ENSC 835: High-Performance Networks
CMPT835: Special Topics: High-Performance Networks

Project Title: Analysis and Trace Driven Simulation of H.323 VoIP Traffic

Fall 2003
Final Project

Nenad Laskovic
nenad_laskovic@sfu.ca

Savio Lau
saviol@sfu.ca
http://www.sfu.ca/~saviol/835proj.htm

Table of Contents

Table of Contents 1
Abstract 2
Introduction 3
  What is VoIP 3
  VoIP topology 4
  VoIP Protocols 4
  Codec 5
  Modeling voice traffic 7
H.323 Protocol 8
  H.225 RAS Signaling 8
  H.225 Call Signaling 9
  H.245 Control Signaling 10
  Message Exchange Procedure 10
Trace files 12
Trace file analysis 16
NS-2 implementation 20
  UDPH225RAS 20
  UDPH225CS 20
  H323GkApp 20
  H323CltApp 21
Results 23
  Trace analysis results 23
    Call duration 23
    On-off periods 25
    Interpacket distribution during on-periods 30
  NS-2 Simulation Results 32
Conclusions 34
Future work 35
Acknowledgement 36
References 37

Abstract

Although Voice over IP (VoIP) has been introduced for some time, the adoption rate for this form of voice communication has been low until recently. One contributing factor is that VoIP can provide lower-priced international phone calls compared to traditional phone lines. Another factor is that VoIP can reduce telephone related costs in enterprise settings. In addition, protocols such as Multi-protocol Label Switching (MPLS) and Reservation protocol (RSVP) have been proposed to increase the quality of service (QoS) of VoIP. These factors make VoIP much more attractive than before.

Previous projects have analyzed VoIP performance using Constant Bit Rate (CBR) traffic, to mimic video/audio-streaming applications. Instead, this project's aim is to implement a VoIP application using the H.323 protocol in network simulator (ns-2), including many of the signaling functions. Furthermore, the model would include the capability of running trace-driven simulation based on real-traffic traces in addition to using NS-2 traffic generators.

With the help of an oversea ISP provider providing Cisco netflow and tcpdump traces, we examined a VoIP network. We analyzed the traffic trace to identify individual calls and compared the traffic patterns with standard telephony models such as exponential call duration and exponential-on/exponential off traffic pattern. In addition, using NS-2, we ran trace driven and traffic generator driven simulations. With the results, we will discuss the similarities and differences between the genuine traffic and models.

Introduction

Conveying a voice was the first service offered by communications providers, and even today, in the time of the Internet and constantly growing data traffic, the voice traffic is the largest part of whole traffic that is transferred through all types of modern communications networks.

Transferring a voice has gone trough many phases, from original frequency multiplex (FDM), over first voice digitalization using pulse coded modulation (PCM) and time multiplex (TDM), to today's packetized voice and transferring voice over data networks. First two solutions, FDM and TDM, are used in classical circuit switching telephone networks, whereas third solution is usual for packet switched data networks. Main difference between these two types of networks in respect to the voice signal is that in case of the circuit switched networks one connection (one time slot or one frequency band) is dedicated for one call no matter whether the voice signal is present or not. On the other hand in case of the packet switched networks packets will be sent only during periods when user talks, while the rest of time packet will not be generated. Therefore, we can conclude that the packet switched networks much better utilize link bandwidth and other networks' resources than the circuit switching networks. Percent of saved bandwidth if one use the packet switching for transporting voice can be up to 50%, as a typical conversation may contain 35-50 percent silence [1].A silence periods exist between sentences, but also between words, even inside words, as it illustrated on Figure 1 where it is shown a waveform of two words "digital telephony" [8]. However, in case of congestion the packet switched networks can substantially increase packet delay and loss, while due to bandwidth reservation the circuit switching networks have fixed delay and no loss.

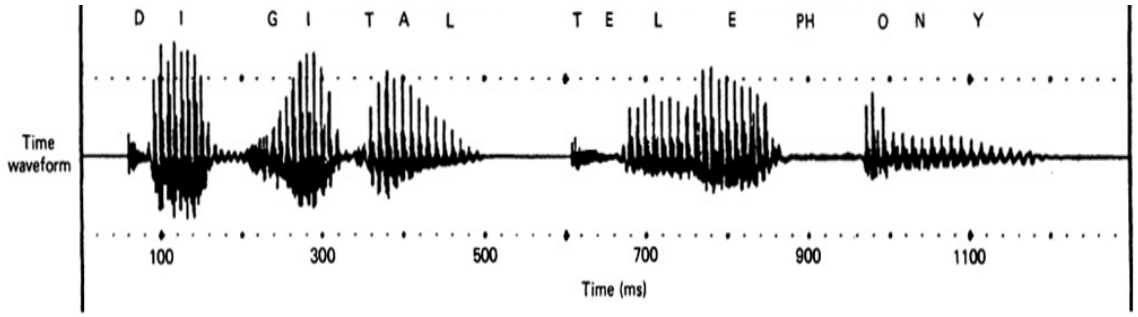

Figure 1 Waveform of words digital telephony

What is VoIP

VoIP stands for Voice over Internet Protocol. Also commonly known as Internet telephony, the term describes the method of transmitting telephone calls over the Internet instead of the traditional Public Switched Telephone Network (PSTN). VoIP has many advantages over PSTN phone services. Call multiplexing reduces the number of

telephone lines required. This reduces the cost of implementation for a new network and is the main reason that enterprises are now embracing VoIP technologies. In addition, as the voice is digitized and compressed, the amount of bandwidth required for VoIP is reduced. Furthermore, silence suppression reduces the amount of traffic sent during each call. These advantages results lower phone bills.

On the other hand, VoIP is not without its disadvantages. One of the reasons VoIP did not enjoy popularity was the poor quality of service. The traffic over the Internet, as described by researchers [15,16,17], is bursty. Constant Bit Rate (CBR) traffic, like multimedia and VoIP traffic (without silence suppression) are received with varying delay and delay jitters. The result of the delay and jitter is poor voice quality. However, researchers have proposed many solutions to improve the QoS of VoIP, including protocols Multi-Protocol Label Switching (MPLS) [18], and Reservation protocol (RSVP) [19], where VoIP traffic gets reserved bandwidth or preferential treatment over the Internet.

VoIP topology

A simple VoIP network is show in Figure 2. There are two types of users, the terminals and gateways. In the most basic form, the Terminals (T1, T2, T3, T4, T5) communicate with each other directly. The terminals can be computers or specialized equipment that supports one of the VoIP protocols. The gateway (GW) is an optional device that digitizes PSTN calls into VoIP calls. Long-distance call providers often implement gateways to provide long-distance phone service over the Internet. Another optional device is the multicast unit, the MCU. The MCU is responsible for repeating multicast packets sent by a terminal or a gateway to the many recipients of a multicast call. It is a required component if the network supports multicast. The gatekeeper (GK) is the last optional device. Gatekeepers are responsible for admission control. One gatekeeper controls an area call a zone. All VoIP users (terminals and gateways) have to request for admission if a gatekeeper is present. If there is enough bandwidth available, a user will be admitted and has permission to call the other party. If there is not sufficient bandwidth, the admission request will be denied. The connection between the routers (R) represents the connection between two VoIP networks through the Internet.

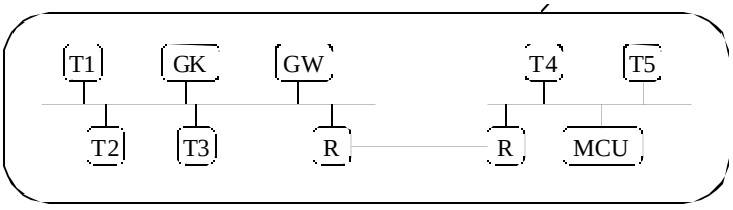

Figure 2 VoIP Network

VoIP Protocols

Excluding proprietary standards, there are two VoIP protocols, the International Telecommunication Union's (ITU) H.323 protocol and Session Initiation Protocol (SIP).

Of the two protocols, H.323 has been around longer and SIP was introduced later to compete with H.323. On top of supporting voice services, the two protocols also manages other medias such as video streaming. As H.323 has been around for a longer period of time, almost all VoIP enabled devices support H.323 but only a selected few supports SIP. Thus, for this project, we will focus on the H.323 protocol.

The two protocols differ mainly in their signaling. Thus, they share a few commonalities. First of all, they both use real-time protocol (RTP) to transmit the voice traffic over the Internet. In addition both protocols conform to the same list of coder-decoders (codecs). For example, the G.711 is the codec for PSTN communication and has a rate of 64kbps. This is a codec that all VoIP protocols have to support. Some of the common codecs are G.723 and G.729.

Codec

Conversion of the voice signal into packets is done by codec (short for coder/decoder). Input of a codec is the voice signal and its output is stream of packets. Firstly, a codec samples the voice signal, usually with constant frequency. Samples will be then digitalized, i.e. analog values of voice samples will be rounded to some predefined discrete values, and finally coded in payloads of packets. Finally, in order to transfer packets only during time when the voice signal is present, one has to isolate idle periods in the voice signal, i.e. to locate silence in the voice signal. Numerous predictive algorithms are developed for that cause. One of them is the Voice Activity Detection (VAD) algorithm. VAD algorithm is a part of the codec, and it will estimate beginning of new idle period on basis of group of the voice signal samples, not just only on basis of the voice signal level. Figure 3 illustrates stream of packets obtained from the voice signal shown on Figure 1. When user speaks codec will generate burst of packets that is known as on-period of the VoIP traffic. Periods of silence in conversation are known as off-periods of the VoIP traffic, because there are no packets during that time.

Table 1 shows characteristic of three widely used codecs, G.711, G.723.1, and G.729 [10]. Main parameters of a codec are sampling frequency, codecs bit rate, number of samples per packet, transmission rate and payload size. One can make tradeoff between quality of the decoded voice and transmission by fine tuning parameters of the codec. For example higher sampling frequency will give better voice quality, but it will also cause increase of transmission rate. Similarly, increasing number of samples per packet will decrease transmission rate, but it will increase processing time that will lead to higher packet latency.

