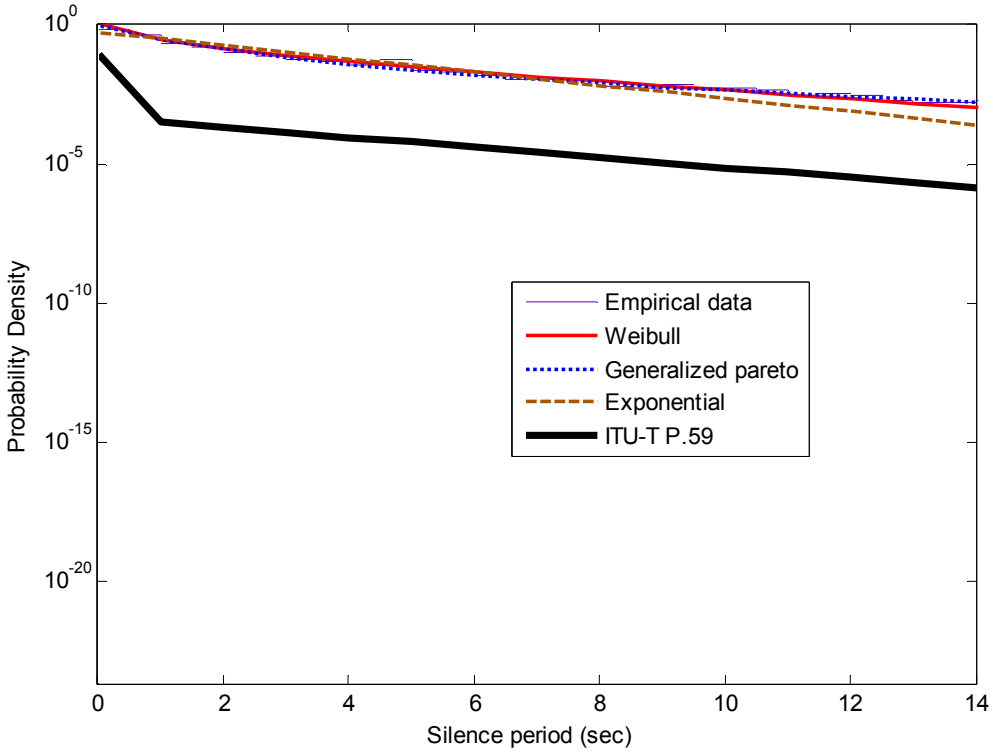

Figure 20: Silence period PDF

The silence period PDF describes the distribution fit up to 14 seconds (figure 20). 99% of all silence periods are shorter than 14 seconds. When looking at the silence period distribution the only model that is near to fit is the generalized Pareto. The ITU-T P.59 model fails completely—as in the case with talkspurt—to model the silence periods. The Exponential distribution fails at approximately the same period length as for talkspurt. This leaves Pareto and Weibull as candidate distributions.

