

An ultimate encoding layer for real-time packetized voice streaming and experiments over a multi-hop wireless network

Manthos Kazantzidis
kazantz@cs.ucla.edu

Tsuwei Chen
tsuwei@cs.ucla.edu

Yuri Romanenko
yuri@cs.ucla.edu

Mario Gerla
gerla@cs.ucla.edu

UCLA
Computer Science Dept.
Wireless Adaptive Mobility Lab

Abstract

Future integrated networks are expected to offer packetized voice and other multimedia conferencing services to mobile users over wireless links. Wireless networks cannot easily support multimedia applications due to the media high probability, burstiness, persistence of errors, and delay jitter suffered by packets. This is not only due to queuing but due to link-layer (MAC) protocols as well. Mobility results in even higher loss rates and jitter for the higher layers. Adding real-time, reliability and multicast constrains to ad-hoc, multihop or cellular configurations makes the problem even more difficult. Providing for good end user perception in such environments cannot be done by applying traditional methods such as scheduling and reservations as in wired fixed networks. The high variance of the media response has to be handled by managing (monitoring and adapting to) QoS. A finer and more aggressive layering can improve perception in some applications intended for use in these or in hybrid systems. A lower layer can be defined as the minimal set of information that is needed to keep the application alive under extremely adverse network conditions. In this paper, using a real testbed and our audio on demand application ([1], [2], [3]) we propose one more possibility for the lower rate layer which is useful in speech streaming over such networks. A text transcription can be generated from the audio stream using a speech recognition engine at the sender side. The text traverses the path easier (very low bit rate) and more reliably (e.g. using redundancy). The data can be displayed at the receiver side in a caption window or, more importantly, the speech can be reproduced using the transcription with a text-to-speech synthesizer. Our experiments show that, by adapting to QoS and using our transcription scheme, end user perception can be greatly enhanced, and meaningful communication can be sustained even at most adverse network conditions.

1. Intro

As a continuation of [1], [2] and [3] we propose one more possibility for the bottom layer which is especially useful in speech streaming over wireless ad-hoc networks. A text transcription can be generated from the audio stream using a speech recognition engine. The text traverses the path easier (very low bit rate, possibility to use small packets) and more reliably (e.g. using redundancy). The data can be displayed at the receiver side in a caption window or, more importantly, the speech can be reproduced using the transcription with a text-to-speech synthesizer.

When the network conditions, as measured by the QoS monitor, suggest switching to this bottom very low bit rate layer the real-time speech stream can still be comprehensible at the client side. The switching must be properly indicated by the adaptation process, so that it is triggered at the point where the layer above it, would not produce comprehensible speech. Experiments in our wireless multi-hop testbed show that the communication quality is substantially upgraded even when adverse network conditions persist.

Our application allows the receiver to be inside a moving vehicle. The driver can listen to the transmission without ever having to look at the display. Such application becomes useful in a combat or emergency situation when information lines are down but wireless LANs can be quickly deployed. Also, think of an ambulance driver receiving traffic information from other drivers and/or a helicopter. Our caption embedding scheme can also be used when saving bandwidth is essential as for example, in wireless building conference for exchange of information among more parties than voice channels.

Initially we introduce our testbed and the environment where our experiments are performed. In section 2 issues related to the speech recognition, text-to-speech synthesizer and synchronization with the audio stream are discussed. In section 3 we present our

experiments. We conclude in section 4 and indicate future work in 5.

2. Testbed and Environment

Our multi-hop wireless testbed consists of a server, a client, a gateway and one or more interfering stations as shown in Figure 1. Both server and client run on NT/95 platform, whereas the gateway runs on Linux. A WaveLan I [5] is used for wireless networking. In order to get reproducible network conditions, both server and client can run on the same machine (without a gateway) and packet loss and delay jitter values can be simulated. In this paper we present experiments with a real multi-hop testbed.