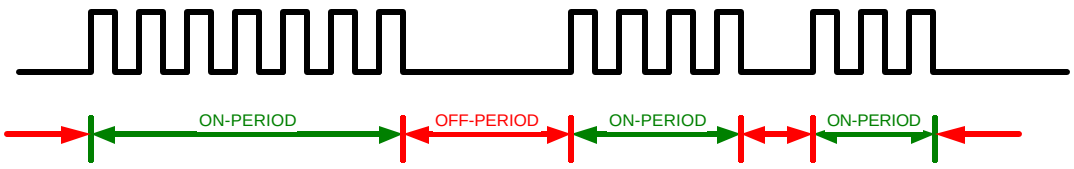

Figure 3 VoIP packets that correspond to words from Figure 1

Compression Technique	G.711	G.723.1			G.729		
Sampling frequency (Hz)	1000	33.3			100		
Codec Bit Rate (Kbps)	64	6.3			8		
Samples per packet	20	1	2	3	1	2	3
Transmission rate (Packet/sec)	50	33.3	16.7	11.2	100	50	33.3
Payload Size (Bytes)	160	24	48	72	10	20	30
Packet Size without MAC header (Bytes)	200	64	88	112	50	60	70
Overhead %	20%	62%	45%	36%	80%	67%	57%
Bandwidth at full rate (Kbps)	85.6	20.8	13.7	85.6	51.2	29.6	22.4
Bandwidth with VAD (Kbps)	55.6	13.5	8.9	55.6	33.3	19.2	14.5

Table 1 Comparison of four codec types G.711, G.723.1, G.729

Figure 4 illustrate one VoIP packet. Number of bytes in payload depends of chosen codec and codec's parameters. For instance if compression technique is G.729, and each packet contains one sample of voice signal, payload will have 10 bytes, and as a result headers will be 85% of whole packet. In some realization more than one sample is contained in one packet (Table 1), usually two or three, which will decrease amount that packet headers have in whole packet from 80% to 67 % or 57 %, in the same way placing more than one sample in one packet will decrease transmission rate. These decrease of headers overload and transmission rate are paid with longer processing time, because it is needed to wait for two or three samples to send one packet.

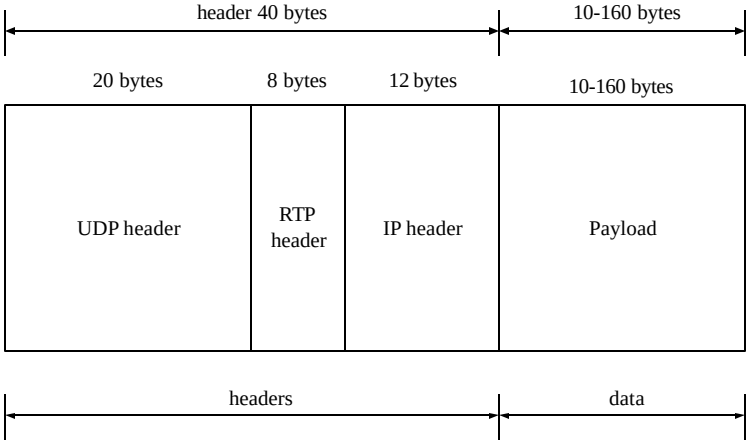

Figure 4 Format of VoIP packet

Modeling voice traffic

There are several proposed models for simulating voice traffic.

The simplest model approximates voice traffic as constant bit rate (CBR) traffic. This approximation is coarse, because does not describe off-periods of the VoIP traffic, and as we already mentioned off-periods are 35-50 % of typical calls.

The other approach in modeling the voice traffic, views separately on-periods and off-periods, and describes duration of each of those periods using exponential distribution with different mean value. This model of the VoIP traffic is known as exponential on/off model. In our project we compared exponential on/off model with real VoIP traffic.

A part of these two models, there is whole family of models that came from classical telephony networks, and that use Markov chains of different length to model speech [9]. Comparison between these models and real traffic can be one of more possible ideas for future studies.

We were concentrated on second model because is simple and wildly use in ns2 community.

H.323 Protocol

The ITU H.323 protocol is a protocol suite comprising of a number of other protocols including Real-time protocol (RTP and RTCP), H.225.0 Registration, Admission and Status (RAS) signaling, H.225 Call Signaling, H.245 Controlling Signaling and a number of other optional protocols, such as the H.450 supplementary services (call hold, call waiting and call security). The RTP portion of the protocol stack is the data portion whereas RTCP, H.225 RAS, H.225 Call Signaling and H.245 Control Signaling are the control portion of the protocol stack. The H.323's strongest point is that the included protocols are either existing protocols or modified protocols from other technologies. For example, H.225 Call Signaling is a derivative of the Q.931 messaging system used in ISDN networks. This allows H.323 companies to quickly bring VoIP devices to market using existing protocol stacks. The basic components of H.323 protocol suite and their relationship to the OSI model are shown in Figure 5. In this project, we are mainly interested in the signaling that is related to call setup and teardown.

<table>
<tr><td>Audio / Video Application</td><td colspan="4">Terminal / Application Control</td></tr>
<tr><td>Audio/Video Codecs</td><td rowspan="2">RTCP</td><td rowspan="2">H.225.0 RAS Signaling</td><td rowspan="2">H.225.0 Call Signaling</td><td rowspan="2">H.245 Control Signaling</td></tr>
<tr><td>RTP</td></tr>
<tr><td colspan="3">Unreliable Transport (UTP)</td><td colspan="2">Reliable Transport (TCP)</td></tr>
<tr><td colspan="5">Network Layer</td></tr>
<tr><td colspan="5">Data Link Layer</td></tr>
<tr><td colspan="5">Physical Layer</td></tr>
</table>

Figure 5 H.323 protocol stack

H.225 RAS Signaling

H.225 Registration, Admission and Status (RAS) signaling is the method gateways and terminals use to communicate with the gatekeeper. It is transported over UDP. If a gatekeeper is not present then H.225 RAS signaling is not used. In this project, we are interested in a three particular set messages: Registration, Admission and Disengage. There are other messages sets such as ones related to Bandwidth, and Status that we are not interested in.

Registration set of messages consists of Gatekeeper Registration Request (GRQ), Gatekeeper Registration Confirm (GCF), and Gatekeeper Registration Reject (GRJ). When a terminal or gateway (the VoIP endpoint) first power up, it is required to register with the zone's gatekeeper, if one is available. Upon receiving the request (GRQ), the gatekeeper can admit or deny an endpoint based on its user list. If an endpoint is on the authorized user list then GCF is sent. Upon receiving a GCF, a terminal or gateway is considered registered. An unregistered endpoint cannot initiate any calls.

If an endpoint is registered, the human user can places calls. At the start of each call, the terminal or gateway first has to check if the network has enough available bandwidth, analogous to the check for dial tone. A registered VoIP endpoint performs this check by sending an Admission Request (ARQ) message to the gatekeeper. If there is enough bandwidth available, the gatekeeper will reply with an Admission Confirm (ACF) message. If there is not sufficient bandwidth within the network, then the gatekeeper will reply with an Admission Reject (ARJ) message. If a terminal or gateway receives the ACF message, it can proceed to place the call another user with the use of H.225 call signaling. If a terminal or gateway is rejected it is prevented from placing a call.

At the end of each call, a terminal or gateway has to notify the gatekeeper. This releases bandwidth for another VoIP call. A VoIP host performs this by sending a Disengage Request (DRQ) message. Under normal circumstances, the gatekeeper will reply with a Disengage Confirm (DCF) message. However, if the host that sends the message does not have an ongoing call then the gatekeeper will send a Disengage Reject (DRJ) message to indicate an error. A disengaged endpoint returns to the Registered status.

H.225 Call Signaling

H.225 Call Signaling is the communication protocol between two VoIP endpoints, whether it is a terminal or a gateway. It is transmitted over TCP. If a VoIP network has a gatekeeper, H.225 Call Signaling cannot be started until a VoIP endpoint is registered and admitted to place a call. If one is not available then an endpoint can directly start H.225 Call Signaling. There are 4 messages that we are interested in for this project, which are Setup, Alert, Connect, and Release Complete. There are many other H.225 Call Signaling messages and there are many fields within them such as capability description. However, modeling these additional messages and parameters does not improve our simulated model so they are left out of the implementation.

If a VoIP endpoint wishes to place a call, it first needs to get an ACF message from a gatekeeper through an ARQ message, if a gatekeeper is available. When an ACF message has been received or if no gatekeeper is available, then the endpoint can start calling the other endpoint by sending a Setup message. The Setup message contains many fields, including the sender and receiver address, and the codec capability of each endpoint.

On recipient of a Setup message, the receiving endpoint also has to be admitted to accept the call through an ARQ message to its own gatekeeper. If a gatekeeper is not available or in the case of receiving a ACF message from the gatekeeper, the receiving endpoint sends an Alert message to the call initiator. The Alert message is analogous to the phone ringing in PSTN networks, notifying the initiator that communication between the endpoint is established but the receiving endpoint's user has not picked up the phone yet. When the receiving end user picks up the phone, the endpoint will send a Connect message to the initiator. When the initiator endpoint receives the Connect message, a VoIP call is considered established. At this point in time, the endpoints can setup logical channels used for H.245 Control Signaling.

When a call is finished, either one of the endpoints can hang up. When a user hangs up, the endpoint sends a Release Complete message to the other endpoint and ends all further data and signaling transmissions. At the same time, the endpoint is required to send a DRQ message to the gatekeeper if one was used previously. When the other endpoint receives the Release Complete message, it will perform the same by sending a DRQ message to the gatekeeper and release all its data and signaling channels.

H.245 Control Signaling

H.245 Control Signaling protocol manages the media stream between call session participants. It is transmitted over TCP. It controls properties such as the media format (supported codecs), and bit rate. In video conference calls, H.245 Control Signaling is also responsible for synchronizing the video and voice streams. Note that H.245 Control Signaling is not responsible for carry data over the network. It only controls the data. Under the H.323 specification, H.245 Control Signaling can be integrated into the H.225 Call Signaling data stream, for faster call establishment in a technique known as fast connect. In the case of our project, we assume that fast connect is part of the network and the H.245 Control Signaling messages are within the H.225 Call Signaling data streams. Thus, for the purposes of the project the H.245 Control Signaling is ignored, as it does not deal with call setup and teardown.

Message Exchange Procedure

is an example of a typical VoIP call setup and teardown. It includes elements form the above three signalling protocols. In a network with a gatekeeper, a VoIP endpoint first sends an H.225 RAS ARQ message when it wants to initiate a call. Once the endpoint receives an H.225 RAS ACF message from the gatekeeper confirming admission, it contacts the second endpoint with a H.225 Call Signaling setup message. The second endpoint performs admission request and also receives an ACF message. At this time, the second endpoint sends an H.225 Call Signaling Alert message to the first endpoint notifying that the call is established and is waiting for the user to accept the call. When the call is accepted the second endpoint sends a H.225 Call Signaling Connect message to the first endpoint. At this point the call is established.

The two endpoints then exchange their capability sets through H.225 Call Signaling. The capability set consists of codec identifications that the endpoint can support. The two endpoints picks media formats that both endpoints can support. Once this is completed the two endpoints establish H.245 Control Signaling logical channels that manages the voice or video media streams.

With the control signaling setup, the voice / video data can are sent between the two endpoints using RTP. When the call ends, one of the endpoints (in this case, endpoint 1) starts by closing its H.245 Control Signaling logical channels. The other endpoint (endpoint 2) closes its logical channels and acknowledges the first endpoints closing request. The first end point ends the connection by sending the H.225 Call Signaling

Release Complete message and sends a H.225 RAS DRQ to the gatekeeper to release the bandwidth. When the second endpoint receives the Release Complete message it will disengage from the gatekeeper also.

