Synchronized Two-way Voice Simulation Tool for Internet Phone Performance Analysis and Evaluation

Adrian E. Conway
Tel: 617-466-2440 Fax: 617-890-9320 E-mail: aec0@gte.com

Sue B. Moon[1]
Tel: 413-545-3179 Fax: 413-545-1249 E-mail: sbmoon@cs.umass.edu

Paul Skelly
Tel: 617-466-2807 Fax: 617-890-9320 E-mail: pas0@gte.com

GTE Laboratories Incorporated
40 Sylvan Road
Waltham, MA 02254
USA

Abstract: A simulation tool is developed for the performance analysis and evaluation of Internet phone applications. The tool can be used off-line to simulate real two-way phone conversations under different Internet loss and delay conditions. The simulation model includes access links that connect an Internet service provider to the Internet, as well as background Internet phone and data traffic. It also includes details of Internet phone implementations such as encoding, packetization, silence detection, and the IP, UDP, and RTP protocols. An original feature of the simulator is that it takes into account explicitly the synchronization of talkspurts in a two-way conversation. Example results of two-way conversations under various delay and loss conditions are provided in audio files. Pointers to download the files are at http://www-net.cs.umass.edu/~sbmoon/synch.html.

[1] Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA 01003, USA. Work carried out at GTE Laboratories in 1996 as a Summer Member of Technical Staff.

1. Introduction

Internet phone has recently gained much attention as a computer-to-computer application for conducting voice conversations over the Internet. Software products are now available commercially or as freeware. Examples of commercial packages include Cooltalk 1.0 by Netscape Comm. Corp and Internet Phone 3.2 by VocalTec Inc. A recent survey of existing products is given in [VEN]. Examples of freeware packages include `nevot` [SCH2] and `vat` [JAC]. The present main attractive feature of Internet phone is that there are basically no direct charges associated with the calls that are made. One can use it to make local, long-distance, or international phone calls between computers on the Internet without any significant service charges. Systems are also being developed currently to interconnect Internet phone calls with the public switched telephone network. The present main disadvantage of internet phone is that it relies on the best effort service that is provided by the existing Internet. Consequently, the perceived quality of the voice communication is variable. Measurements on routes indicate that end-to-end Internet packet losses can be quite high [BOL,SAN]. Losses in the range of 30 - 50% are not atypical [YAJ]. The delay is also quite variable, depending not only on the existing traffic but also on the number of routers and 'firewalls' that may be traversed. When the conditions are favorable, however, near toll quality can be reached. Another disadvantage is that, except for a few experimental audio conferencing tools, most commercial products are incompatible and standards do not yet exist.

At this point in time, it is unclear what the future may hold for Internet phone. The extent to which it evolves depends, of course, on how the Internet evolves in terms of bandwidth and real-time service offerings. A variety of opinions can be heard in the press and in the computer-communications and telecommunications communities. At one extreme, Internet phone is viewed as a novelty with no real potential. At the other extreme, it is viewed as the 'third-generation' of telephony - the first generation being that based on analogue circuit-switching and the second being that based on digital circuit-switching. Whatever the opinions may be today, the reality is that Internet phone is growing in use among Internet users. It will also be offered as a 'value-added' feature by a growing number of Internet service providers (ISPs) in the near future.

In order for an ISP to offer Internet phone service at a certain level of quality either over a private 'engineered' network or subnetworks that are owned or operated by other organizations, it is necessary that one have an understanding of the limits of packet loss and delay that can reasonably be tolerated. In the case of a private network, one can then dimension and plan the network appropriately to meet the traffic demands of Internet phone customers and satisfy the loss and delay limits. In the case where an ISP is providing Internet phone service using the subnetworks of other organizations, one can then plan out a loss and delay 'budget' for the subnetworks that are to be traversed so that the end-to-end loss and delay limits are satisfied and the desired overall level of quality is achieved. In the standards making process, it is also necessary that one be able to quantify these limits.

The determination of the loss and delay limits for telephony of a reasonable quality over the Internet is not a simple problem. For circuit-switched voice, the maximum tolerable delay is well known to be 400ms or less [ITU]. This delay limit is dependent on the end-to-end delay that can be tolerated before a voice conversation becomes awkward. It also depends on the limits that can be handled by echo cancellers. In computer-to-computer Internet telephony, however, the delay and loss are variable and there is a full-duplex end-to-end channel so that there is no echo problem (in the traditional sense). Playout algorithms [MOO,RAM] at the receivers are also used to compensate to a certain extent for variable delay. The quality that is expected of Internet telephony may also be lower than that of conventional telephone service depending on the price that a user may be charged by an ISP in the future. Hence, the delay and loss limits may be significantly different from those associated with conventional public telephone quality. Moreover, Internet phone is designed to be simply a voice communication device. Therefore, it does not need to faithfully reproduce the signals required to support analogue modem and fax devices.

The delay and loss requirements of packet voice have been studied extensively in the past (see, for example, [COH,JAY,MON]). However, given the particularities of the Internet architecture, the implementation details of different products, and the recent evolution of networking and computing, it is now necessary to re-examine the issues in a more modern context. Details that now need to be considered include the packetization period, Internet architecture, protocol stack overheads of IP, UDP, and RTP [SCH4], silence/talkspurt detection algorithms, playout delay adjustment algorithms, encoding and compression schemes, and error recovery mechanisms. To evaluate the delay and loss limits for different parameter settings, it is necessary to evaluate the voice quality under a multitude of scenarios. This should be done in a controlled experiment. There are two possible approaches in which such an experiment might be carried out. In the first approach, we could build a *real-time* simulator and have pairs of users evaluate voice quality in real-time as a function of network loss and delay. In the second approach, we could build an *off-line* simulator, record voice conversations, run them through the simulator off-line for various loss and delay scenarios, and then evaluate the resulting voice quality off-line. A basic problem with the first approach is that real-time simulation is, in general, not feasible since too much processing is required if we want to include the effects of, for example, a multitude of background Internet phone calls and other data traffic, complex loss and delay processes, or multiple routers and subnetworks. The problem with the second approach is that if we simply run recorded voice conversations through an off-line simulator, then we cannot properly capture the effect that the end-to-end delay has on the interaction between the speakers.