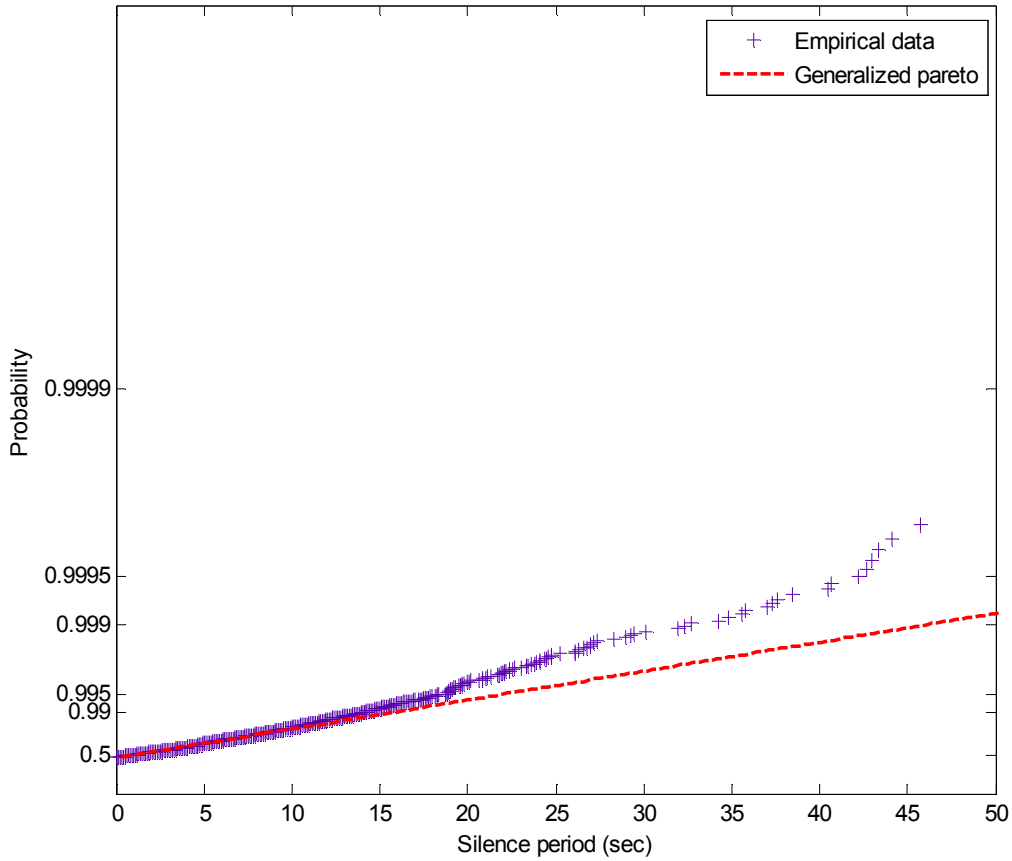

Figure 21: Silence period: Pareto distribution probability plot

The probability plot for the silence period (figure 21) shows that the Pareto distribution matches silence periods up to approximately 17 seconds. This is a good fit considering that 99% of the silence periods are below 14 seconds.

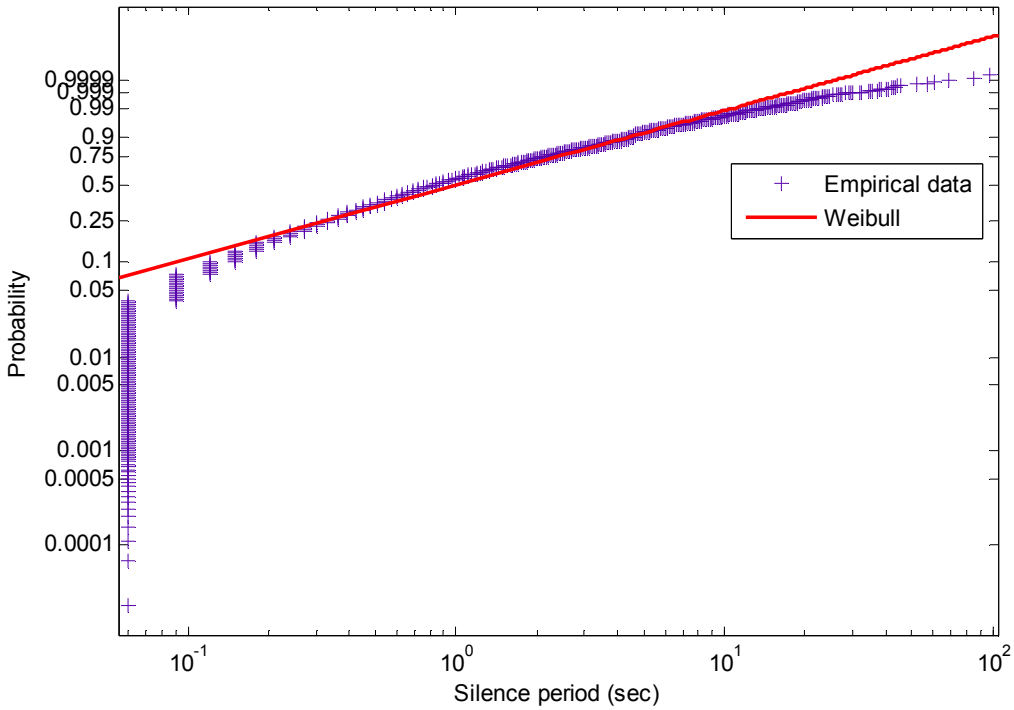

Figure 22: Silence period: Weibull distribution probability plot

The probability plot for Weibull (figure 22) shows that the distribution tends to slight overestimate the shorter periods. For the rest of the interesting interval up to, and past, 14 seconds it fits well.