When both endpoints have disengaged form the gatekeeper the call is ended.

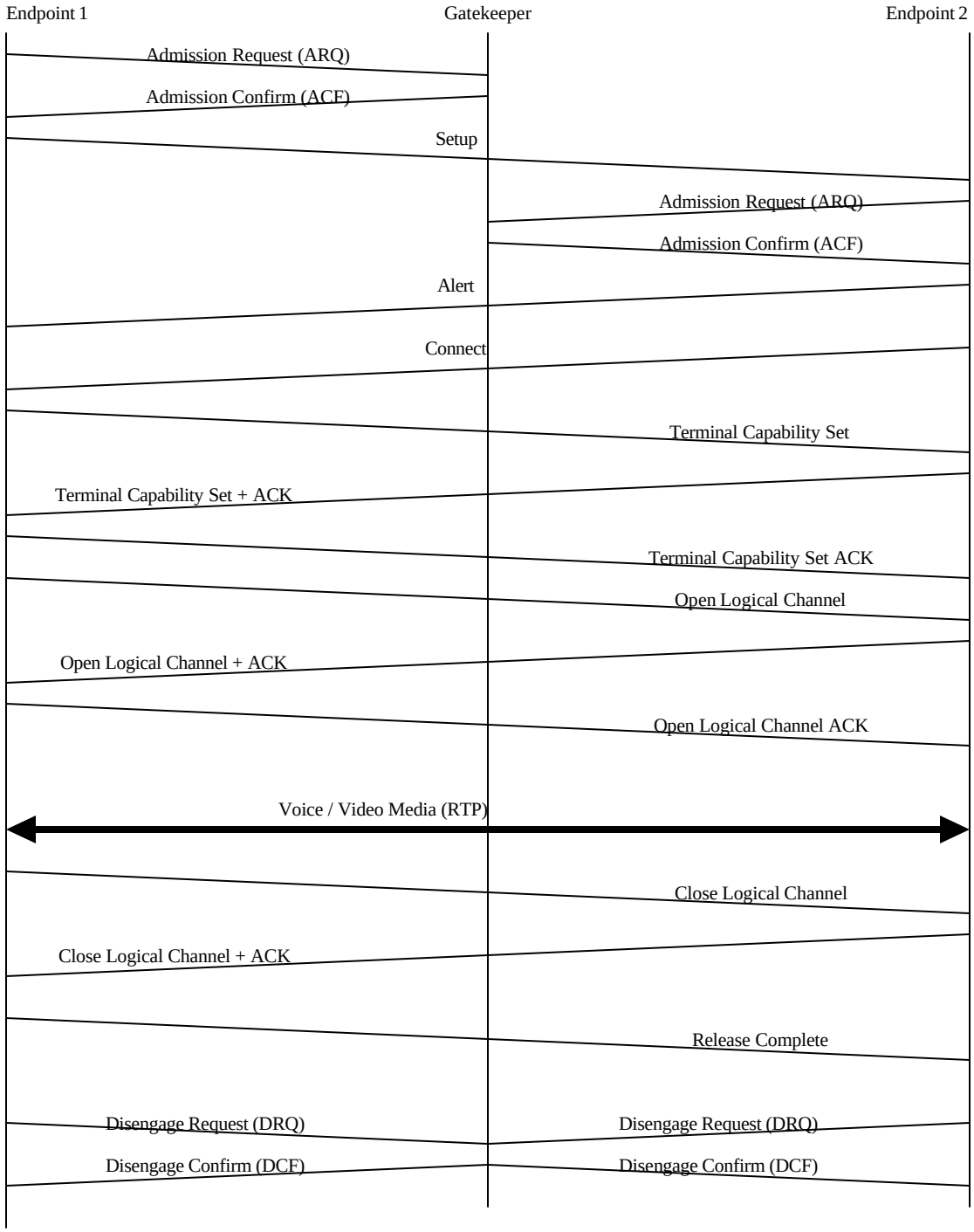

Figure 6 VoIP Message Exchange Procedure

Trace files

As a part of this project we analyzed traces of VoIP traffic obtained from one VoIP provider in Belgrade, Serbia and Montenegro. The VoIP provider is user of ISP Logikom from Belgrade. Chief technical officer of Logikom Milos Prodanovic provide us the VoIP traces and helped us a lot during this project.

Simplified topology of the VoIP provider (denoted as VoIP provider 1) is shown on Figure 7. The VoIP provider 1 uses Cisco AS5300 access server [3] as a VoIP gateway, which is connected on the one side to the other VoIP provider located in the North America (denoted as VoIP provider 2), and on the other side to 24 telephone lines.

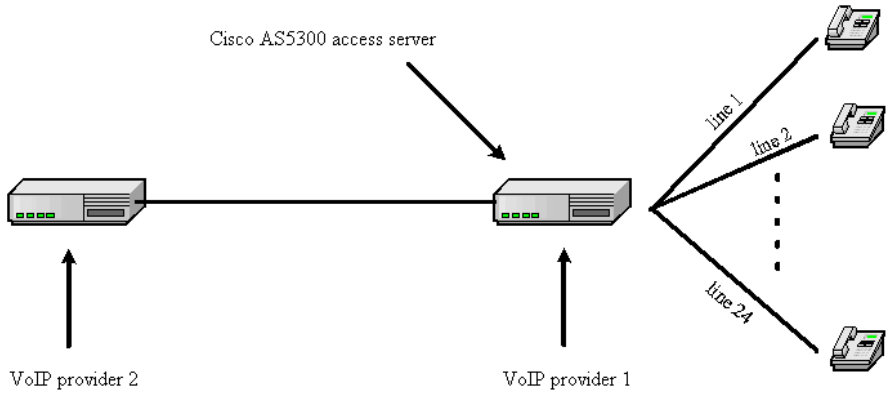

Figure 7 Simplified topology of two VoIP providers

Conversion of input telephone signals to IP packets and internal structure of Cisco AS5300 access server are illustrated on Figure 8. For each of 24 input telephone lines in this VoIP gateway exist one codec that converts input audio signal in stream of packets. All 24 output packet streams from codecs go to statistic MUX that aggregates them in one packets stream that forms output of this VoIP gateway. Similarly to this conversion of input telephone signals to IP packets VoIP gateway converts input stream of IP packets into output telephone signals. Details of this function are omitted from Figure 8 since they are analog to previously described conversion.

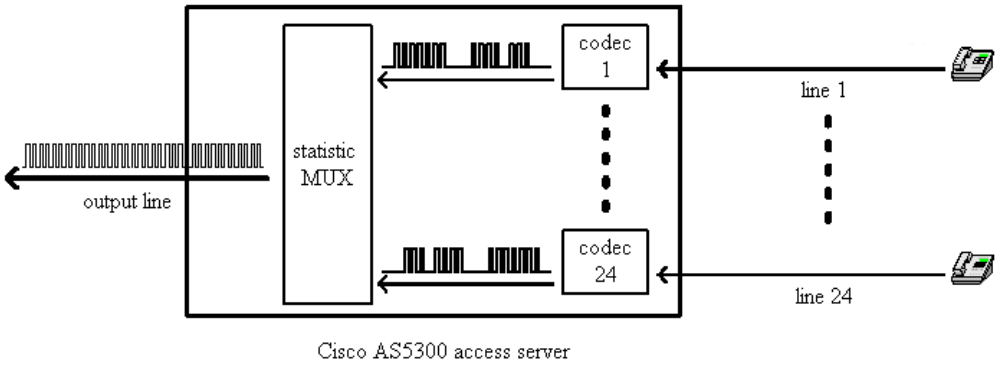

Figure 8 Conversion of input telephone signals to IP packets and internal structure of Cisco AS5300 access server

All traces that we used in our project are collected on the output and input line of VoIP gateway. In the rest of the text we will use terms VoIP provider and VoIP gateway referring to the particular VoIP provider and its VoIP gateway from which we obtained the VoIP traces.

There are two types of trace files that we used. First type is NetFlow trace [4] Cisco's standard for describing IP traffic. In case of NetFlow trace files, packets are aggregated in flows, where is flow defined as a unidirectional stream of packets between a given source and destination—both defined by a network-layer IP address and transport-layer source and destination port numbers. Specifically, a flow is identified as the combination of the following seven key fields:

1) Source IP address
2) Destination IP address
3) Source port number
4) Destination port number
5) Layer 3 protocol type
6) ToS byte
7) Input logical interface (ifIndex)

These seven key fields define a unique flow. If a flow has one different field than another flow, then it is considered a new flow. NetFlow only supports accounting for IP unicast traffic flow[4].

For analysis of NetFlow files we used software packet "Flow-tools" developed by Mark Fullmer [11]. Flow-tools is library and a collection of programs used to collect, send, process, and generate reports from NetFlow data.

Part of NetFlow file is given in Table 3. Real IP addresses of VoIP providers 1 and 2 from Figure 7 are changed in 10.0.0.1 and 10.0.0.2 respectively, and this change of real IP address we will use in the rest of the text. First field in Table 3 is a time stamp that indicates when flow started. Format of the time stamp is described in Table 2 on example of the time stamp 1113.12:27:09.166.

The NetFlow time stamp 1113.12:27:09.166					
11	13	12	27	09	166
Months	Days	hours	minutes	seconds	milliseconds

Table 2 The NetFlow time stamp format

Start time	End time	Source address	Source port	Dest. address	Dest. port	Prot.	Packets	Octets
1113.12:27:09.166	1113.12:27:19.534	10.0.0.1	18097	10.0.0.2	20743	17	4	552
1113.12:27:09.718	1113.12:27:25.426	10.0.0.1	16392	10.0.0.2	18132	17	544	32298
1113.12:26:36.026	1113.12:27:36.926	10.0.0.1	17764	10.0.0.2	1720	6	917	53709
1113.12:26:37.534	1113.12:27:37.622	10.0.0.1	16632	10.0.0.2	18426	17	2003	117843
1113.12:26:37.946	1113.12:27:34.298	10.0.0.2	16431	10.0.0.1	18137	17	12	1968
1113.12:27:27.026	1113.12:27:27.030	10.0.0.2	35166	10.0.0.1	32546	6	2	80
1113.12:27:27.866	1113.12:27:34.386	10.0.0.1	17441	10.0.0.2	17437	17	2	328
1113.12:26:51.190	1113.12:27:34.906	10.0.0.1	16633	10.0.0.2	18427	17	11	1804
1113.12:27:28.578	1113.12:27:35.134	10.0.0.2	16431	10.0.0.1	18137	17	2	328

Table 3 NetFlow

The other type of trace files used in this project is TCPdump files [5]. TCPdump is a program for capturing packets developed in Network Research Group (NRG) of the Information and Computing Sciences Division (ICSD) at Lawrence Berkeley National Laboratory (LBNL) in Berkeley, California [6].