In this paper, we develop a *two-way* voice simulation tool for Internet phone performance analysis and evaluation. The tool can be used off-line to simulate two-way phone conversations while taking into account implementation details, parameter choices, user-specifiable Internet loss and delay characteristics, other background Internet phone traffic and data traffic, and the effect of multiplexing at the access links connecting an ISP to the Internet. Since the actual simulation is done off-line, the tool is along the lines of the second approach described above. Apart from being of use to determine the delay and loss limits for Internet phone, the simulation tool can also be used for the qualitative validation of analytically based sizing models for access links and other multiplexing points. In other words, we can use the simulator to get a feeling for what real voice conversations sound like when they pass through a multiplexing point that has been sized using some mathematical model.

The principal novel contribution in the simulator is the notion of modeling explicitly the inter-speaker synchronization that exists between the talkspurts in the original unperturbed two-way voice conversation. By doing this, we are able to simulate more realistically the effect that variable end-to-end delays have on the interaction between the speakers, vary the experiment in a controlled manner, and thereby better determine in a qualitative manner the delay and loss limits for reasonable phone quality. To the best of our knowledge, such a two-way synchronized voice simulation technique has not been considered previously. Of course, the simulator has certain limitations since it cannot capture how delay and loss affects the information *content* (e.g. extra interruptions and repetitions due to excessive delay) in a two-way interaction. The idea for the two-way simulation technique was inspired by [BRA2], where an analytical Markov model is developed for talkspurt generation in a *two-way* voice conversation. This model is a generalization of the *one-way* two-state Markov model that has been adopted almost invariably in analytical packet voice studies to date (see, for example, [DAI,GRU,HEF,LEE,SRI]). The two-way model is much more complex than the one-way model but it is an attempt to capture in an analytical model the synchronization of talkspurts in a two-way conversation.

In the following section, we first provide a functional view of the Internet phone simulation tool and explain in general terms how a two-way phone conversation is synchronized in the simulation. In Section 3, we then go into more detail and describe how we first record two-way voice conversations using a pair of Sun SPARCstations and process the resulting audio files. We also explain how silence detection and packetization is carried out. In Section 4, we explain in more detail how we synchronize the talkspurts in the simulated two-way conversation. In Section 5, we provide a complete logical block diagram of the simulation tool to show how we proceed from collecting audio samples to processing the simulation output and constructing the final audio files that can be played out and evaluated. We also describe in more detail the parameters that may be changed in the simulation. In Section 6, we present some example simulation results to demonstrate two-way conversations under different loss and delay conditions. The results are in audio files that are in Sun audio file format. Pointers to download the files are located at `http://www-net.cs.umass.edu/~sbmoon/synch.html`. Finally, in Section 6, we make some concluding remarks and indicate some directions in which the present work is being extended.

2. Internet Phone Simulation Model

An overall functional view of our simulation model is provided in Fig. 1. In the figure, we have Speakers *A* and *B* at computers who conduct a two-way voice conversation using an Internet phone application. The voice signal of each speaker is sampled, encoded, and packetized by the computer. The resulting packet stream is then transmitted over the access link and the Internet. Sampling, encoding, and packetization is described in more detail in Section 3. The simulation is driven by the packet voice streams that are generated by the two speakers (see `a-a` and `b-b` in Fig. 1). These streams are generated prior to running the simulation by recording a direct two-way telephone conversation and then processing the voice signals off-line. The recording procedure and processing is described in more detail in Section 3. When the packet voice traffic arrives at its destination, the received packets are played out according to a certain playout algorithm, as shown in Fig. 1. During the simulation, we store the playout times of packets at each side (see `b-a` and `a-b` in Fig. 1). We also store the transmission times of packets at each side (see `a-a` and `b-b` in Fig. 1). The voice signals are then reconstructed off-line after the simulation is finished using the original voice samples and the stored packet transmission and playout time information.

The access links to the Internet are full-duplex. Each link is modeled by two finite buffer FIFO queues, one for each direction of transmission. The links could be, for example, T1 links with a data payload transfer rate of 1.344 Mbits/s. At the access links, we can also incorporate other background traffic. This is meant to take into account the presence of other Internet phone traffic and data traffic, such as web browsing, that is multiplexed on the same access links. For instance, an ISP multiplexes the traffic from many customers onto one or more access links to the rest of the Internet. Note that, in the model, this background traffic only passes through the FIFO queues. We do not include the background traffic in the Internet model itself since the intent of our simulation tool is to evaluate the performance of an Internet phone conversation for different access link scenarios and *existing* or *projected* Internet traffic conditions. The generation of the background traffic is described further in Section 5.

In the center of Fig. 1, we have the Internet model. In the present version of this model, the Internet impairments are represented simply by an end-to-end delay density function $D(t)$ and a loss probability P. The delay density includes propagation delays. The delay and loss processes are assumed to be independent. The impairments in the two directions are also assumed to be independent. The sampling of delays from the given density function presents a packet ordering problem that we shall revisit in Section 5.