6.3 Call inter-arrival

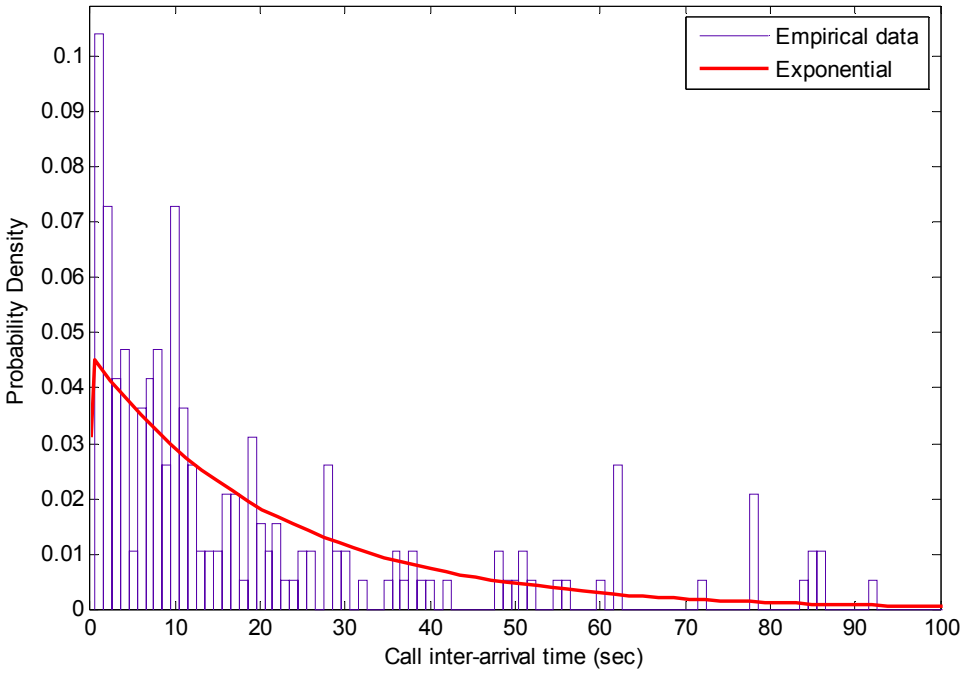

Figure 23: Call inter-arrival time PDF

The call inter-arrival PDF (figure 23) exhibits a linear-linear scale with a truncated X-axis that stretches to an inter-arrival time of 100 seconds. This is done because 99% of the inter-arrival times have proved to be under ~110 seconds and 95% of the data lies under 80 seconds limit not leaving that many calls in the interval over 100 seconds thus motivating the truncation. As the PDF (figure 23) shows, the exponential distribution fails to accurately model the very short inter-arrival times and it decays to fast, failing to estimate the more heavy tail of the inter-arrival data.