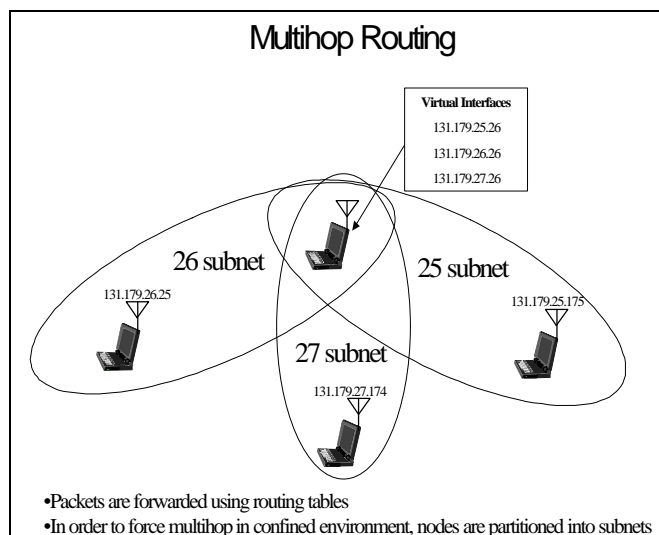


Figure 1: The topology used in the experiments. Three subnets are used to create a multihop connection. Two subnetworks are used for the server and the client respectively. One is used for the interference. The gateway has virtual interfaces to all subnets.

The server takes input from a microphone and sends speech in real time to the client via the gateway. Audio packets are sent to the client via UDP. The client application on the other end receives the audio packets and plays them out on the audio I/O device. Alternatively, the server can get the input audio stream from a file. This allows for conducting experiments easily. The playback takes place in real-time following an initial buffering delay.

When the audio stream is speech, a text transcription is generated using a speech recognition engine and sent redundantly piggybacked to the audio packets. When the server streams a pre-recorded audio file, the caption is taken from a file (pre-transcribed speech). Every audio packet sent within a time window of 2 seconds contains the text recognized in the last 2 seconds.

In order to easily conduct experiments a pre-recorded speech sample is manually transcribed in an accompanied text file. In this way the same input text stream can be used in different experiments. The text is separated in 2 seconds time windows as seen in Fig. 2.

When the speech is pre-transcribed the text is broken into 2 seconds windows and the whole window is sent with any audio packet generated in this time frame. Even if only one audio packet gets through (uniformly over time) every 2 seconds the text-to-speech synthesizer will be able to produce meaningful and continuous speech at the client side by using the text stream. When the audio stream is sent at 8KHz (8000 samples per second) mono (8-bits per sample) (7 Kbytes/sec) packetized every 240 bytes –as our adaptation process indicates for constantly high loss rates– approximately 50 packets are sent in the window of 2 seconds. This means that the caption is replicated 50 times. With an (arbitrary) average of 18 bytes of text per 2 seconds the replicated text stream reaches approximately 520 bytes/sec. These numbers can be changed to meet any audio stream to text stream bandwidth ratio or reliability requirements.

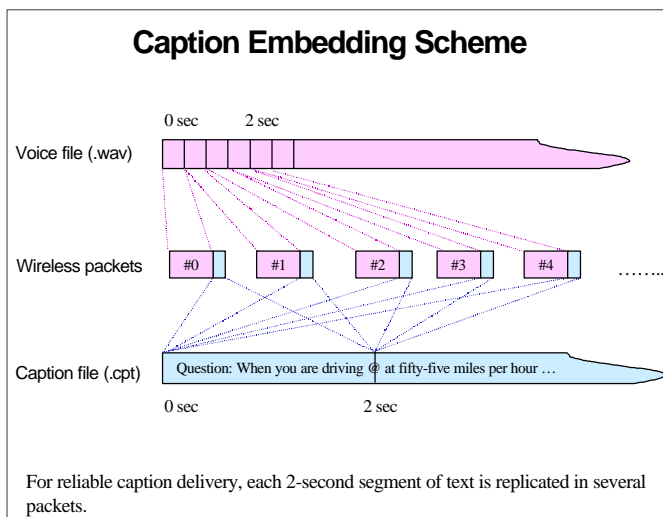


Figure 2: Pre-transcribed text is separated in 2 second time windows and sent redundantly piggybacked to the audio packets

The client constantly monitors number of packets lost as well as the delay jitter, and periodically sends QoS feedback to the server via TCP. The server switches between layers (sampling rates) of the audio stream and changes the packet size dynamically based on the QoS information received (see also [1]). When the packet loss rate is over a defined threshold the client uses the text-to-speech (TTS) synthesizer to reproduce the speech. Note that the switching between TTS and the audio stream is performed at the client side using local, up-to-date

information. It does not depend on the QoS feedback channel. Its behavior and efficiency is affected by the QoS adaptation as noted in section 4.