We now explain in general terms how we model the synchronization of talkspurts in the two-way conversation. When two speakers are involved in a telephone conversation, there is inherently a great deal of synchronization between the talkspurts of the conversants. For example, (i) Speaker *A* asks a question and then Speaker *B* responds after a certain amount of time, or (ii) Speaker *B* talks and Speaker *A* then interrupts at a certain point in time during the talkspurt of Speaker B, and so on. One of the main features of our simulator is that we explicitly incorporate such synchronization in our simulation so that the effect of variable delays can be seen in the resulting two-way interactive voice conversation. This is accomplished using triggers, as illustrated in Fig. 1, where talkspurts from Speaker B arrive at the A side and trigger Speaker A to start talking (that is, sending packets) at certain points in time, ~~and~~ *vice versa*. Further details are provided in Section 4.

3. Recording and Packetization of Two-Way Voice Conversations

To record a two-way voice conversation, it is necessary that the two speakers be isolated acoustically so that the voice signals can be recorded separately without any mixing. This is done by placing the two speakers in separate offices that each contain a Sun SPARCstation and a conventional telephone. The two Suns are connected by a LAN. The two-way voice conversation between the speakers is then made using the telephones that are connected through the local PBX. At each telephone, we attach the Sun SPARCstation microphone in close proximity to the microphone in the telephone handset. The voice signal at each end can then be recorded separately. The recording should be done with no silence detection activated so that different silence detection algorithms can be applied off-line to the voice recordings, as desired.

The recording of the voice signal at each Sun can be carried out easily using `audiotool` that is available on SPARCstation 20 with Solaris 2.4. A problem with simply using `audiotool`, however, is that we need to synchronize the time at which the recordings are started on both sides since we will need to establish the timing relationships between the talkspurts of the two speakers, say Speakers 'A' and 'B'. Such a synchronization facility is not available in `audiotool`. To circumvent this problem, we can instead use `nevot` [SCH2,SCH3] to make the recordings since the source code is available and we can modify it readily to produce a timestamp at the beginning of an audio file when a recording is started. In order to synchronize the clocks on the two Suns, we use NTP [MIL1]. This protocol provides clock synchronization "to within a few tens of milliseconds" [MIL2] in the Internet of today. The procedure is then to record the voice signals on both sides using the modified version of `nevot` and then delete the timestamps and a part of one of the audiofiles so that the voice samples of the two speakers are synchronized. Let T_x denote the timestamp in the audiofile for Speaker x, let δ denote the voice sampling period, and assume without loss of generality that $T_A \leq T_B$. Then the number of samples that should be deleted from the audio file for Speaker A is $\lceil (T_B - T_A)/\delta \rceil$. This is an approximate number since NTP provides a coarser granularity of timing than audio samples, but it is a good enough approximation for the present purposes since a time shift of several milliseconds in the playout of talkspurts is imperceptible. Let the resulting synchronized audio files be denoted by `a.au` and `b.au`.

In the recording of voice traces using `nevot`, we can specify either 64 Kbit/s 8-bit PCM μ-law, 64 Kbit/s 8-bit PCM A-law, 8KHz 13Kbit/s GSM, 8KHz 32Kbit/s DVI, or

8KHz 5.6Kbit/s LPC, to name a few encoding schemes. We can also specify 16KHz and 44.1KHz encoding but the use of these schemes depends on the machine. By simply modifying the `nevot` software, virtually any other standardized or proprietary encoding and compression scheme can be incorporated in our simulation tool. Hence, the tool also enables us to compare the performance of Internet phone applications with different voice encoding and compression schemes. In the simulation runs that we have carried out so far, we have used 64 Kbit/s 8-bit PCM μ-law.

Having obtained the synchronized voice sample files `a.au` and `b.au`, the next step is to perform silence detection on these files. Silence detection is used in packet voice to reduce the number of packets that need to be sent. To detect silence periods, we first divide the sample stream into frames, where each frame consists of a set of consecutive samples that is to be packetized, as illustrated in Fig. 2. The number of samples in a frame depends on the packetization interval Δ that is selected. This parameter is a software variable that can be adjusted in our simulation tool. In [JAY], it is recommended that Δ be set in the range of 16 to 30ms. In the experiments that we have carried out so far, we have used $\Delta = 20$ms. Note that `nevot` uses $\Delta = 20$ms. Audio devices on Sun SPARCstations and SGI workstations generate a block of audio samples every 20ms if the sampling rate is set at 8KHz. This is why most voice applications use 20ms as the default packetization interval. Nevertheless, it is still adjustable. It can also be noted that 20ms of 64Kbit/s 8-bit PCM is a relatively long period for voice. It corresponds to 160 octects of voice data per packet. In contrast, we can note that in ATM there is at most only 48 octets of voice data in each cell (48 octets is an upper bound when no AAL is used). By making Δ larger, we reduce the packet transmission rate and the overall protocol overhead at the expense of an increase in packetization delay.

To decide which frames correspond to silence periods, we use the adaptive silence detection algorithm that is used in `nevot`. This algorithm uses a running threshold, based on the average sample amplitude, to decide if frames correspond to either talkspurt or silence periods. Since the silence detection is implemented as a software module in our simulation tool, we can also incorporate in our simulation tool other silence detection algorithms, such as the one in the GSM audio codec. The process of identifying silence and talkspurt frames is also illustrated in Fig. 2.

Having identified the frames that correspond to silence periods, it now remains to describe how the samples are packetized for transmission on the Internet. In certain Internet phone implementations such as `nevot`, we not only transmit frames that correspond to talkspurt periods but we may also transmit certain frames that correspond to silence periods. This is done to have a small period of background noise at the beginning and end of talkspurts so that the talkspurts sound more natural at the receiving end. The basic idea is to packetize and transmit a certain number of frames that precede and follow a talkspurt. More precisely, the procedure is as follows. Let H denote the number of frames that are to be transmitted after a talkspurt and let G denote the number of frames that are to be transmitted before a talkspurt. The parameter H is called the *hang-over*. At the end of every talkspurt, H more packets are transmitted even though they are detected as silence. When a new talkspurt begins, G packets are sent out before the first packet of the new talkspurt is transmitted. To implement this algorithm, the system must cache G frames during a silence period. The procedure is illustrated in Fig. 2 for H=2 and G=2. Note that all silence periods up to H+G frames in length are effectively eliminated. By changing H and G, we can take into account different implementation details in our simulator. `nevot`, for example, uses the default values H=2 and G=3, but these parameters can be modified.