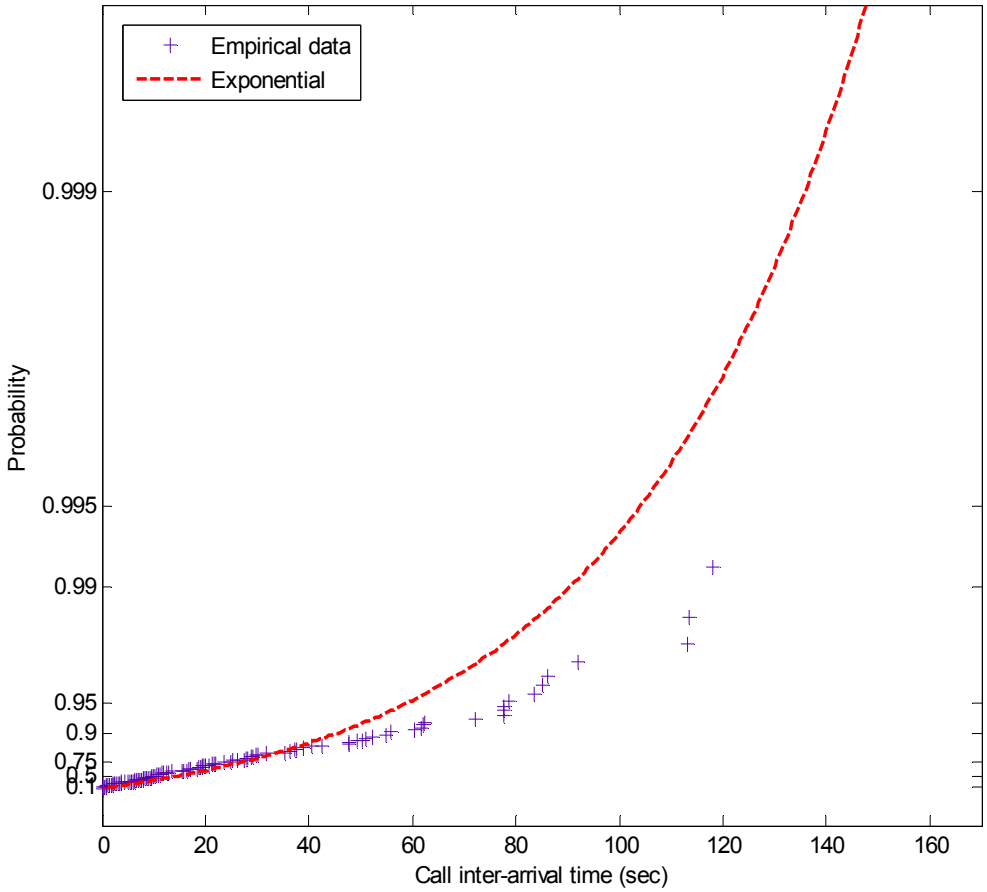

Figure 24: Call inter-arrival time: Exponential distribution probability plot

Studying the probability plot (figure 24) for the exponential distribution of inter-arrival times we see that it actually fits well up to an inter-arrival time of 40 seconds.

7 Discussion

The call holding times are found to match the lognormal distribution, another close fit is the Pareto distribution. For talkspurt and silence, the lognormal and generalized Pareto distribution fits respectively. The Exponential distribution fails to model the observed call inter-arrival times. Our results show that for all studied properties of our data the heavy-tailed distributions are the one that gives the best fit.

Call holding times have been thought to be accurately modelled by the exponential distribution. However our study shows that this model is not the best to use for this purpose, this is also found in [1,2,3,4,6,8]. In [10] however it is stated that the Exponential model still works. In [1,4,8,10] all studies are performed on VoIP or PSTN while [3],[6] perform their research on cellular networks. Although the values of call holding times differ between these types of network, partly based on traffic fees [2],

there seams to be an overall trend that denies the old belief that call holding times could be modelled with the exponential distribution.

Instead our results, as in [3,4,7] and in contrast to [8], suggests that the Lognormal distribution is best suited to model call holding times. Although V.A. Bolotin [4] states that the lognormal distribution only fits well when compared to a single subscriber line, we find that it also appropriately models the universities 60 lines. Further [4,5,6] finds the Lognormal distribution to be a good fit but suggests a combination of two Lognormal distributions for better accuracy.

The Pareto distribution comes next in preference even though it tends to overestimate the number of very short calls. The Pareto distribution also starts to deviate for call holding times longer than 1000 seconds, which is about 2% of all calls in our traces and would not imply any significant effect.

By using a distribution that underestimates the level of short calls—such as the exponential distribution—the level of call signalling and the performance of those devices that handle it will be underestimated. A large quantity of short calls will produce a bursty call signalling traffic [4]. The distribution will fail to predict the burstiness of the call signalling thus the buffer size needed at the call signalling handler will be underestimated.

Our sample data contains a large number of these short calls, probably due to both the fact that our samples are retrieved from a business organization which uses an answering machine that answer calls made to employees not currently in and that callers greeted by this machine tends to hang up almost immediately. This implies that for modelling traffic not in possession of these specific characteristics, such as residential traffic, the Pareto distribution, as suggested in [8], may still be an appropriate choice. In [8] the data is collected from the Hungarian Network environment. The network name implies a national or public network where the traffic is presumed to be a mix of business and residential calls.

Since we are basing the call length on the RTP stream we are not considering call setup times, this lead to the fact that the ISDN lines are busier for a slightly longer time than what is reflected by the collected data, since the call signalling is reserving capacity prior to setting up the RTP-stream. However, the RTP-streams are the data that have the largest impact regarding network load.