Any adaptation procedure can be easily incorporated in our programming model, as well as any layering/encoding. The purpose of these experiments is to validate that adaptation is important in improving quality of perception of multimedia data in wireless networks, and, that when the audio stream is voice our caption embedding scheme further improves end user perception.

2.1 Wireless Network Interface Cards

Throughout the experiments presented in this paper, WaveLAN NICs have been used connected to portable devices through PCMCIA. As in ad-hoc networks the nodes operate on a peer-to-peer level with no access point or wired system.

<i>Frequency</i>	<i>902-928 MHz (915 Mhz)</i>	
<i>Modulation Technique</i>	<i>SS DQPSK</i>	
<i>Data Rate</i>	<i>2Mb/s</i>	
<i>Media Access Protocol</i>	<i>CSMA/CA</i>	
<i>Bit Error Rate</i>	<i>$<10^{-8}$</i>	
<i>Range:</i>		
<i>Open configuration.</i>	<i>180-240m</i>	<i>600-800ft</i>
<i>Semi-Open configuration</i>	<i>60-120m</i>	<i>200-400ft</i>
<i>Closed configuration</i>	<i>27-33m</i>	<i>90-110ft</i>

Carrier Sense and Multiple Access with collision avoidance (CSMA/CA) is used in WaveLAN I as the MAC layer protocol. If the channel has been idle for longer than t_{long} , seconds, the sender waits for random time, t_{rand} and then transmits immediately, otherwise, it waits until the channel becomes idle for $t_{long} + t_{rand}$. After transmitting a packet the sender monitors the channel to make sure its transmission does not collide with others. A collision may still occur if two stations transmit a packet simultaneously. If a collision is detected, the stations involved will back off for a random period of time, according to the binary exponential algorithm. In addition to the collisions caused by two nodes transmitting simultaneously, the limit of radio transmission range unveils the effect of hidden terminal which causes collisions that can only be detected by the receiver but not by the sender. Without using a more sophisticated collision avoidance scheme (e.g. RTS/CTS used in 802.11) in MAC layer, this type of collisions further degrade the channel throughput, since the damaged packets must be handled by retransmission in either the MAC or the transport layer.

In section 4 we frequently refer to MAC layer characteristics in explaining our results.

More details and information on the lower layers characteristics and protocols can be found in [5].

3. Speech Recognition and Text-to-Speech

3.1 Speech Recognition Engine

Currently Microsoft Speech Recognition Engine and API version 3 is used to transcribe discrete speech on the fly. Discrete speech, unlike continuous, requires a speaker to pause between words. Experiments with continuous speech are currently under way.

Speech recognition technology is rapidly improving. However, current state of the art Automatic Speech Recognition cannot be expected to work sufficiently without training and fine-tuning of its parameters. The most important performance boost for the speech recognizer can be achieved by training or better yet, training for a specific grammar and/or vocabulary. In a speech recognizer, it is desirable that parameters like the following are programmable through an API. Recognition accuracy and response is affected by those. This indicates that ASR accuracy is highly dependent on the specific environment it operates: microphone type and auto-gain, expectation of echo canceling, expended energy of noise, percentage of processor time that the engine expects to use during constant speech, speaker recognition, confidence level, time that the speech-recognition engine waits before discarding an incomplete phrase or regarding a phrase as complete after the user has stopped speaking, etc.

In order to get sufficient performance out of the speech recognizer we use a limited domain grammar [4] made up of app. 1000 words and no grammar rules. Because the speech recognizer works much better for the trained words, and user-defined vocabulary training is not supported, our grammar includes words that are used by Microsoft's test sets. In this way, we can achieve an almost perfect (100%) hit rate.

In general, application specific grammars and/or vocabularies can be used to limit the speech recognition engine possible responses so that the resulting accuracy is acceptable and our captioning scheme can depend on them to reproduce the speech. For example a system used to broadcast routing information in a city has a naturally limited vocabulary.

3.2 The Text-to-Speech synthesizer

We use Microsoft's text-to-speech synthesizer contained in the Microsoft Speech Suite 3.0.