Each frame that is to be packetized for transmission is sent in an individual packet. To this data, we add a 20 octet IP header, an 8 octet UDP header, and a 12 octet RTP header [SCH4], as illustrated in Fig. 3. This is the packetization procedure that is followed in `nevot`. With Δ = 20ms and 64Kbps 8-bit PCM, the protocol overhead is 20%. By modifying the packetization process, we can also consider other protocol stack implementations in our simulation. Note that the RTP header contains a timestamp and a sequence number that is used in the playout algorithms.

4. Synchronization of Talkspurts in Two-Way Conversation

We now describe in more detail how the talkspurts of the two speakers are synchronized in the simulation. Consider the timing diagrams in Fig. 4. These illustrate the timing between Speaker B talkspurts and how the talkspurts of Speaker B are related to those of Speaker A in the *original* two-way conversation. Each talkspurt actually consists of a packet stream at rate Δ^{-1} packets/s. In Case 1 of Fig. 4, Speaker A finishes talking and B then starts talking after t seconds. In Case 2, Speaker A starts talking and Speaker B interrupts t seconds later when Speaker B 'hears' packet x from Speaker A. In Case 3, B talks, falls silent, and then starts talking again after t seconds. In Case 4, both speakers happen to start talking at the same time, t seconds after Speaker B stopped talking. If we reverse A and B in Fig. 4, then we have timing diagrams that illustrate the timing between Speaker A talkspurts and how the talkspurts of Speaker A are related to those of Speaker B in the original conversation.

In the simulation, we use the above defined timing relationships to reproduce the synchronization that exists between talkspurts in the original voice conversation. This is done as follows. We first consider how the synchronization is performed at the side of Speaker B. There are several cases that need to be defined.

(1) *Case 1 in Fig. 4*: A talkspurt from Speaker A finishes arriving at the side of Speaker B at time t_a and Speaker B is silent. The next talkspurt from Speaker B is then *triggered* to begin at time $t + t_a$.

(2) *Case 2 in Fig. 4*: A talkspurt from Speaker A is in the process of arriving at the B side and Speaker B is silent. The arrival of packet x from Speaker A then *triggers* the transmission of the (interruption) talkspurt from Speaker B.

(3) *Case 3 in Fig. 4*: A talkspurt from Speaker *B* finishes when Speaker *A* is silent. Speaker *B* then falls silent at time t_b. The next talkspurt from Speaker *B* is then *scheduled* to be sent out starting at time $t_b + t$.

(4) *Case 4 in Fig. 4*: Both speakers are silent and then both speakers send out talkspurts at the same time. The talkspurt from Speaker *B* is then scheduled to be sent out at time $t_B + t$. This is also done in the case where $t_a > t_b$.

The synchronization performed at the side of Speaker *A* is also as described above except that we simply reverse *A* and *B*.

5. Logical Block Diagram and Model Parameters

The complete simulation model depicted in Fig. 1 has been programmed in C. We make use of C subroutines from SIMUL [SCH1] that simplify the construction of the discrete-event simulation program. Subroutines from SIMUL are also used to gather performance statistics such as the delay and loss at the finite buffer FIFO queues. A logical block diagram of the simulation tool is illustrated in Fig. 5. This shows the entire sequence of processing required from collecting voice samples to reproducing the conversation that is heard on either the *A* or *B* side.

In Fig. 5, we first use the `nevot` program to record voice samples in a file. As mentioned previously, the `nevot` software can be modified to consider different encoding schemes. The output of `nevot` is a `.tau` file. This file contains the voice samples as well as the timestamp that indicates when the recording was started. The `filter` program then takes as input the `a.tau` and `b.tau` files, removes the timestamps, and deletes a portion of one of the files, as described in Section 3, to synchronize the two voice traces. The synchronized voice samples are contained in the `.au` files. Following this, the `ts` program is run to detect the talkspurt frames in the voice sample files. The `ts` program can be modified to incorporate virtually any silence detection algorithm. The output of `ts` is a `.nevot` file. The `.nevot` file specifies which frames correspond to talkspurt periods. Following this, the `a.nevot` and `b.nevot` files are processed by `trig` to determine all the triggering and scheduling information that is required to synchronize the talkspurts at both sides in the simulation. The output of `trig` is `a.trig` and `b.trig`.

The central part of the tool is the discrete-event simulation program `2way` that simulates the queueing system depicted in Fig. 1. This program takes `a.trig` and `b.trig` as inputs and outputs the `.result` files. The file `a-a.result` is a record of the departure times of Speaker *A* packets from the *A* side. The file `a-b.result` is a record of the playout times of Speaker *B* packets at the *A* side (after the playout algorithm). The definitions of `b-b.result` and `b-a.result` are apparent. The `2way` program contains many parameters of the queueing system model that may be modified by the user. These will be described further below.

It now remains to construct audio files corresponding to `a-a.result`, `a-b.result`, `b-b.result`, and `b-a.result`. The `.result` file only contains packet playout times. Hence, it is necessary to build audio files by effectively filling the packets with voice samples. The program `repro` takes a `.result` file as input and builds an audio file using the voice samples that are contained in a `.au` file. The pair of audio files corresponding to each side is then mixed using the program `mix`. The audio file `a.b.au` is then a reproduction of the two-way conversation as seen at the *A* side. The file `b.a.au` is the two-way conversation as seen at the *A* side. These files can be listened to using `audiotool`. Note that if we listen to `a.b.au`, then Speaker *A* sounds unperturbed while Speaker *B* seems to be the only one suffering from loss and delay. This is simply due to the fact that we are listening in at the *A* side where Speaker *A* is physically located. If we listen to `b.a.au`, then Speaker *B* sounds unperturbed while Speaker *A* seems to be the only one suffering from loss and delay.