For modelling speech patterns some researchers suggest their own models [12],[34]. In [10] it is suggested that the Gamma distribution fits well for talkspurt and Weibull for silence periods. Others have classified talkspurt and silence periods into periods of different length, thus using a multiple state markov based model [12],[34]. Our study, as in [8,10,12], shows that the exponential distribution do not accurately model neither talkspurt nor silence. [8] suggests the Pareto distribution for both talkspurt and silence but our study shows that for talkspurt, the Pareto distribution behaves much like the

exponential, although it starts to deviate a little later, and turns to overestimate the longer talkspurt. As earlier mentioned we suggest the lognormal distribution for modelling talkspurt and, in contrast to [8], we suggest the generalized Pareto distribution for modelling the silence periods. In our background study we have not found any research supporting our results regarding the talkspurt distribution.

An overestimation of long talkspurts, such as the result of modelling talkspurts with the exponential distribution, will imply a greater bandwidth needed. Thus using a distribution with those characteristics for modelling bandwidth requirement could result in an over dimensioned network.

Our study of call inter-arrival times suggests a more heavy-tailed distribution than the classically used exponential distribution. Further, our data is based on the busy hour for a network including 300 IP telephones and the number of calls during the busy hour was found to be 192. Further research is needed in order to prove that our results are valid in other network environments.

Our results show that inter-arrival times shorter than 40 seconds are accurately modelled by the exponential distribution which suggests that it may work for environments with short inter-arrival times. Further, our findings suggest a heavy tailed distribution of inter-arrival times. If the distribution used for modelling fail to estimate the heavy tail the result would be that the longer inter-arrival times will be overestimated, thus suggesting a lighter load on gateways and call signalling units.

A large proportion of short inter-arrival times in combination with short call holding times would be the scenario that stresses the call signalling units the most. Choosing a model that underestimates the shorter intervals of both call holding and inter-arrival could lead to a control structure that do not cope with this scenario.

8 Conclusion

In this study we have successfully developed methods to collect and derive statistical data from network traffic using freely available tools. We have matched the derived data to statistical models using Matlab. The implications of our results for network dimensioning have been briefly discussed.

We have found that for call holding, inter-arrival and speech pattern the exponential distribution do not follow the empirical data. By comparing our data to distributions commonly used in research, we have found that for call holding and talkspurt modelling the lognormal distribution fits well. For silence we find that the generalized Pareto distribution is a good fit. The results of our study could be useful for researchers and for developing tools for network dimensioning.

References

1 C. Whai-En, H. Hui-Nien, L. Yi-Bing. "Modelling VoIP Call Holdng Times for Telecommunications," Network, IEEE, vol.21, no.6, pp.22-28, November-December 2007.

2 P.E. Heegaard. "Evolution of Traffic Patterns in Telecommunication Systems," Communications and Networking in China, 2007. CHINACOM '07. Second International Conference on, pp.28-32. 22-24 Aug 2007.

3 C. Jedrzycki and V.C.M. Leung, "Probability Distribution of Channel Holding Time in Cellular Telephony Systems," Vehicular Technology Conference, 1996. "Mobile Technology for the Human Race"., IEEE 46th, vol.1, pp.247-251, 28 Apr-1 May 1996.

4 A.V. Bolotin, "Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis," IEEE JSAC, vol.12, no.3, 1994, pp.433-438.

5 F. Barchelo and J. Jordan, "Channel Holding Time Distribution In Cellular Telephony," Elect. Lett., vol.34, no.2, 1998, pp.146-147.

6 F. Barcelo and J. Jordan, "Channel Holding Time Distribution In Public Telephony Systems (PAMR and PCS)," IEEE Trans. Vehic. Tech., vol.49, no.5, 2000, pp.1615-1625.

7 L. Brown *et al*., "Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective," J.Amer. Stat. Assn., vol.100, no.469, 2005, pp.18.

8 T.D. Dang, B. Sonkoly, S. Molnár, "Fractal Analysis and Modeling of VoIP Traffic," Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International, pp.123-130, 13-16 June 2004.

9 R. Parkinson. Traffic Engineering Techniques In Telecommunications. white paper, Infotel Systems., Virginia. [Online]. Available: http://www.infotel-systems.com/Downloads/TrafEngWhitePaper.pdf

10 J. Seger. "Modelling Approach for VoIP Traffic Aggregations for Transferring Tele-traffic Trunks in a QoS enabled IP-Backbone Environment," in *Interdomain performance and simulation (IPS) Workshop, Salzburg, Austria, 2003*

11 Telephone transmission quality objective measuring apparatus: Artificial conversational speech, ITU Recommendation P.59, Mar. 1993.