In a text-to-speech synthesizer, parameters like the following may be used to improve speech comprehension and synthesizer to real speaker synchronization: percentage of processor time that the engine expects to use during constant speech, baseline average speed, the word that is currently being played and/or the exact byte in the audio stream that is currently being played, etc. Features that support security and multimember conferencing in a TTS synthesizer are programmable parameters such as pitch of the speaking voice, personality of the voice, gender of the voice and age of the voice. By encoding the text and these

parameters, listeners can determine whether the information comes from a known trusted source.

The synchronization between switching the reproduction of speech from the text stream and the audio stream is very important in improving end-user perception. Some of the problems encountered are different when the speech is pre-transcribed rather than real-time. When the speech is pre-transcribed, recall that every packet in the 2 second window contains the caption for these 2 seconds. When the speech is real time the text piggybacked to each packet is the text recognized in the last 2 seconds of speech. The synchronization problem is particularly

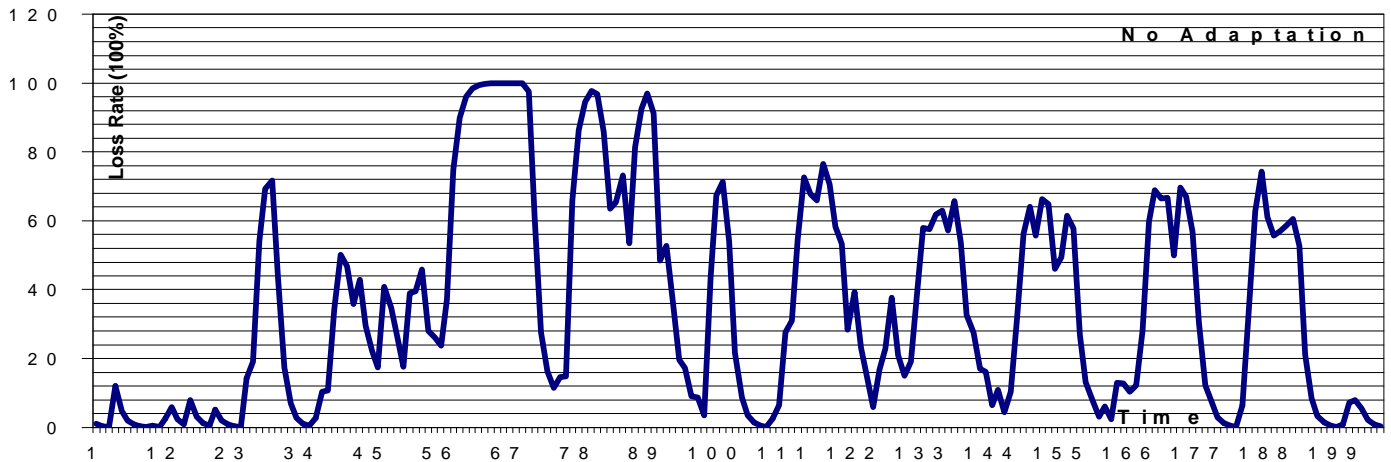


Figure 3: Loss Rate when no adaptation is performed, and with standard experiment interference. Audio stream is packetized in 960 byte packets and sampled at 22000 samples per second and 8 bit per sample. Interference is 'ping' packets of 40 bytes attempting to fill the channel