The parameters of the queueing system model in `2way` that may be modified by the user include the access link speeds, the buffer sizes of the FIFO queues, the specification of the background traffic at each access link, the Internet loss probabilities in both directions, and the delay density function in both directions. The background traffic consists of other Internet phone traffic and data traffic. The data traffic arrivals are generated by a subroutine that is user-definable. This allows us to consider virtually any arrival process for the data traffic. The packet length process for the data traffic is also user definable in a subroutine.

The background Internet phone traffic is generated by a subroutine in which we can specify an arbitrary number of Internet phone sources. The packet traffic from each phone source is synthesized by generating talkspurts and silence periods using a well-known and widely adopted two-state Markov model for voice (see, for example, [HEF,SRI]). The talkspurts are then packetized, as in Section 3. Hang-over packets and cached packets are also added in, as in Section 3. In the Markov model, the talkspurt and silence periods are independent and exponentially distributed. The mean talkspurt and silence lengths are 352ms and 650ms, respectively [SRI]. The subroutine generates the packet process for each source, superimposes the resulting packet streams, and then presents the aggregate phone traffic to the FIFO queue. We use separate and independent background Internet phone traffic generators for each FIFO queue.

The Internet loss is generated simply using an independent Bernoulli trial with probability *P*. The `2way` program can, however, be modified easily to accommodate more complex loss processes such as burst losses. The end-to-end delay is generated from a delay density function $D(t)$ that is user definable. When a packet is presented to the Internet model, the delay is drawn from the specified density function. Although this seems simple enough, there is a reordering problem. Suppose that packet i arrives at the Internet model at time t_i and the sampled delay is d_i and then packet i+1 arrives at time t_{i+1} and the sampled delay is d_{i+1}, where $t_{i+1} > t_i$. If $t_i+d_i > t_{i+1}+d_{i+1}$, then packet i+1 arrives at the destination end of the Internet before packet i. In other words, the sequencing of the packets is lost. To avoid this problem, we keep resampling d_{i+1} until $t_i+d_i > t_{i+1}+d_{i+1}$ is satisfied. The unused samples are stored for later use. When we determine the delay for subsequent packets, we first try to use the largest stored sample that does not lead to missequencing before generating any new delay samples from $D(t)$. To find the largest sample that can be used among the stored samples, we use a binary search. Since this delay generation procedure uses samples that are generated directly from $D(t)$, the marginal density function for the Internet delay is generated as specified.

6. Example Simulation Results

We now present some simulation results to demonstrate some two-way voice conversations under different delay and loss conditions. The results are in audio files that are in Sun audio file format. Pointers to download the files are located at `http://www-net.cs.umass.edu/~sbmoon/synch.html`. These files can be played out using `audiotool`. In all the examples, the access links are T1 links. We use the parameters Δ=20ms, H=2, and G=3. We consider six examples. In the first two examples, there is background phone traffic at the access links. In the remaining examples, we eliminate the background traffic completely and set the buffers at a large size so as to be able to concentrate on the effects of the Internet model impairments. In the examples, we use four different delay distributions D_1, D_2, D_3 and D_0. In D_1, the delays are in the range (0.1, 1.0) and the mean is 0.46 seconds. In D_2, the range is (0.5, 1.5) and the mean is 0.96 seconds. In D_3, the range is (1.0, 1.9) and the mean is 1.47 seconds. In D_0, there is no delay.

In the first example, the number of buffers at each FIFO queue is set at 30 packets. There are 25 background phone sources at each FIFO queue. The file

`adrian-paul.au`

is an original conversation between Adrian and Paul with no impairments. The file

`adrian-paul.delay1.0.ex1.au`

is the resulting conversation between Adrian and Paul, as heard at Adrian's side, with delay distribution D_1 and P=0%.

In the second example, the number of buffers at each FIFO queue is set at 30 packets and there are 30 background phone sources at each FIFO queue. The file

`adrian-paul.delay2.0.ex2.au`

is the conversation between Adrian and Paul, as heard at Adrian's side, with D_2 and P=0%. The respective conversations at Paul's side are in the files

`paul-adrian.delay1.0.ex1.au`, `paul-adrian.delay2.0.ex2.au`.

Listening to these audio files, we can hear that the quality of the conversation is quite good in the case of `delay1.0.ex1`. In the case of `delay2.0.ex2`, the quality is relatively bad.

In the third example, the number of buffers at each queue is set at 100 and there is no background traffic. We consider a conversation between Adrian and Sue with no delay and P=33%. The conversation at Sue's side is in the file

`sue-adrian.delay0.33.ex3.au.`

The fourth example is the same as the third except that P=50%. The conversation at Adrian's side is in the file

`adrian-sue.delay0.50.ex4.au.`

Listening to these last two files, we can hear that, even with 33% loss, the speaker is still quite intelligible. At 50% loss, however, the speaker is very difficult to understand.

In the fifth example, the number of buffers at each queue is set at 100 and there is no background traffic. The conversation is between Adrian and Paul with D_1 and P=10%. The conversation at Paul's side is in the file

`paul-adrian.delay1.10.ex5.au.`

The last example is the same as the fifth except that we use D_3 and P=25%. The conversation at Paul's side is in the file

`paul-adrian.delay3.25.ex5.au.`

Listening to these files, we can hear that the case `delay1.10.ex5` is acceptable while the case `delay3.25.ex5` is very difficult to understand.