12 P. Pratong, T.J. Erke, K.M. Ahmed. "Analysis and Modeling of VoIP Conversation Traffic in the Real Network," Information, Communications and Signal Processing,2005 5th International Conference on, pp.388-392, 06-09 Dec. 2005.

13 The History Of VoIP. [Online]. Available:

http://www.whichvoip.com/voip/articles/voip_history.htm

14 H. Schulzrinne *et al.,* RTP: A Transport Protocol for Real-Time Applications, RFC3550. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt

15 H. Schulzrinne. RTP: About RTP and the Audio-Video Transport Working Group. Available: http://www.cs.columbia.edu/~hgs/rtp/

16 Network Sorcery. [Online]. Available: http://www.networksorcery.com/enp/protocol/rtp.htm

17 H. Schulzrinne., SIP: Session Initiation Protocol. [Online]. Available: http://www.cs.columbia.edu/sip/

18 D. Willis and K. Drage., Session Initiation Protocol (sip). [Online]. Available: http://www.ietf.org/html.charters/sip-charter.html

19 V. Handley and V. Jacobson., SDP: Session Description Protocol, RFC2327. [Online]. Available: http://www.ietf.org/rfc/rfc2327.txt

20 B. Goode. "Voice Over Internet Protocol (VoIP)," Proceedings of the IEEE, vol.90, no.9, pp.1495-1517, Sep 2002.

21 Cisco, "Voice Over Ip," Cisco Systems., San Jose, CA, 2005. [Online]. Available: http://www.cisco.com/warp/public/788/pkt-voice-general/bwidth_consume.htm

22 Cisco, "Cisco Networking Academy Program: IP Telephony v1.0," Cisco Systems., San Jose, CA, 2005.

23 M. Arango *et al.,* Media Gateway Control Protocol (MGCP), RFC2327. [Online]. Available: http://www.apps.ietf.org/rfc/rfc2705.html

24 University West, "Växel Och Reception," [Online]. Available: http://www.hv.se/extra/pod/?action=pod_show&id=7&module_instance=16

25 D.C. Montgomery, G.C. Runger, N.F. Hubele, "Engineering Statistics," 2nd ed., New York: John Wiley & Sons, 2001, pp.49-56.

26 MATLAB. Natik, Massachusetts: The MathWorks Inc.

27 Packetscan. GL Communications Inc., [Online]. Available: http://www.gl.com/packetscan.html

28 Mediapro. Radcom Equipment Inc., [Online]. Available: http://www.radcom.com

29 Tcpdump version 3.9.5. [Online]. Available: http://www.tcpdump.org/

30 Gnome Sound Recorder. [Online]. Available: http://library.gnome.org/users/gnome-sound-recorder/index.html.en

31 SoX – Sound eXchange. [Online]. Available: http://sox.sourceforge.net/

32 P. Pratong, T.J. Erke, K.M. Ahmed. "Analysis and Modeling of VoIP Conversation Traffic in the Real Network," Information, Communications and Signal Processing,2005 5th International Conference on, pp.388-392, 06-09 Dec. 2005.

33 OpenH323. [Online]. Available: http://www.voxgratia.org/

34 A. Biernacki. "Voip Source Model Based On The Hyperexponential Distribution," in *Proc. World Academy Of Science, Engineering And Technology.* (Prague, Czech Republic, Feb 24-26, 2006, vol.11, pp.202-206)

35 Simph323. [Online]. Available: http://packages.ubuntu.com/sv/gutsy/simph323

36 V. Toncar. oh323tut version 1.1.1. [Online]. Available: http://toncar.cz/openh323/tut/files.html

37 G. Combs, Wireshark version 0.99.6. [Online]. Available: http://www.wireshark.org/download.html

38 Wireshark. KnownBugs – OutOfMemory., [Online]. Available: http://wiki.wireshark.org/KnownBugs/OutOfMemory

39 M. Dallachiesa. Rtpbreak version 1.3. [Online]. Available: http://xenion.antifork.org

40 H. Schulzrinne., RTP Profile for Audio and Video Conferences with Minimal Control, RFC1890. [Online]. Available: http://www.ietf.org/rfc/rfc1890.txt

41 G. Combs, Tshark version 0.99.6. [Online]. Available: http://www.wireshark.org/

42 Tcpslice version 1.2a3. [Online]. Available: http://www.tcpdump.org/

A Random variables

A random variable can be described as a random function where the exact outcome is not known a priori, only to a certain degree of probability [25]. The length of telephone calls (as well as talkspurt, silence and call inter-arrival times) can be described as random functions.