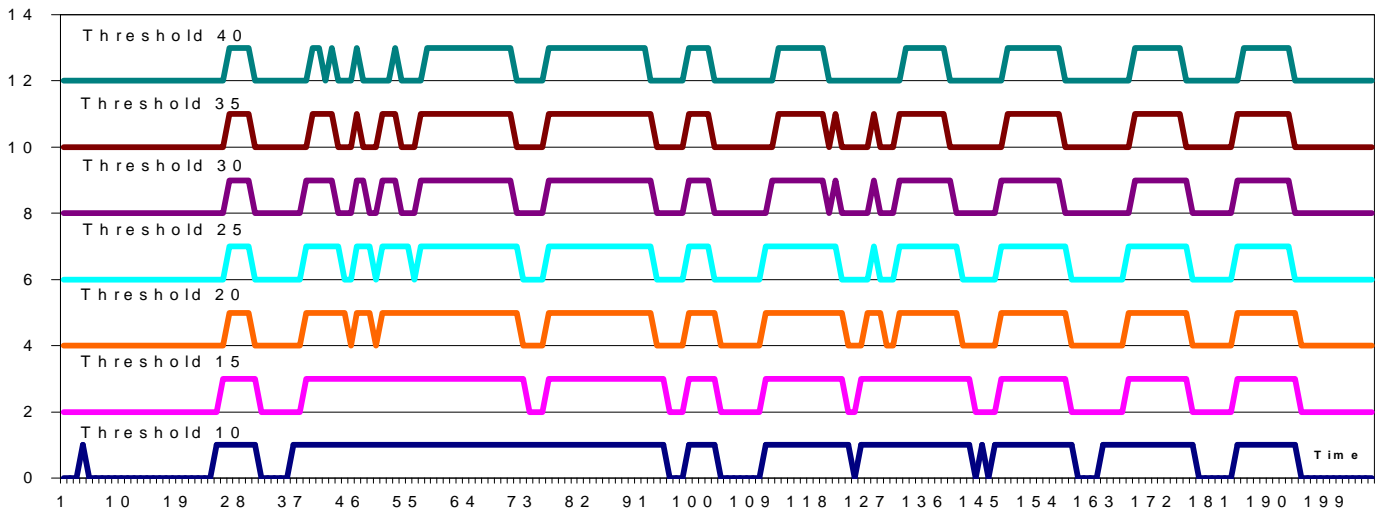


Figure 4: Visualization of text synthesizer use: switching between audio (even number) and text stream (odd number) with different TTS-thresholds in first experiment

difficult when using pre-transcribed speech. We use pre-transcribed speech for easily contacting our experiments but the system is more likely to be intended for real-time speech. However, by using a TTS speed adaptation algorithm, the synchronization between the two layers is sufficient for meaningful speech even with pre-transcribed speech.

Another issue relevant to TTS synthesizer use is setting the threshold; we expect that a dynamic computation of the threshold will produce better results. It must be set in such a way that frequent switching is avoided. Due to the CSMA/CA MAC protocol intense traffic will cause highly variable loss rates over time in a real-time application (see next section).

4. Results

Extensive experiments with pre-recorded and pre-transcribed speech have been performed. They show that incorporating transcription and using a text-to-speech synthesizer at the receiver end allows for more comprehensible speech streaming. These successful experiments indicate that similar results can be expected when the transcription is produced by a speech recognition engine on the fly.

Our goal is to sustain an audio quality high enough for meaningful communication of real-time speech. In our lab we can hear the differences. In this paper for each experiment we show the measured loss rate over time. A graph indicating the switching between the audio and the text stream accompanies the experiment. We show three experiments, one without using any adaptation mechanism which would require a QoS feedback path, one with using adaptation in the same environment and one with adaptation and more intense interference to emulate behavior in extremely adverse conditions. In this last environment adaptation is not enough to prevent corruption of the audio stream. Even at 8Khz and 240 bytes payload the loss rates are more than 70% most of the time.

In these experiments the interference –generated by a node as seen in Fig.1- is intense in order to better evaluate our scheme. The interfering station is trying to fill the channel with ping packets of 40 bytes in the first two experiments. In the third experiment in order to achieve greater interference a ping packet of 160 bytes is used to fill the channel. Furthermore the distance between the stations is increased so that the increased propagation delay will prevent the CSMA/CA from successfully detecting the busy channel. In this way the number of unsuccessful packet

transmissions is increased, resulting in a highly lossy environment.

In Fig.3 we can observe the loss rates measured by the application's QoS monitor at the receiver side. The MAC layer prevents any node from constantly transmitting, resulting in a highly variable and oscillatory loss rate for our real-time application. When the loss rate is high, the ping packets use the channel. The audio server senses the channel, finds it busy, and according to CSMA/CA (see section 1.1) delays sending the packets to the client. The client considers the packets lost when these arrive later than the playback point.