Part of the TCPdump file for two TCP and two UDP packets is given bellow:

01:30:01.942693 10.0.0.2.46601 > 10.0.0.1.80: S [tcp sum ok] 644030637:644030637(0) win 5840 <mss 1460,sackOK,timestamp 245991535 0,nop,wscale 0> (DF) (ttl 62, id 43202, len 60)

4500 003c a8c2 4000 3e06 ce9d 0a00 0002
0a00 0001 b609 0050 2663 20ad 0000 0000
a002 16d0 d758 0000 0204 05b4 0402 080a
0ea9 886f 0000 0000 0103 0300

01:30:01.942697 10.0.0.2.46601 > 10.0.0.1.80: S [tcp sum ok] 644030637:644030637(0) win 5840 <mss 1460,sackOK,timestamp 245991535 0,nop,wscale 0> (DF) (ttl 62, id 43202, len 60)

4500 003c a8c2 4000 3e06 ce9d 0a00 0002
0a00 0001 b609 0050 2663 20ad 0000 0000
a002 16d0 d758 0000 0204 05b4 0402 080a
0ea9 886f 0000 0000 0103 0300

01:45:40.936917 10.0.0.1.19422 > 10.0.0.2.16810: udp 32 (id 13246, len 60)

4500 003c 33be 0000 fa11 021f 0a00 0001
0a00 0002 4bde 41aa 0028 0000 8012 ae61
6e8d 89cc 1f43 d214 409e b631 4c0a d926
1a46 756a f5be 5413 d189 2056

01:45:40.957836 10.0.0.1.19422 > 10.0.0.2.16810: udp 32 (id 13246, len 60)

4500 003c 33be 0000 fa11 021f 0a00 0001
0a00 0002 4bde 41aa 0028 0000 8012 ae62
6e8d 8a6c 1f43 d214 7853 5e26 c122 0661
6010 f8eb 4a69 a81a 1db1 6816

For each packet are given:

1) Time stamp (time when packet is sent) in hours, minutes, seconds and microseconds,
2) Source IP address and Source Port, separated with point,
3) Destination IP address and Destination Port, separated with point,
4) Flags
5) Checksum
6) Description of some important fields in transport protocol and IP header
7) Hexadecimal representation of the first 60 bytes in case of TCP and the first 80 bytes in case of UDP.

Trace file analysis

NetFlow traces that we used captured almost three days of a VoIP traffic. First set of data was recorded from 8th November 2003 at 3 PM to 10th November 2003 at 12 AM, and that set contains only flows which destination was VoIP provider 1 from Figure 7. The other set of data was recorded from 10th November 2003 at 3 PM to 11th November 2003 at 2 PM, and contains flows that are originate and ended in VoIP provider 1 from Figure 7. Total size of those two set of data is 20Mb.

TCPdump captured all packets that are sent or received in VoIP gateway from Figure 7 between 12:42 AM and 13:18 AM on 13th November 2003. Size of this TCPdump file is 700Mb. Along with TCPdump file we used NetFlow traces that recorded flows between 12:26 AM and 14:26 AM on 13th November 2003.

Part of our project was analysis of trace files. First goal was to revise well know fact from classical telephone system telephone that distribution of call duration has exponential distribution, and to see is that assumption true in case of VoIP traffic. Our motivation for this revision was that all calls in traces that we used are intercontinental calls that could have different duration distribution. Additionally, along with this revision we planned to study distribution of call numbers during a day, i.e. traffic's daily activity.

The second goal of trace file analysis was to match exponential on/off model of the VoIP traffic with traces. To achieve this goal we had to derive some statistic from trace files. As we mentioned the trace files that we used had recorded both received and sent packets from VoIP gateway. We decided to use only packets that VoIP gateway had sent for measuring on and off-periods duration. The reason to use only output traffic was that we assumed that statistic of received packets would be influenced with conditions of interconnecting networks between two VoIP providers. Furthermore, packet loss could change statistics of the received traffic. For all those reasons we decide to observe only packets that were generated in the VoIP gateway. On the other hand, duration of calls is not as much influenced with arrival times of individual packets, therefore in that case we could use it both input and output traffic.

The third goal was compare durations of interpacket times during on-periods with CBR traffic. Although codecs inside VoIP gateway generate VoIP packets with constant rate, interpacket time of output traffic is not constant. Reason for this is that packets from all 24 codecs compete for only one output line of the VoIP gateway, whit the result that VoIP packets from same call in aggregate traffic will not be spaced evenly. Therefore, we wanted to examine distribution of interpacket times in the output traffic.

Fist of all we had to isolate separate calls from aggregate traffic. For separating calls we used fact that both VoIP gateways allocate new port number for each new call. Consequently, each call has different pairs of source and destination ports. Although, all calls have unique pairs of ports, in trace file existed and some other UDP connections that are not VoIP calls that also have unique pairs of ports. We adopt following criterions that one UDP connection has to satisfy to be VoIP call:

1) Average size of packets has to be close to the size of data packet, because one VoIP call contains more than 90% of data packets. Accordingly we define that average size of packet has to be between 90% and 100% of the size of data packet. The size of data packet for input traffic is 70 bytes, and 60 bytes for output traffic.
2) Average number of packets per second has to be between in 50% and 100% of transmission rate, because typical telephone call contains up to 50% of off-intervals [1]. Transmission rate for input traffic is 33.3 packets per second and 50 packet per second for output traffic.
3) Duration of call has to be longer than 1 minute. The reason for this limit is that in trace file exist great number of UDP connections which duration is only a couple of seconds. That UDP connection cannot be assumed as regular telephone calls, so we had to assert a limit for duration of one call.

Example of simplified TCPdump file that correspond to one call originating from VoIP gateway is given below in Listing 1. Each line represent one packet, where first field is the time when packet is sent. Between lines are given differences between sent times of each packet in microseconds.

Next step was to find on and off-periods in each call. From documentation for Cisco AS5300 access server [3] and TCPdump files we conclude that voice is coded using G.729 standard and that each VoIP packet contains two samples of the voice signal for the output traffic and tree samples for the input traffic. It follows that length of the voice packets without MAC header in the trace files that we used, the output traffic, is 60 bytes. Consequently transmission rate during on-periods is 50 packets in second, and time between two sequential packets is 20 milliseconds. The time difference between two sequential VoIP packets is not constant, as we can see from Listing 1, but its average value is approximately 20 milliseconds. As we mentioned, the cause of this jitter is statistical multiplexer in VoIP gateway.

VAD algorithm used in speech coders G.723.1 and G.729 of ITU-T recommendation is described in Annex A of G.723.1 ITU-T recommendation [12][13]. G.723.1 recommendation Annex A defines Silence Insertion Descriptor (SID) packets which require fewer bits than the active speech frames and are transmitted during off- periods. ITU-T REPORT R 30 defines length of SID packets to 15 or 16 bits depending on the options [14]. Decoder will use a Comfort Noise Generator (CNG) algorithm during off-periods to generate an artificial noise, because absence of any signal is unpleasant for user, and even can reduce the intelligibility of the speech. The main feature of this CNG algorithm is that the transmission of SID packets is not periodic: for each SID packet, the algorithm makes the decision of sending a SID packet or not, based on a comparison between the current SID packet and the preceding SID packet. Decision how many packets will be during off-period depends on the level of noise. In one off-period can be one SID packet in case of constant level of noise or several SID packets in case that level of noise is changing.

In the TCPdump files exist two types of packets, data packets with length of 60 bytes, and packets which length is 41 bytes. We assumed that these packets are SID packets, even they have only one byte of payload. In addition to fact that length of those packets is significantly less than length of the rest of packets, a time difference between those packets and next packets was usually much grater than 20 milliseconds, what is average difference between two VoIP packets.

The first assumption was to simply use SID packets as indication of off-periods. After analysis we discovered that many off-periods have duration equal or less then 20 milliseconds, which implies that in fact there was no break in packet stream. Example of such off-period is given in line 4 of Listing 1. We can see that difference between packets 4 and 5 is less than 20 milliseconds, and from that we can conclude that packet 4 does not indicate beginning of an off-period, than it is an oscillation in functioning of VAD algorithm. Hence packet 4 is just a part of VAD algorithm transient behavior between an end of one on-period and beginning of next off-period. Complete transition from voice to silence is illustrated in Listing 1 from line 4 to line 17. As we can see detection of silence in speech is not simple task, and often is hard to clearly distinguish an end of one period from a beginning of the other. Oscillations of VAD algorithm result that duration of on and off-periods can have values from 20 milliseconds to 10 seconds.

In Listing 1 can also be observed burst of SID packets, in lines 13 to 16. As we mentioned these burst means that level of noise is variable during of period. These 5 SID packets do not convey VoIP data but are still part of the VoIP traffic. Decision that one has to make is whether these packets should be considered as a part of on-period or as a part of off-period, in which case they should be ignored. In order to simplify traffic modeling we decide to ignore these packets, because time difference between them is significantly greater than 20 milliseconds, what is the average interpacket time, and for that reason they cannot be considered as part of on-period. We used following rules to determine limits of on and off-periods:

1) A SID packet is an end of an on-period only if the next packet comes after more than 20 milliseconds. (By this rule SID packets in the line 4 and the line 11 cannot be taken as the and of the on-period. On the other hand SID packets in the line 9 and the line 13 are ends of on-periods.)
2) An end of one on-period is also beginning of the next off-period.
3) End of an off-period is the first new data packet. Also, an end of off-period is beginning of the next on-period. This rule implicitly says that all SID packets during off-period will be ignored. (For example SID packets in lines 14, 15, 16.)

1) 12:42:19.305607 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31485, len 60) +21188
2) 12:42:19.326795 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31485, len 60) +20458
3) 12:42:19.347253 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31485, len 60) +17033
4) 12:42:19.364286 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +18966
5) 12:42:19.383252 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +22389
6) 12:42:19.405641 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +19276
7) 12:42:19.424917 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +22685
8) 12:42:19.447602 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +9945
9) 12:42:19.457547 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +41987
10) 12:42:19.499534 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +6783
11) 12:42:19.506272 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +19488
12) 12:42:19.525756 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 60) +10119
13) 12:42:19.535955 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +59461
14) 12:42:19.595416 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +29881
15) 12:42:19.625297 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +29401
16) 12:42:19.654698 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31502, len 41) +65044
17) 12:42:20.309742 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31510, len 60) +17570
18) 12:42:20.327312 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31510, len 60) +20662
19) 12:42:20.347974 10.0.0.1.17860 > 10.0.0.2.20068: udp (id 31510, len 60)

Listing 1 Example of simplified TCPdump file that correspond to one call (between lines is given difference between sent times of each packet in milliseconds)

NS-2 implementation

The H.323 model is implemented in NS-2 version 2.26. Besides using standard NS-2 models, the implementation adds two new agents and two new applications. The two new agents are UDPH225RAS and UDPH225CS that implemented functions of H.225RAS and H.225 Call Signaling, respectively. The two new applications are H323CltApp and H323GkApp that are the implementation of the H.323 client and the H.323 gatekeeper.