Take for example telephone calls that arrive at an office, it cannot be determined for how long each call is going to take. But knowing—based on earlier collected empirical data—how the length of many previous calls are distributed, calculations can answer the probability for a certain call duration on any arbitrary call that arrives.

The length of telephone calls is further an example of a continuous random variable with a semi-infinite interval. Meaning that the length of a call can take on any value within a positive interval, and that the probability to land at a certain value is virtually equal to zero. One can consider the interval where the values (the call length) can land (state space) to have an infinite number of landing points. Hence, the probability equals zero to land at a certain spot, the probability for a certain value cannot be calculated [25].

To exemplify this, say that a certain call is exactly 15.45364235 seconds long, the probability that a call is going to land at exactly this value is—as said before—very unlikely. On the other hand, saying that one or more calls are going to be between 15-16 seconds is more likely.

B Script for call length calculation

```
#!/bin/sh

#rtpbreak is used with the -W and -w flag to disable the
#extraction of the RTP-session as well as the RTP-payload.
#further is -p used to specify a libpcap filter, in this case
#only considering RTP-streams originating from the two voice gateways
# -r instructs rtpbreak to read from the dumpfile specified (out.pcap).

# Extract the RTP-sessions
./rtpbreak -W -w  -p 'ip src host <IP_ADDRESS_OF_FIRST_GATEWAY> or ip src host
<IP_ADDRESS_OF_SECOND_GATEWAY>' -r out.pcap

#The file produced by rtpbreak (rtp.0.txt) contains a lot of  information about the RTP-streams that
#it has detected, such as source and destination IP address, dropped packets etc.
#To filter out only the rows beginning with "closed:" the text
#file from rtpbreak is concatenated through the UNIX 'grep'
#command and the result
#is saved in file 'calls1.txt'.

#Filter out closed calls
cat rtp.0.txt | grep 'closed: ' > calls1.txt

#To filter out the calltime the UNIX command 'sed' is used to only save the calltime length
#information and space the minutes #and seconds with a tab character.
#The result is saved in file 'calls2.txt'.
#
# Example:
#
#Input to sed --> * [rtp0] closed: packets inbuffer=0 #flushed=6602 lost=0(0.00%),
#call_length=2m12s
#                              #  Output from sed --> 2            12

#Filter out calltime
sed -e 's/\*.\[rtp0\].closed: packets inbuffer=0.flushed=[0-9]*.lost=[0-9]*([0-9]*.[0-9][0-
9]%),.call_length=//g' -e 's/m/\t/g' -e 's/s//' < calls1.txt > calls2.txt

#In order to prepare the data for inport to matlab, all time information is to be converted from the
#form minutes/seconds to seconds.
#This is done in the "time_conv.awk" filter (described in appendix H).

#Convert calltime from minutes and seconds > seconds.
cat calls2.txt | awk -f time_conv.awk > calls3.txt

#Clean up
rm *.pcap
rm rtp.0.0.txt
rm rtp.0.txt
rm calls1.txt
rm calls2.txt
```

C Script to find the busy hour

```
#!/bin/sh
#
#This script calculates call arrival rate on a one hour basis, it is primarily used to identify the busy
#hour.
#

#tcpdump is given the '-r' flag and takes the name of the capturing file as an argument. The result
#of the filtering is saved in the file 'start_calls.pcap' using the '-w' flag.
#
#The length of the udp header is eight bytes (rfc768) and by instructing tcpdump to skip over the
#udp header one is arriving directly at the beginning
#of the header in the rtp protocol. The second byte in the rtp header contains the marker bit and
#payload type value. For a payload type of PCM u-law
#(rfc3551) set to 0 and marker bit set to 1 (rfc3550), this field shall evaluate to 0x80.
#

tcpdump -r out.pcap -w start_calls.pcap 'udp[9:1]=0x80 and (ip src
<IP_ADDRESS_OF_FIRST_GATEWAY> or ip src
<IP_ADDRESS_OF_SECOND_GATEWAY>)'

#
#thark is used to count and summarize the packets for every 3600 seconds (e.g one hour) and puts
#the result in 'call_arrival_rate.txt'
#

tshark -z io,stat,3600 -r start_calls.pcap > call_arrival_rate.txt
```