In Fig.5 we show the loss rate when adaptation in sampling rate and payload size is on. Since the interference is intense, our adaptation process switches to sending 8Khz audio packetized in 240 bytes, as opposed to 22Khz and 960 byte packets. The adaptation process used is similar to the one we describe in [1]. Here the loss rates are smaller (averaging 13 as opposed to 40) and the oscillatory nature is not present. The latter is due to the change in payload size for the audio stream. The 'ping' (interference) packet of 40 bytes and the audio packet of 240 bytes are now comparable and have a comparable probability of getting the channel and being transmitted. Furthermore the 8Khz audio stream will require the channel less frequently than the 21Khz in the first experiment. This results in more packets reaching the client before the playback point.

In Fig. 7 the loss rate observed with the high interference. The 8Khz, 240 payload audio stream is corrupted at these high loss rates averaging 77%. In this environment our captioning scheme can sustain meaningful communication. With almost cut-off loss rates of 99%, still one packet goes through in the 2 seconds time window and the conference is kept alive.

The remaining graphs (Fig. 4, 6, 8) contain the same information. They show clearer the switching between the audio and the text stream for different threshold values. A 'high; on each line indicates that the speech is reproduced from the TTS synthesizer. Different lines are shown corresponding to different thresholds from 10% (lower line) to 40%. In Fig.4 we can observe the behavior of multiplexing the layers when the loss is oscillatory (experiment with no adaptation). In Fig.6 the switching is much more frequent resulting in a degradation of the level of perception. Fig.8, where the loss rate is constantly very high even with the audio at 8KHz and 240 byte payload, shows that our scheme produced a very high level of perception in an environment where the audio stream would not be comprehensible at all.

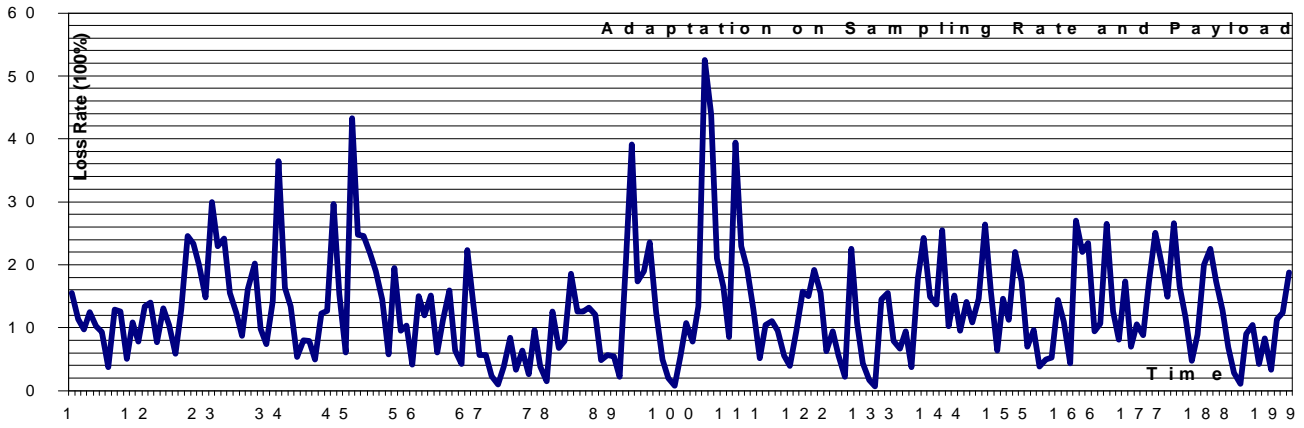


Figure 5: Loss rates when adapting to QoS, and with standard experiment interference. Audio stream is packetized in 240 byte packets and sampled at 8000 samples per second and 8 bit per sample (due to the adaptation). Interference is 'ping' packets of 40 bytes attempting to fill the channel