UDPH225RAS

The H.225 RAS signaling is implemented as an extended UDP agent. The H.225 RAS messages are modeled as a new packet header type H225RAS. Besides testing code, the cores of this agent are the send and receive functions. Function "sendmsg" receives H225RAS headers from application and forwards them to another UDPH225RAS agent. Function "recv" receive H225RAS headers from another UDPH225RAS agent and forwards it up to the application.

UDPH225CS

H.225 Call Signaling is implemented in a similar manner to the H.225 RAS signaling by extending the UDP agent and adding a new packet type. The new UDP agent is called UDPH225CS and the new packet type is called H225CS. For modeling purposes, we have to model the H.225 Call Signaling as a UDP agent instead of a TCP agent contrary to the specification. The reason for this is twofold. First of all, it is much easier to develop the H.225 Call Signaling using UDP since its implementation would be very similar to the H.225 RAS Signaling. In addition, as the TCP agent only returns byte size and not data, we have to implement data sending through a wrapper like the TCPApp described in the NS-Manual. However, all attempts to implement the signaling in this manner had resulted in mysterious errors even though the implementation is very similar to the TCPApp. As a result, we used UDP instead of TCP in the final implementation.

Like the H.225 RAS implementation, the core of the UDPH225CS agent lie in the send and the receive functions. Function "sendmsg" receives H225CS headers from the application and forwards them to another UDPH225CS agent. Function "recv" receive H225CS headers from another UDPH225CS agent and forwards it up to the application.

H323GkApp

The H.323 gatekeeper is implemented using the H323GkApp Application class. The gatekeeper connects to the UDPH225RAS agent. The relationship between the

H323GkApp, and the UDPH225RAS agent is shown in Figure 9. It is responsible for answering VoIP endpoint's RAS messages. The three that the gatekeeper answers are Gatekeeper Registration Request (GRQ), Admission Request (ARQ) and Disengage Request (DRQ). In this implementation, we assume that all of the clients are well behaved. Thus, the gatekeeper always answers GRQ with GCF, which is confirming reply. The gatekeeper is preset to have certain channel bandwidth available for VoIP endpoints (by default or through the TCL script). When an ARQ comes, the gatekeeper looks at the available bandwidth and decides if there is enough bandwidth remaining. If enough bandwidth is available, the gatekeeper subtracts the requested bandwidth (inside the H225RAS message) and admits the caller with an ACF reply. If insufficient bandwidth exists, the gatekeeper replies with an ARJ. The reverse happens for DRQ and DCF. DRQ are always answered with DCF (assuming well behaved clients) and the requested bandwidth is released. The gatekeeper has "start" and "stop" functions that turn it on and off. The gatekeeper will only respond to request messages if it's on.

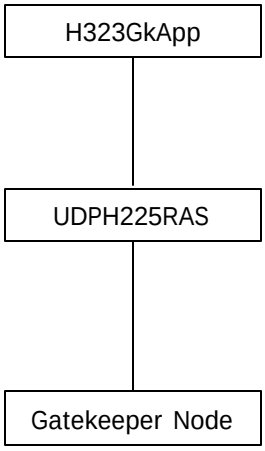

Figure 9 Gatekeeper application class relationships

H323CltApp

The H.323 Client application is the most complicated class in this project. Each client connects to two agents and one traffic generator. The relationship between all parts of the Client application is found in Figure 10. The H323CltApp connects to the UDPH225RAS and UDPH225CS for RAS Signaling and Call Signaling, respectively. The pointer to TrafGen allows H323CltApp to control NS-2 standard traffic generators. The TrafGen can be exponential-on/off (Exponential), constant bit rate (CBR), trace-driven (Trace) and many others. Instead of implementing our own traffic generators, the TrafGen pointer allows us to start and stop the traffic generators through TCL::instance.evalf() commands. The Traffic generator is connected to a standard NS-2 UDP agent. On the receiving end, the Null agent collects the data send by the traffic generator at the opposite endpoint and destroys it. Thus, the UDPH225RAS and UDPH225CS agents are the control portions whereas the traffic generator TrafGen, the

UDP agent and the Null agent represents the voice data portion of the VoIP client application.

The H323CltApp is started through the TCL script "start" command. At this point the client will try to contact the gatekeeper and register. With the "call" command the client will try to get admitted into the network and call the endpoint it's connected to. The "hangup" TCL command will end an ongoing call and "stop" will place the client in standby mode unresponsive to incoming requests.

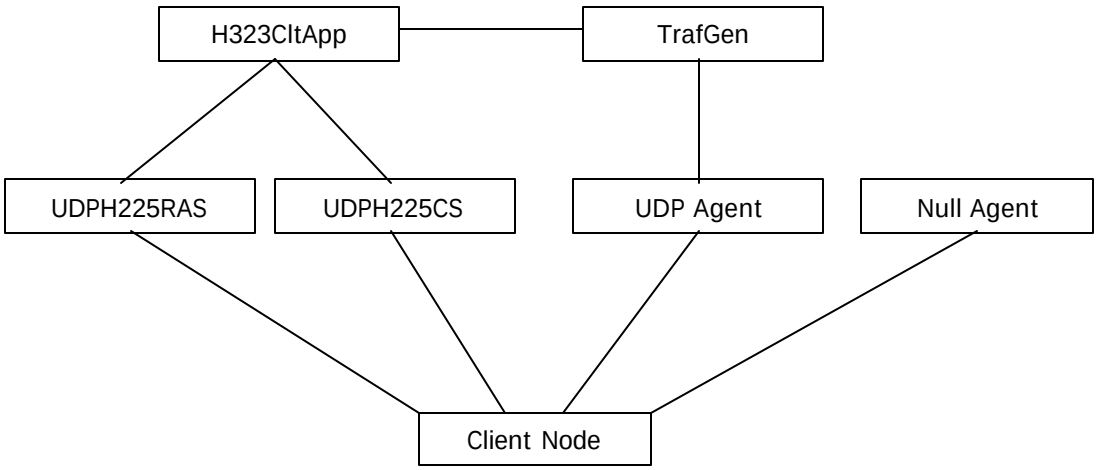

Figure 10 Client application class relationships

Results

Trace analysis results

From the TCPdump traces we isolate 37 individual calls, with duration from 5 do 35 minutes. Additionally, from NetFlow we separated 3161 individual calls in period of three days.

Call duration

First goal was to revise well know fact from classical telephone system telephone that distribution of call duration has exponential distribution, and to see is that assumption true in case of VoIP traffic. Unfortunately, due to small number of day for we have NetFlow files we could not have done analysis of daily activity. Even though, numbers of calls in the three day that we observed, 9th, 10th, and 11th of November, show some indication of daily patterns. Figure 11 displays number of calls on 9th November 2003.

Figure 12 and Figure 13 illustrate distribution of call duration in trace files and files generated by random generator with exponential distribution, respectively. Distributions in both cases have the same mean value, and the same number of samples. The gap on Figure 12 is result of our decision not to include calls shorter than one minute. Although, distribution on Figure 12 and Figure 13 show some similarity, QQ plot of these two distributions Figure 14 reveals differences for larger values of the call duration. Still, more precise analysis of call duration requires traces for larger number of days.

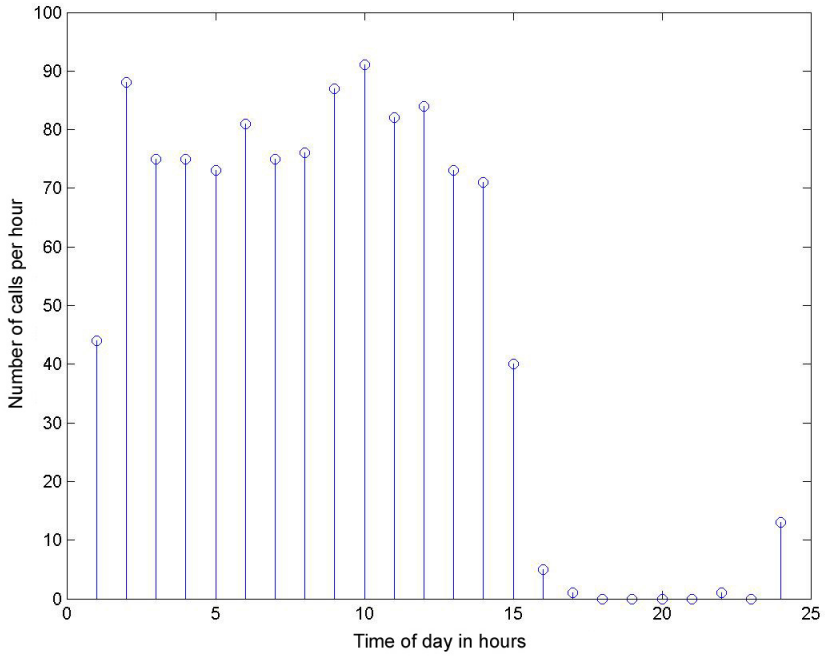

Figure 11 Distribution of calls during one day (9th November 2003)

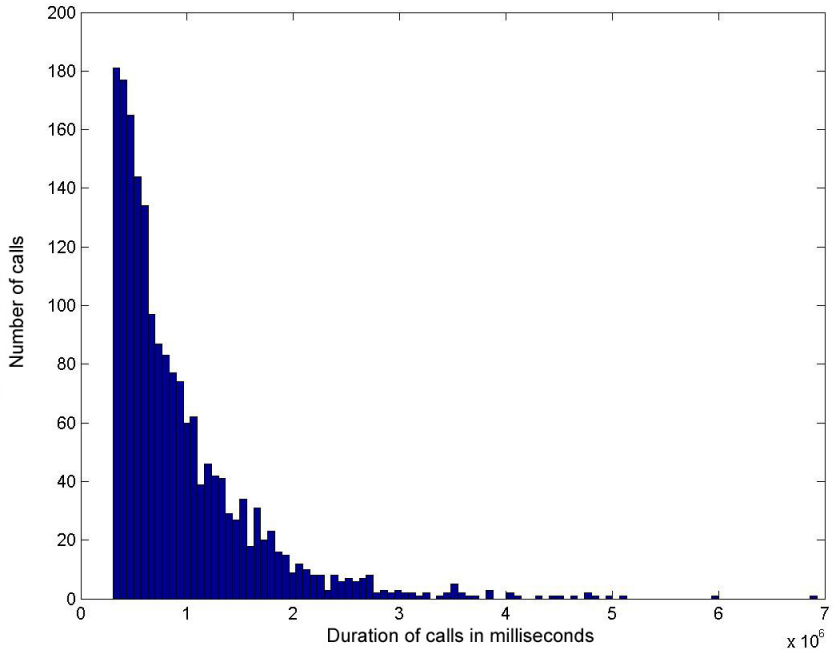

Figure 12 Distribution of call duration in trace files (Total number of calls is 3161)