7. Concluding Remarks

A simulation tool has been developed to evaluate off-line the performance of Internet phone applications under various scenarios and parameter settings. The simulation includes Internet access links, background Internet phone and data traffic, and Internet loss and delay modeling. The main original feature of the simulator is that it takes into account the synchronization between talkspurts in two-way conversation. Example results of two-way conversations under different delay and loss conditions may be downloaded from a web page.

There are several directions in which the present work is being extended. Firstly, we are developing a graphical user interface (GUI) for the tool. This will make the tool much easier to use, especially when one wishes to explore a large parameter space. Secondly, we are planning to use the tool to study the delay and loss limits that can be tolerated in Internet phone applications. Finally, we plan to develop a library of different Internet loss models, Internet delay models, and background data traffic models that can be used in the simulator. Much research is currently being pursued on developing loss, delay, and traffic models for the Internet. As research results become available, we plan to incorporate them in our tool. We also plan to place more example audio files in the web page as we accumulate further results.

References

[BOL] J. Bolot, End-to-End Packet Delay and Loss Behaviour in the Internet, in *Proc. ACM SIGCOMM'93*, San Francisco, CA, pp. 289-298, Sept. 1993.

[BRA1] P.T. Brady, A Statistical Analysis of On-Off Patterns in 16 Conversations, *The Bell System Technical Journal*, 47, pp. 73-91, 1968.

[BRA2] P.T. Brady, A Model for Generating On-Off Speech Patterns in Two-Way Conversation, *The Bell System Technical Journal*, 48, pp. 2445-2472, 1969.

[COH] D. Cohen, Issues in Transnet Packetized Voice Communication, *in Proc. Fifth Data Communications Symposium*, pp. 6.10-6.13, Snowbird, UT, Sept. 1977.

[DAI] J.N. Daigle, and J.D. Langford, Models for Analysis of Packet Voice Communications Systems, *IEEE Journal on Selected Areas in Communications*, 4, 6, pp. 847-855, 1996.

[GRU] J.G. Gruber, A Comparison of Measured and Calculated Speech Temporal Parameters Relevant to Speech Activity Detection, *IEEE Transactions on Communications*, 30, 4, pp. 728-738, 1982.

[HEF] H. Heffes, and D.M. Lucantoni, A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance, *IEEE J. Selected Areas in Communications*, 4, 6, pp. 856-868, 1986.

[ITU] Telecommunication Standardization Sector of ITU, *ITU-T Recommendation G.114 Technical Report*, International Telecommunication Union, March 1993.

[JAC] V. Jacobson, and S. McCanne, vat. `ftp://ftp.ee.lbl.gov/conferencing/vat/`.

[JAY] N.S. Jayant, Effects of Packet Loss on Waveform Coded Speech, in *Proc. Fifth Int. Conference on Computer Communications*, Atlanta, GA, pp. 275-280, Oct. 1980.

[LEE] H.H. Lee, and C.K. Un, A Study of On-Off Characteristics of Conversational Speech, *IEEE Transactions on Communications*, 34, 6, pp. 630-637, 1986.

[MIL1] D.L. Mills, Network Time Protocol (Version 3) Specification, Implementation and Analysis, *Network Working Group Report RFC-1305*, University of Delaware, pp. 113, March 1992.

[MIL2] D.L. Mills, Improved Algorithms for Synchronizing Computer Network Clocks, *IEEE/ACM Transactions on Networking*, 3, 3, pp. 245-254, 1995.

[MON] W. A. Montgomery, Techniques for Packet Voice Synchronization, *IEEE Journal on Selected Areas in Communications*, 6, 1, pp. 1022-1028, 1983.

[MOO] S. B. Moon, J. Kurose, and D. Towsley, Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms, to appear in *ACM/Springer Multimedia Systems*.

[RAM] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, Adaptive Playout Mechanism for Packetized Applications in Wide-Area Networks, in *Proc. of IEEE INFOCOM '94*, Toronto, Canada, pp. 680-688, June 1994.

[SAN] D. Sanghi, A.K. Agrawala, O. Gudmundsson, and B.N. Jain, Experimental Assessment of End-to-End Behaviour on Internet, in *Proc. IEEE INFOCOM'93*, San Francisco, CA, pp. 867-874, March 1993.

[SCH1] H. Schulzrinne, SIMUL Discrete Event Simulation Package, University of Massachusetts at Amherst, 1991.

[SCH2] H. Schulzrinne, Voice Communication Across the Internet: A Network Voice Terminal, *Technical Report*, TR 92-50, Dept. of Computer Science, University of Massachusetts at Amherst, July 1992.

[SCH3] H. Schulzrinne, Guide to NeVoT 3.33, 1995.

[SCH4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RFC 1889*, RTP: A Transport Protocol for Real-Time Applications, Audio-Video Transport Working Group, IETF.

[SRI] K. Sriram, and W. Whitt, Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data, *IEEE Journal on Selected Areas in Communications*, 4, 6, pp. 833-846, 1986.

[VEN] G. Venditto, Internet Phones, *Internet World*, pp. 40-52, June, 1996.

[YAJ] M. Yajnik, J. Kurose, and D. Towsley, Packet Loss Correlation in the Mbone Multicast Network, to be presented at *Global Internet Miniconference*, in conjunction with *IEEE GLOBECOM '96*, London, UK, Nov. 1996.