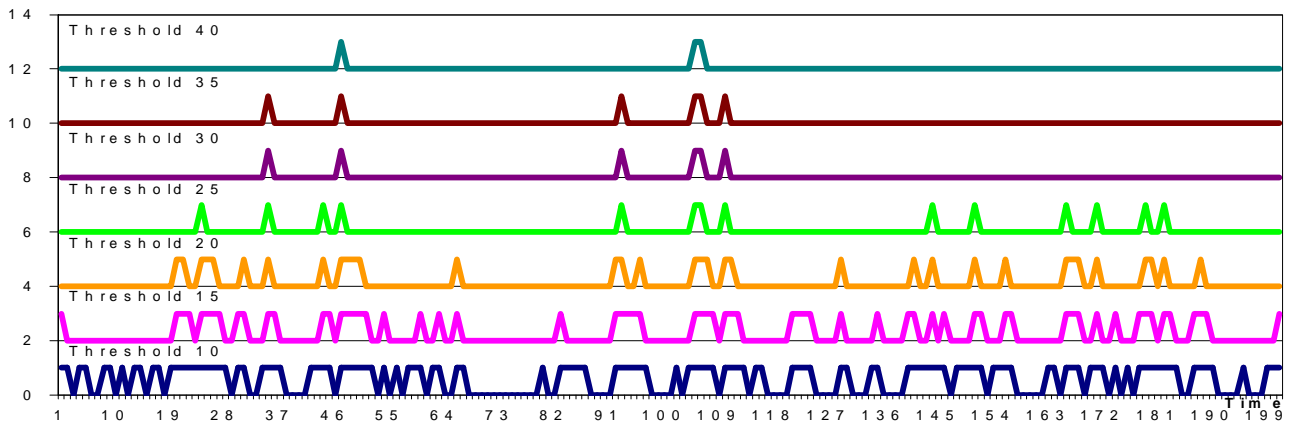


Figure 6: Visualization of text synthesizer use: switching between audio (even number) and text stream (odd number) with different TTS-thresholds in second experiment.

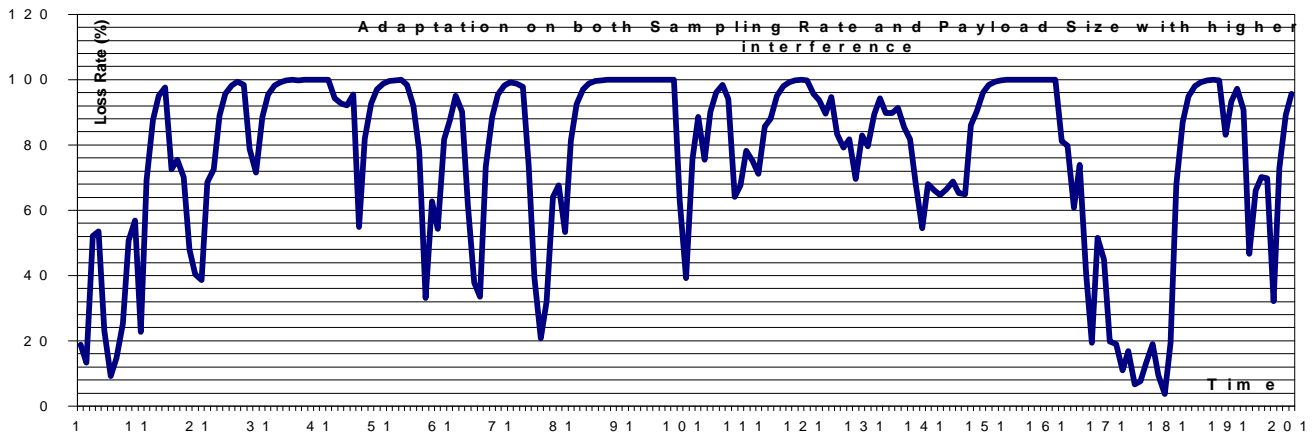


Figure 7: Loss rates when adapting to QoS with extremely adverse network conditions. Audio stream is packetized in 240 byte packets and sampled at 8000 samples per second and 8 bit per sample. Interference is 'ping' packets of 160 bytes attempting to fill the channel. The distance between the stations here is larger and the larger propagation delays result in higher loss rates