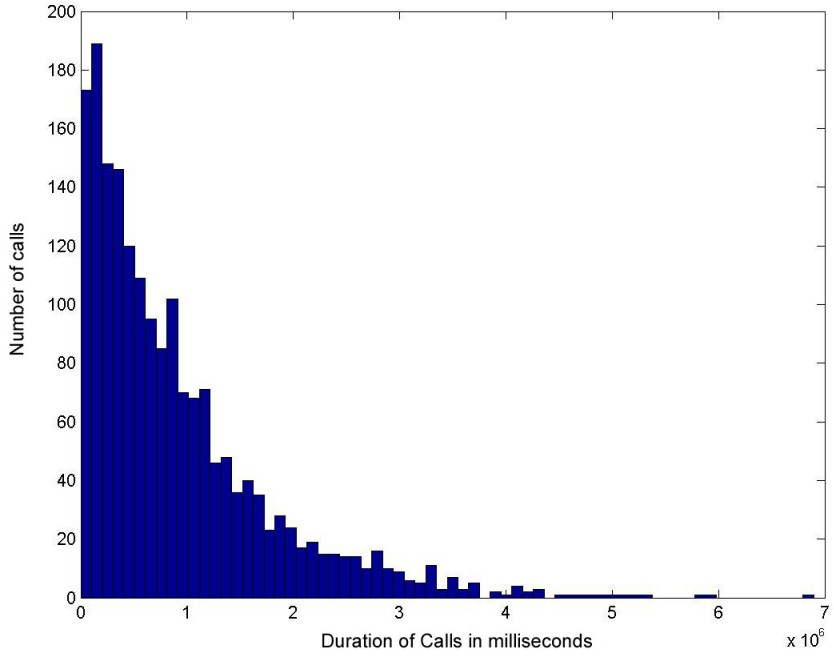

Figure 13 Distribution of call duration generated by random generator with exponential distribution

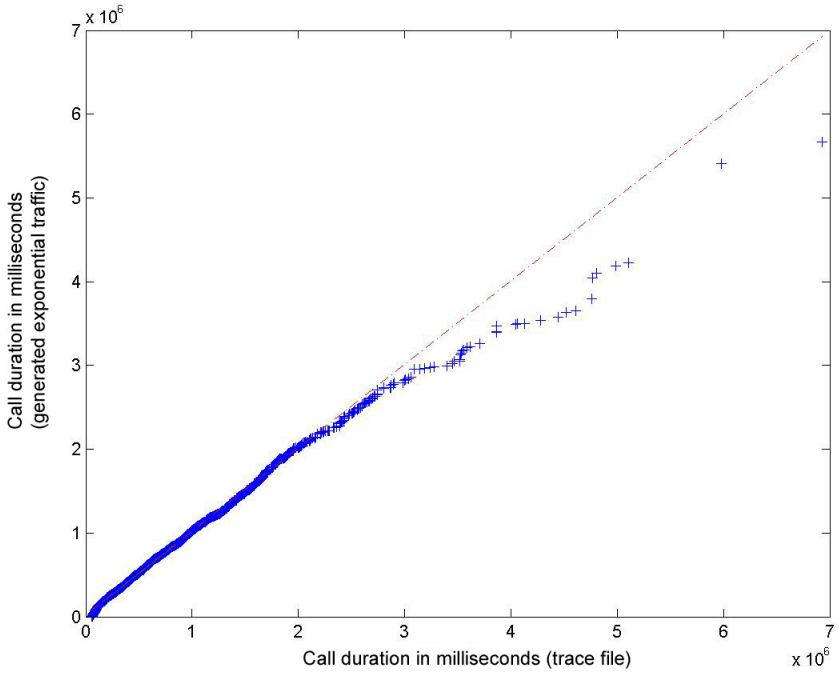

Figure 14 QQ plot of call durations in trace file and in file generated by the random generator with exponential distribution

On-off periods

Second goal of trace file analysis was comparison of on and off-period duration in trace files with distribution produced by random generator with exponential distribution. Figure 15 and Figure 16 illustrate off-periods comparison, while Figure 17 and Figure 18 present on-periods comparison. Random generator from Figure 16 has the same mean value of off-period duration as trace on Figure 15. Similarly, random generator from Figure 18 has the same mean value of on-period duration as trace on Figure 17.

From these two pair of figures it is obvious that exponential distribution is not satisfactory model neither for off-periods nor for on-periods duration. Moreover, Figure 19 QQ plot for off-period distributions and Figure 20 for on-period distribution, are even stronger evidence that distribution of on and off periods in trace files are not the same with exponential distribution.

Distributions in case of real traffic have much more short periods than in case of exponentially generated, with durations less than a couple hundreds milliseconds. Also in real traffic exist periods that last for several seconds that do not exist in exponentially generated. Even that long periods are rare in real traffic they cannot be neglected because there duration is comparable with total duration of the shortest periods. For example, 550

shortest off-periods have average length of 30 milliseconds, and there sum is 15 seconds, while three longest off-periods have total duration of almost 13 seconds.

The fact that real traffic has on and off periods which duration can be in rang from a tenth milliseconds up to a tenth of seconds can be explained on the one hand by characteristic of human speech and on the other hand with functioning of VAD algorithm.

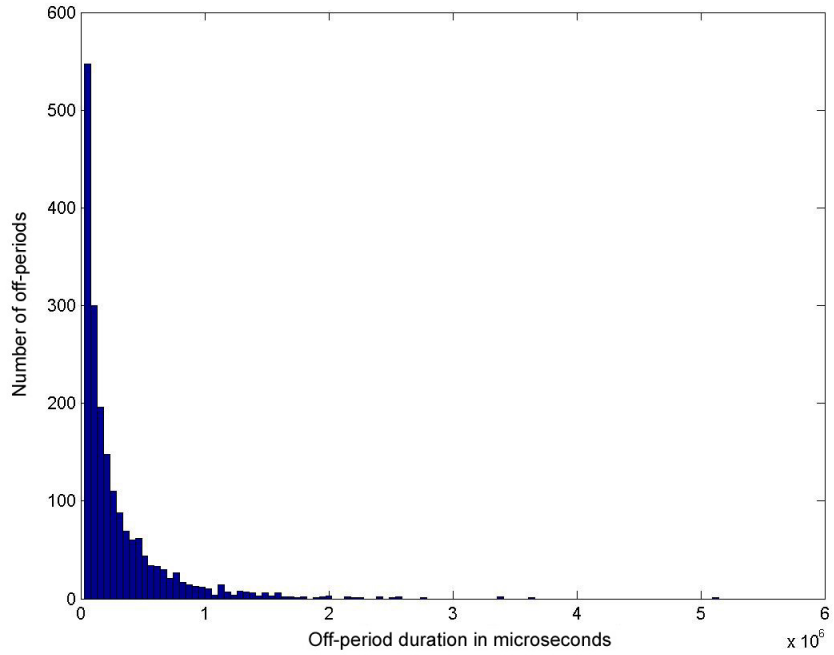

Figure 15 Distribution of off-periods in trace file of one call

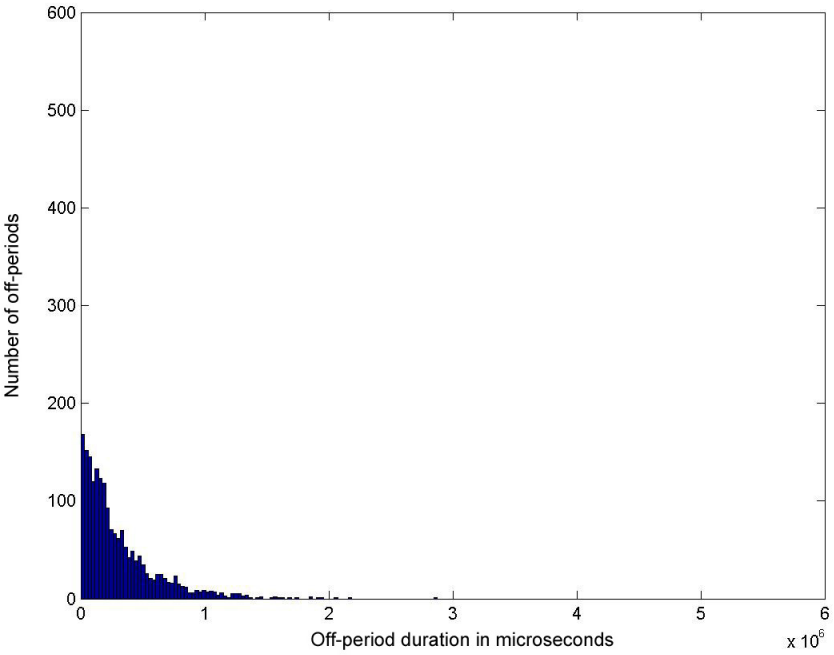

Figure 16 Distribution of off-periods generated by random generator with exponential distribution