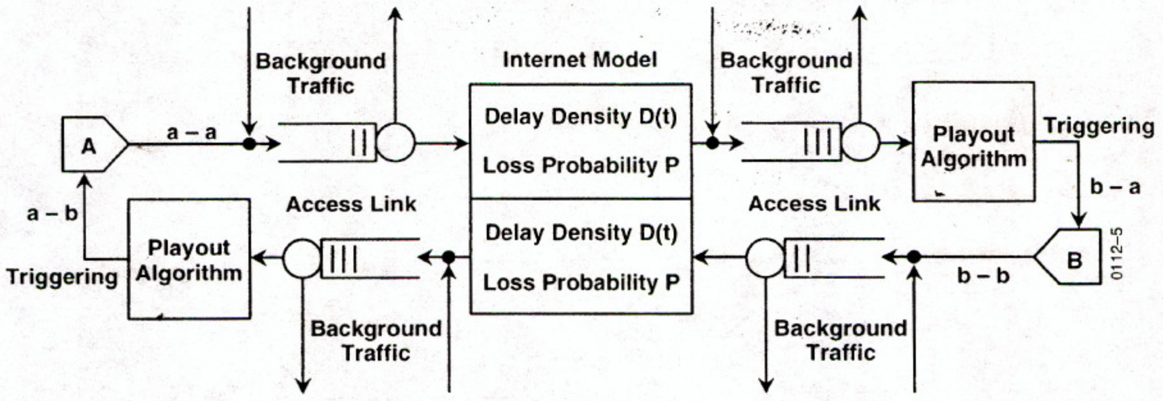

Fig. 1. Internet Phone Simulation Model

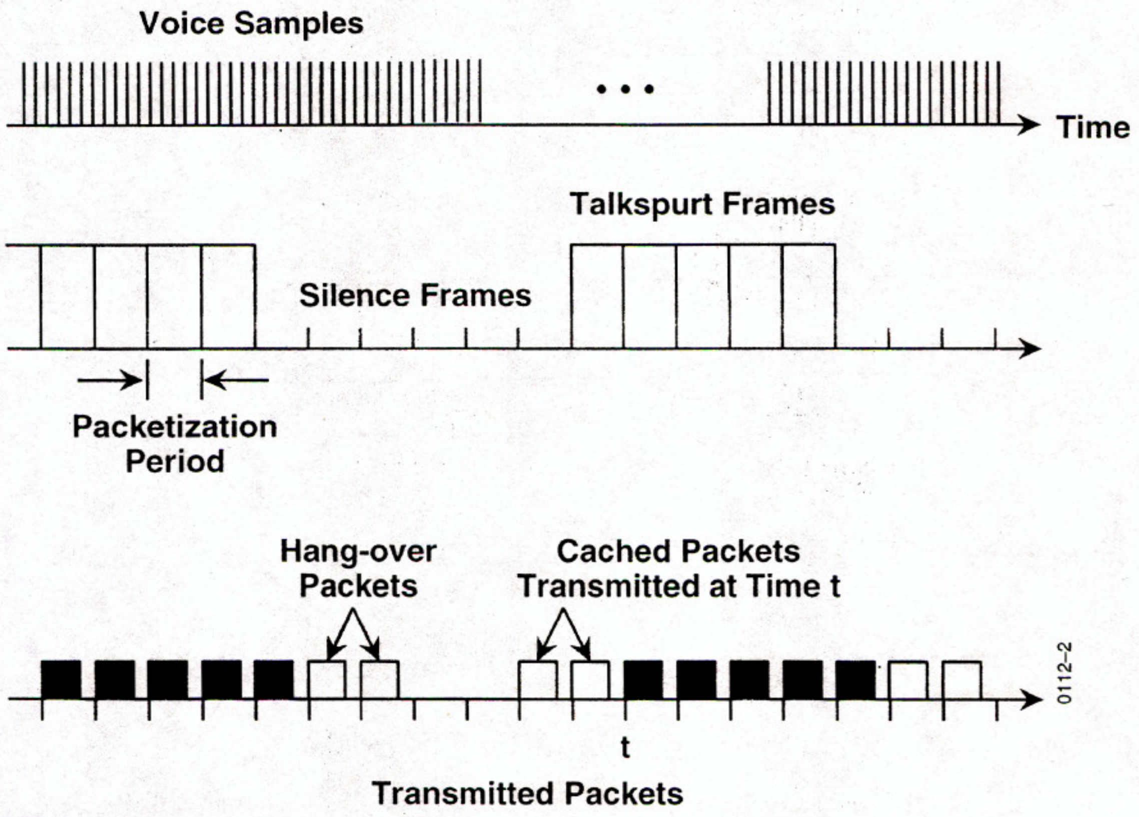

Fig. 2. Talkspurt/Silence Detection and Packetization of Voice Samples

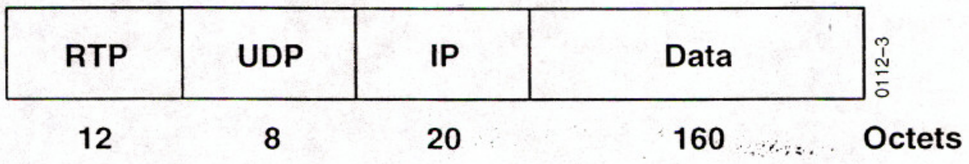

Fig. 3. Internet Packet Format and Protocol Overheads ($\Delta = 20$ms)