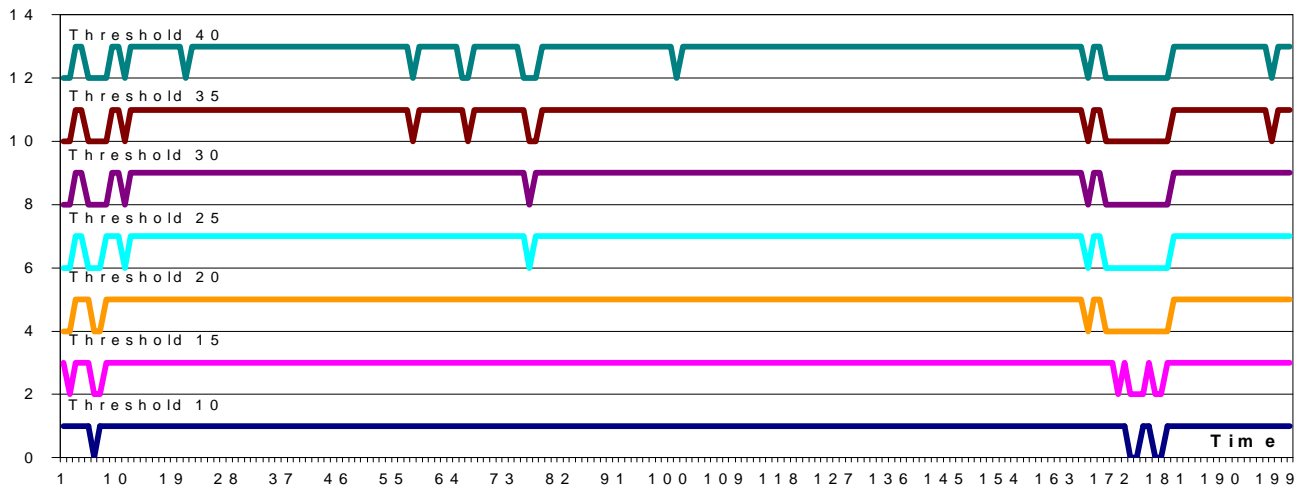


Figure 8: Visualization of text synthesizer use: switching between audio (even number) and text stream (odd number) with different TTS-thresholds in third experiment

5. Conclusion

We presented a way to stream real-time speech over very high and highly variable loss rates as those observed in networks with wireless links. Our goal is to keep a meaningful voice communication even when the sender, the receiver or intermediate hop stations are in a high interference zone. Our experiments with pre-transcribed speech show a very significant improvement in user perception. When the loss rate is constantly very high the voice conference is kept alive by the redundantly sent captioning of the speech. As speech recognition technology improves, using a text stream as an ultimate encoding layer of real-time speech will be possible without the present restrictions.

6. Future work

Future work entails constantly keeping up with speech recognition technology improvements and testing new speech engines. Features relevant to our application are continuous speech, grammar-vocabulary specific training support, sufficient performance without training, and speaker identification. Furthermore, in order to introduce adaptation points inside the network the speech recognizer will have to perform sufficiently with noise introduced by the network. On chip speech recognition would allow application of our scheme on appliances and devices.

References

[1] "Experiments on QoS Adaptation for Improving End User Speech Perception Over Multi-hop Wireless Networks" - Tsu-Wei Chen, Manthos Kazantzidis,

Mario Gerla, Ilya Slain - Quality of Service Mini-conference ICC'99

[2] "QoS Adaptation for Wireless Networks" -MS Thesis UCLA CS- Yuri Romanenko

[3] "Audio-on-Demand Application"- Manthos Kazantzidis, Yuri Romanenko, Ilya Slain

[4] Microsoft Speech Suite Documentation - www.microsoft.com/stg - Microsoft Corp.

[5] "WaveLAN -II: A High-Performance Wireless LAN for the Unlicensed Band" - Ad Kamerman and Leo Monteban - Bell Labs Technical Journal, Summer 1997 pages 118 - 133

[6] "An Extensible Framework for {RTP-based} Multimedia Applications" - J.{Du and et al.}, Network and Operating System support for Digital Audio and Video May 1997 pages 53-60

[7] "{QoS} routing performance in multi-hop, wireless networks" - T.-W. Chen and J.T. Tsai and M. Gerla,IEEE 6th ICUPC Oct 1997

[8] "End-to-End Quality of Service Control using Adaptive Applications" - D. Sisalem -IFIP Fifth International Workshop on QoS 1997

[9] "Architectural considerations for a new generation of protocols" -D. Clark and D. Tennenhouse- Computer Communication Review vol20, num4, Sep 1990, pages 200-208

[10] "{RTP}: A Transport Protocol for Real-Time Applications" -H. Schulzrinne et al.- RFC 1889 Jan 1996