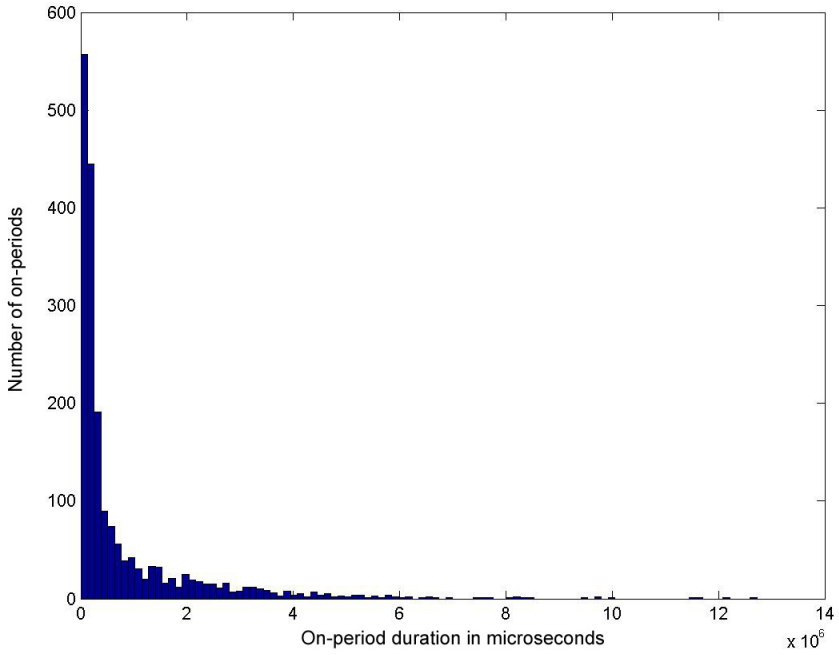

Figure 17 Distribution of on-periods in trace file of one call

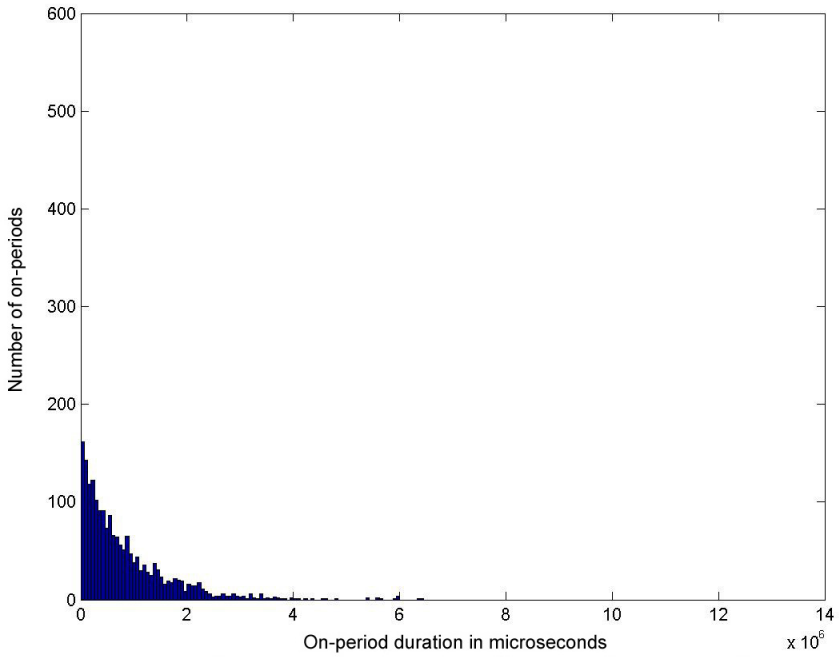

Figure 18 Distribution of on-periods generated by random generator with exponential distribution

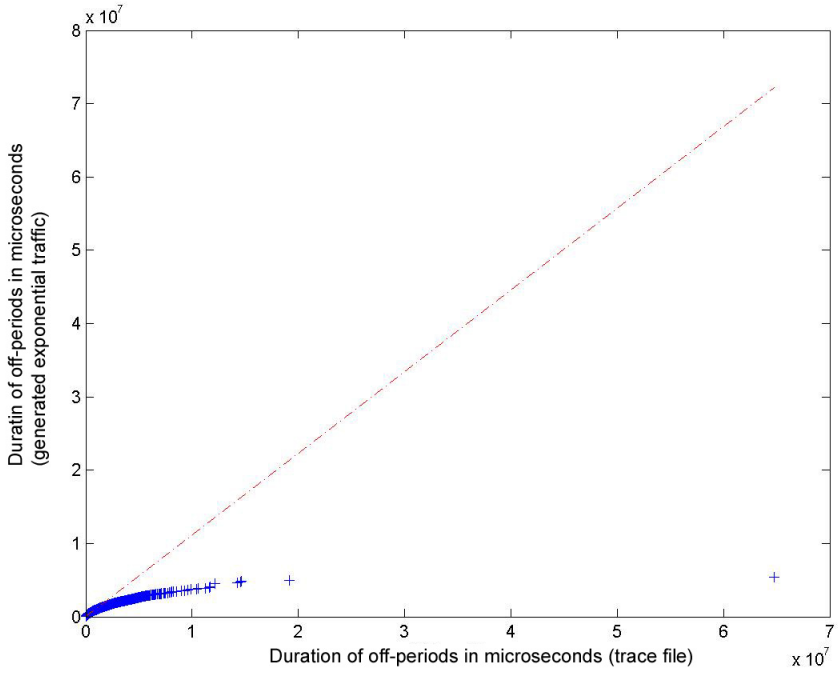

Figure 19 QQ plot of off-periods in for all calls in trace file and in file generated by the random generator with exponential distribution

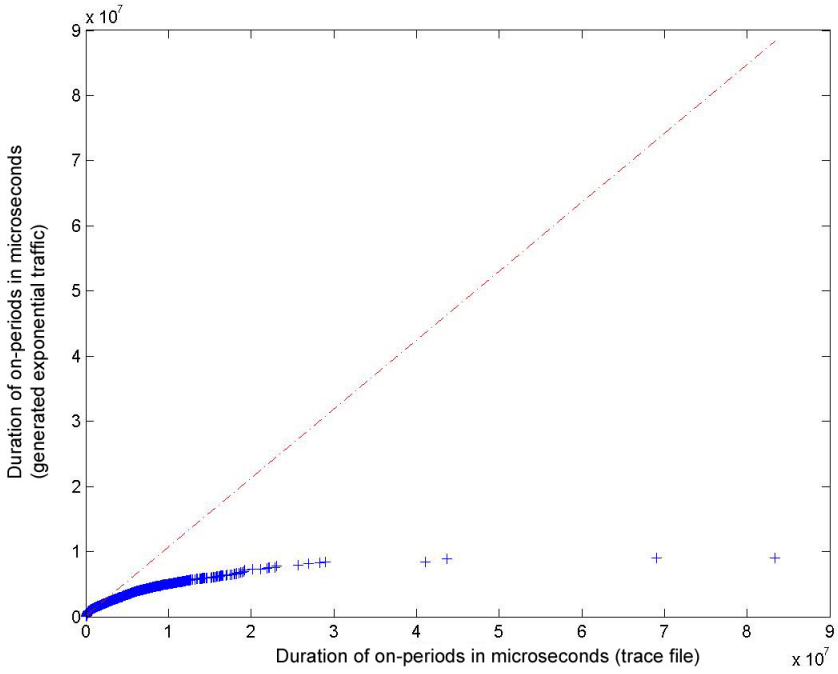

Figure 20 QQ plot of on-periods in for all calls in trace file and in file generated by the random generator with exponential distribution

Interpacket distribution during on-periods

The third goal was compare duration of interpacket times during on-periods with CBR traffic. Our assumption is that if we can approximate interpacket times in trace files with normal distribution with small standard deviation and mean value close to 20 milliseconds, we can model traffic during on-periods with CBR traffic. Figure 21 and Figure 22 illustrate distribution of interpacket times in trace file and files generated by random generator with normal distribution respectively. Although these two figures look similar, interpacket times in case of trace files have some large values that do not exist in generated traffic. These values are reason for difference between that can be observed on QQ plot for two aforementioned distributions from Figure 23. Even that average interpacket time is 19900 microseconds, standard deviation is 3637 microseconds, and 98% percent of all interpacket times lies in the interval between 10 and 30 millisecond, existing of those large values of interpacket time preclude use of CBR traffic as model of packet burst during on-periods.

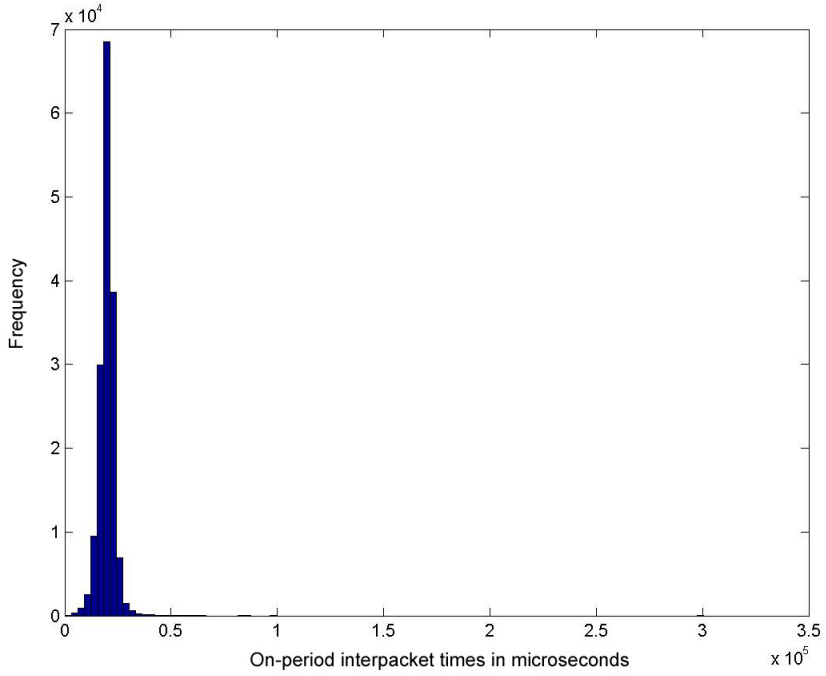

Figure 21 Distribution of interpacket times during on-periods in trace file

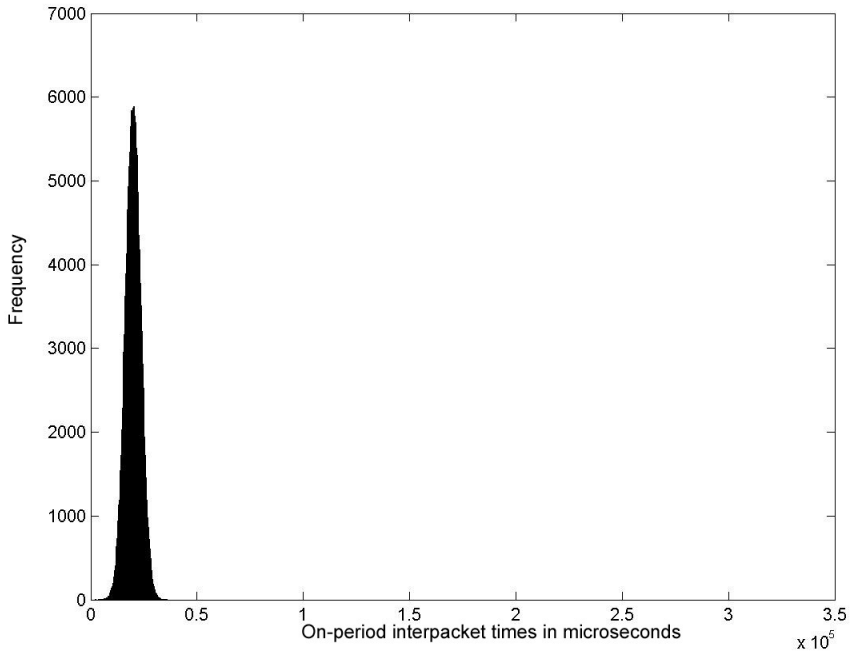

Figure 22 Distribution of interpacket times generated by random generator with normal distribution