D Script to calculate the inter-arrival time

```
#!/bin/sh

#This script is used to extract the busy hour and calculating the call inter-arrival time.

#
#tcpdump is used in the same manner as for finding the busy hour (Appendix C) to reduce the data
#to only contain packets representing the start of a new call.
#

tcpdump -r out.pcap -w start_calls.pcap 'udp[9:1]=0x80 and (ip src
<IP_ADDRESS_OF_FIRST_GATEWAY> or ip src
<IP_ADDRESS_OF_SECOND_GATEWAY>)'

#
#tcpslice is used to extract the busy hour. In this case the busy hour is between 10:00-11:00. The
#data capturing starts at 07:00
#and the starttime of interest is three hours later, that is from 10800 seconds into the capturing file
#(3600 seconds * 3 hours=10800 seconds)
#plus one hour (3600 seconds). This timespan is extracted and saved in file
#'interarrival_busy_hour.pcap'
#

tcpslice -w interarrival_busy_hour.pcap +10800 +3600 start_calls.pcap

#
#finally, tshark prints the time differences between every packet and saves the result in
#'interarrival_time.txt'
#

tshark -r interarrival_busy_hour.pcap -T fields -e frame.time_delta > interarrival_time.txt
```

E Script to calculate silence

```
BEGIN {sptime=0;siltime=0}

#This awk script calculates all silence events and prints them on the form: 24;235;6952;233;5423;
#Where each record is the silence time in seconds.


#The fourth column ($4) is the rtp.marker, if it evaluates to true -1-->
$4 == 1 {
        # -1--> and this is not the first run...
        if(siltime != 0 && sptime != 0){

        # -1--> print the time in seconds ((timestamp - saved timestamp from preceeding
packet)/8000) *
        printf("%2f;", (($2-siltime)/8000))
        }

        # ... else save the timestamp (which is in the second column ($2))...
        else{
        sptime=$2
# ... And get the next line.
        next
        }
}

# if the rtp.marker is not set, save the timestamp.
$4 == 0 { siltime = $2          }


# * The audio is sampled at 8KHz and 50 packets are sent every second, hence each packet is
#carrying 20ms of audio (1/50 = 0.02s = 20ms),
#and 20ms of audio is 160 sample points (0.02*8000 = 160), (according to RFC1889)
#
#In order to calculate the silence time, one can divide the cumulated timestamp (always a multiple
#of 160) by 8000 (the samplerate) to get the result in seconds.
```

F Script to calculate talkspurt

```
BEGIN {sptime=0}

#This awk script calculates all talkspurt events and prints them on the form:
#24;235;6952;233;5423;
#Where each record is the talkspurt time in seconds.

$4 == 1 {
        sptime=$2
        next

}

#The fourth column ($4) is the rtp.marker, if it evaluates to false (end of talkspurt)--->
$4 == 0 {

        # ---> print the time in seconds ((timestamp - saved timestamp from preceeding
packet)/8000)
        printf("%2f;", (($2-sptime)/8000))
}
```

G Basic statistics

Basic statistics					
	Number of calls	Std.dev (sec)	Mean (sec)	Variance	Quantile (sec)
Call holding	25123	262,28	116,74	68789	25%=10, 50%=32, 75%=103, 95%=516
Inter-arrival	192	27	21,18	448,43	25%=2,23, 50%=9,8, 75%=24, 95%=78
Talkspurt	n/a	3,06	1,83	4,14	25%=0.66, 50%=1.2, 75%=2.49, 95%=6
Silence	n/a	2,03	1,96	9,36	25%=0.33, 50%=0.81, 75%=2.1, 95%=6.5

H AWK-filter for time conversion

```
BEGIN {}

#
# Converts time from the format: (minutes)   (seconds)
# To: (seconds)
#
# Time records are printed on the form: 24;235;6952;233;5423;
#
# (records less than 1 seconds are not considered)
{
  if(!($1==0 && $2==0)){
    call_time= ($1*60)+$2
    printf("%2d;",call_time)
  }
}
END {}
```