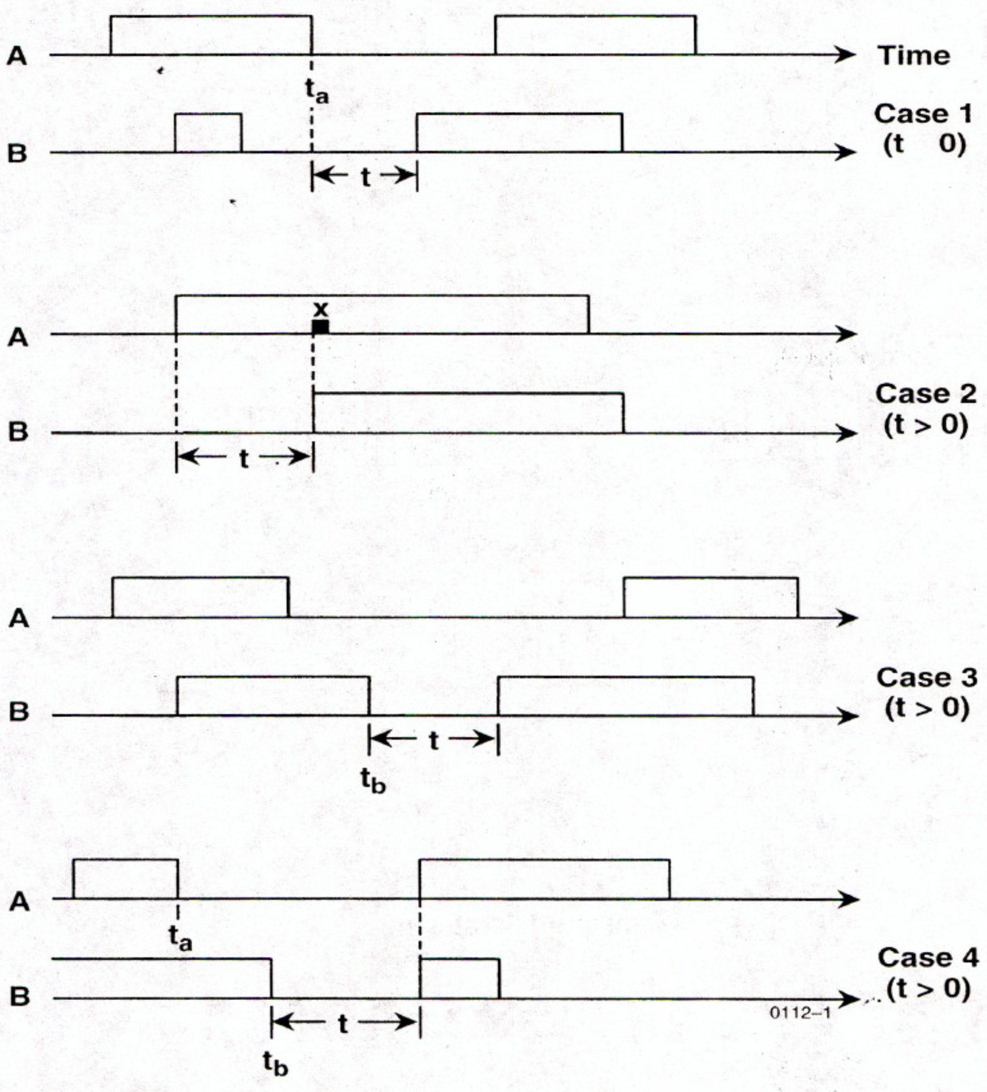

Fig. 4. Timing Diagrams for Generation of Speaker *B* Talkspurts

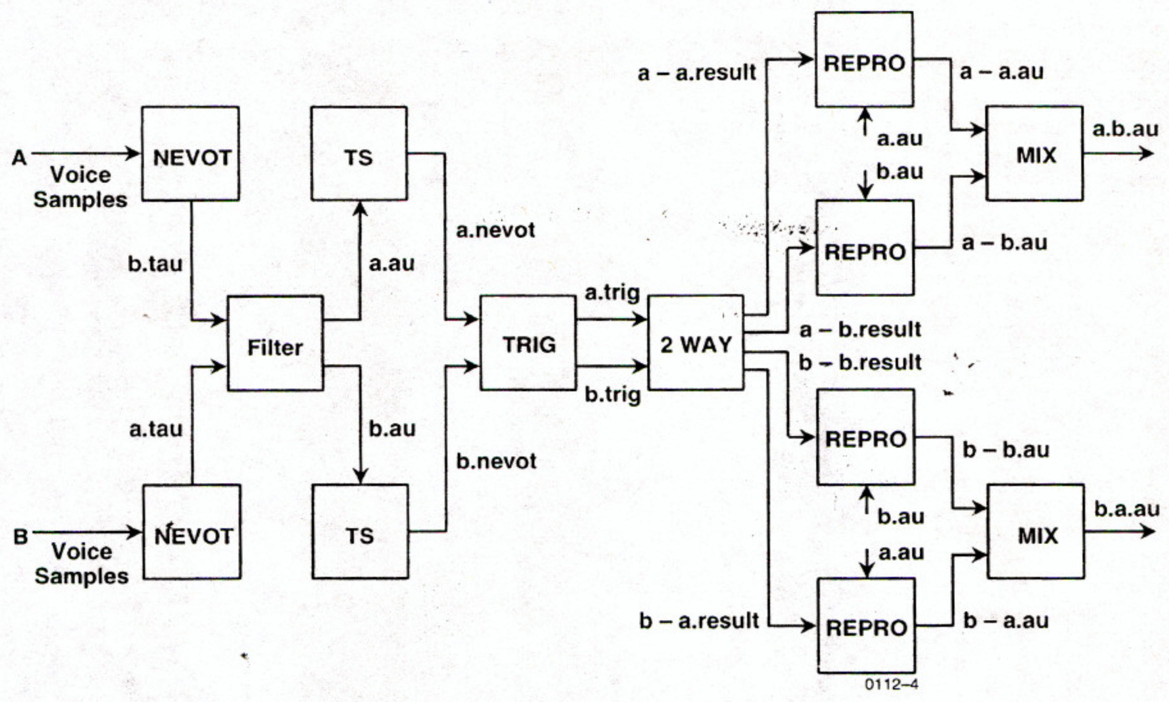

Fig. 5. Logical Block Diagram of Simulation Tool