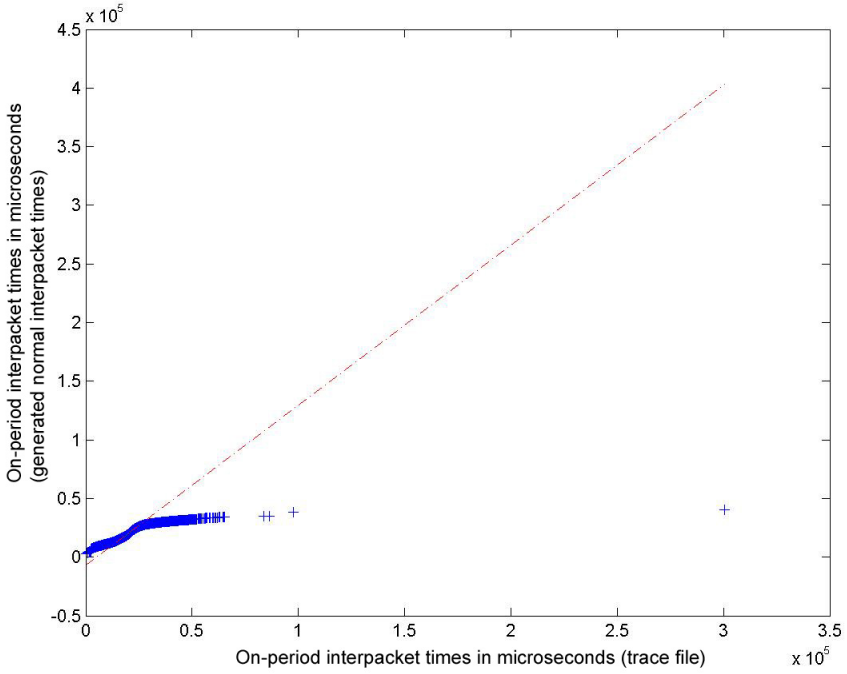

Figure 23 QQ plot of interpacket times in for all calls in trace file and in file generated by the random generator with normal distribution

NS-2 Simulation Results

Figure 25 through Figure 10 are the results of trace driven simulations. In each instance, the simulation is setup similar to Figure 24, but with 24 lines on each side instead of 4 lines. Nodes on the left side are the caller VoIP callers, the nodes on the right side are the VoIP callees, the middle node (8) is the gatekeeper and the 3 nodes 9, 10 and 11 are the routers. Each of the caller clients is connected to the Trace traffic generator. In figure 7, a single trace that contains 20% silence is used, in figure 8 50%, in figure 9 30%. Figure 10 is a simulation where 4 traces (with 20%, 25%, 30%, and 50% silence) are used. In that scenario, 6 nodes are connected with each of the 4 traces. Theoretical CBR traffic using the same codec (G.729) would have resulted in 72kbps. Thus, the 4 graphs show bandwidth utilization improvements with silence suppression. One thing that we wish to but unable to do is to compare Exponential-On/Off traffic with the trace-driven traffic. We have strange classifier errors when we switch from CBR traffic to Exponential traffic even though only two settings are changed. Without simulation results we are unable to do a proper comparison.

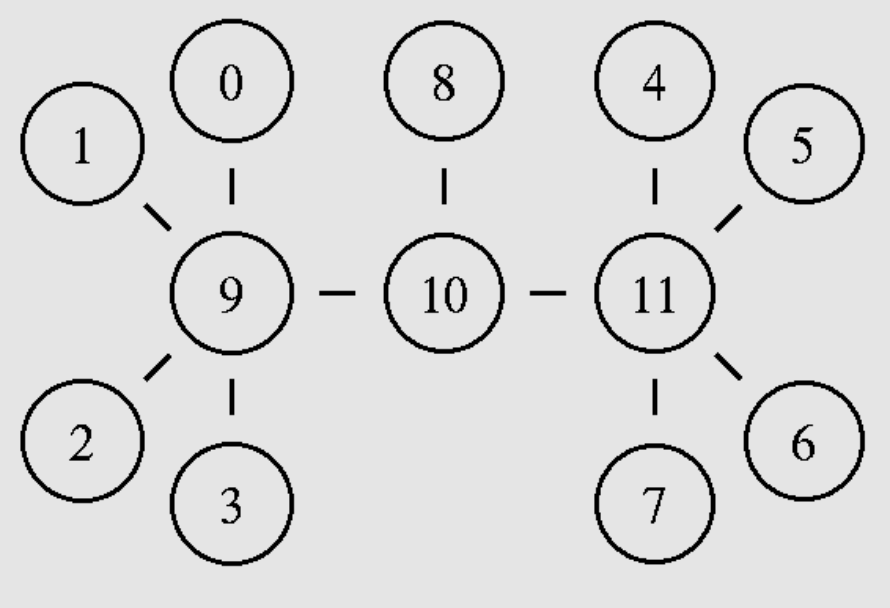

Figure 24 Example of simulation scenario

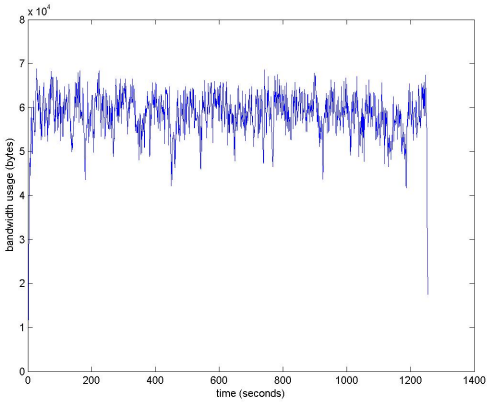

Figure 25 20% silence

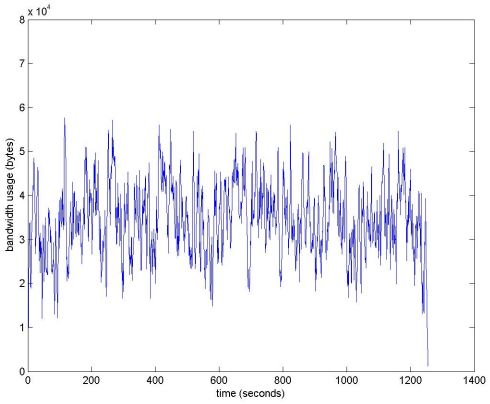

Figure 26 50% silence

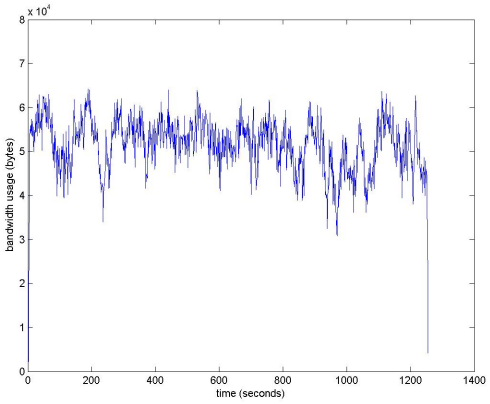

Figure 27 30% silence

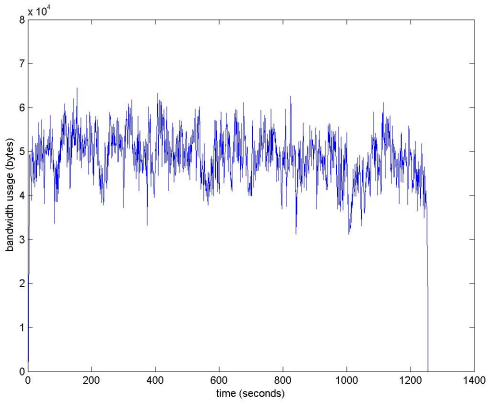

Figure 28 Mix of 20%, 30% and 50% silence

Conclusions

Firstly, we cannot conclude with certainty is exponential distribution good or bad model for duration of VoIP call. Although study on the limited number of call shows some level of similarity.

Secondly, after analysis of trace files, we can say that exponential distribution does not model well on and off-periods duration of VoIP calls.

Thirdly, we also can conclude that exponential distribution is not a satisfactory model for interpacket time during on-periods.

Fourthly, after many frustrating days and nights, we still have no solution for solving the strange errors that we have encountered during simulations. The errors have very little to do with the miniscule changes we have made (such as changing from a CBR traffic generator to a Exponential traffic generator). It would have been better if we could have got the Exponential traffic generator working to have more simulation results for comparison.

Future work

The first step in future extension of this project should be considering other distribution for modeling on and off-periods of VoIP traffic.

In addition, as only parts of H.323 protocols are implemented, others may wish to implement the remaining parts for simulation purposes. Furthermore, one may want to compare the performance of the two signaling portion of SIP and H.323.

Acknowledgement

The authors would like to thank Milos Prodanovic, Chief technical officer of Logikom, Belgrade, Serbia and Montenegro, for comments, discussions, and especially for providing the VoIP traces that helped improve the content of this project.

References

[1] VoIP over Frame Relay with Quality of Service (Fragmentation, Traffic Shaping, LLQ / IP RTP Priority)
http://www.cisco.com/en/US/tech/tk652/tk698/technologies_configuration_example09186a0080094af9.shtml(12/7/03)
[2] www.logikom.net/traces(11/12/03)
[3] Voice over IP for the Cisco AS5300
http://www.cisco.com/en/US/products/sw/losswrel/ps1830/products_feature_guide_chapter09186a008008808d.htm(12/7/03)
[4] NetFlow services solutions Guide
http://www.cisco.com/univerod/oc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm#xtocid7l
[5] TCPdump http://www.tcpdump.org/ (12/7/03)
[6] Network Research Group (NRG) at Lawrence Berkeley National Laboratory (LBNL) in Berkeley http://ee.lbl.gov/(12/7/03)
[7] H.323 Protocol International Telecommunication Union
[8] J. Bellamy, Digital Telephony, 3rd Ed. NY:NY, John Wiley & Sons, 2000
[9] D. Minoli, Delivering Voice over IP Networks, D. Collins, Carieer Grade Voice over IP, New York, New York: McGraw Hill, 2001.
[10] VoIP Bandwidth Consumption
www.tekdata.co.uk/pdfs/bos/VoIP%20Bandwidth%20Consumption.pdf (12/7/03)
[11] Flow-tools http://www.splintered.net/sw/flow-tools/ (12/7/03)
[12] G.723.1 ITU-T recommendation Annex A
[13] G.729 ITU-T recommendation
[14] ITU-T REPORT R 30
[15] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," IEEE/ACM Transactions on Networking, vol. 9, no. 4, pp. 392 - 403, August 2001.
[16] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling," IEEE/ACM Transactions on Networking, pp.226-244, June 1995.
[17] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," /IEEE/ACM Trans. Networking,/ vol. 2, pp. 1-15, Feb. 1994.
[18] D. Minoli, Voice over MPLS: planning and designing networks, New York, New York: McGraw Hill, 2002.
[19] Ed. Braden, et. al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," IETF RFC 2205, September 1997; http://www.ietf.org/rfc/rfc2205.txt.