ABSTRACT

Title of Thesis:	AN ALGORITHM TO EVALUATE THE ECHO SIGNAL AND THE VOICE QUALITY IN VOIP NETWORKS
	Andre Neumann Kauffman, M.S., 2006
Directed By:	Associate Professor, Steven A. Tretter, Department of Electrical and Computer Engineering

Voice over the Internet Protocol (VoIP) has been increasingly popular, but reliability and voice quality remain important factors that limit the widespread adoption of VoIP systems. Providing good voice quality is of major importance for the transition from the PSTN to VoIP networks. There are several non-real-time algorithms that estimate the voice quality such as the PESQ and the E-model. In this thesis we propose a real-time fuzzy algorithm to estimate the echo quality component of the voice quality in VoIP networks. Differently from the existing algorithms, the proposed algorithm does not need a reference signal and has low computational complexity. For these reasons, the proposed algorithm can be embedded in every VoIP system of a network to monitor live calls, giving an estimate of the instantaneous voice quality to the network provider.

AN ALGORITHM TO EVALUATE THE ECHO SIGNAL AND THE VOICE QUALITY IN VOIP NETWORKS

By

Andre Neumann Kauffman

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:
Associate Professor Steven A. Tretter, Chair
Associate Professor Carol Y. Espy-Wilson
Assistant Professor Min Wu

Acknowledgements

I'm very grateful to the Electrical and Computer Engineering Department in the University of Maryland for all their support and for making the past two years a wonderful and fascinating experience for me. I also would like to express my gratitude to Texas Instruments for providing me a Research Assistantship during part of my graduate studies. I would like to thank Dr. Bogdan Kosanovic for all his support, friendship, helpful discussions and directions that greatly influenced and improved this work. I would like to thank Professor Steven Tretter for his friendship, support and guidance during my graduate studies and during the development of this work. Finally, I would like to thank my parents for their unlimited confidence in me and for encouraging me, since the beginning, to pursue knowledge and education.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Voice Quality in VoIP Networks

Voice quality is essential in any communication system that is based on speech transmission. Voice over the Internet Protocol (VoIP) systems have been increasingly popular in the past few years and will continue to spread both in the carrier and enterprise sectors. In fact, current projections estimate that the total market value for services using VoIP is forecast to grow almost ten fold over the next five years. It is clear that VoIP will evolve from being a replacement service for the public switched telephone network (PSTN) market to providing truly converged services to the home and business.

Voice is one of the hardest services to provide on an IP network. The PSTN was built to provide an optimal service for time-sensitive voice applications, with low delay, low jitter, and constant but low bandwidth. IP networks on the other hand have been built to support non-real-time data applications such as email or file transfer. These applications are characterized by bursty traffic, with occasional peaks in demand for high bandwidth, but are not sensitive to delay. During a conversation, humans have little tolerance to delays, jitter, echo (which is a direct consequence of the delay in VoIP networks) and noise (which, for instance, can be introduced during low bit rate voice coding that is commonly implemented by VoIP systems).

In addition to the degrading factors introduced in the PSTN, VoIP networks include additional factors such as latency, delay jitter and packet loss. In order to provide a good quality of service (QoS) for VoIP networks, the existence of an embedded module that assesses the voice quality in each live call is necessary. This embedded module is the main concern of this work.

The performance of a VoIP network can be determined by a variety of parameters such as the availability of the network and dial tone, call setup request processing performance, call completion, call drop rate, one-way voice transport delay, voice quality during the call, and so on. The next table briefly discusses the service requirements that are taken into consideration when evaluating the performance of a VoIP network.

Service requirements	Parameters
Service requirements before call setup	• Availability of dial tone. • Availability of computing and network resources for honoring call processing requests.
Service requirements during call setup	• Total amount of time to setup a call (can vary from 500 ms to 10 s, depending on availability of the network). • The number of simultaneous calls that can be handled without any per call wait.
Service requirements during a VoIP session	• Voice coding and processing delay • Voice packet loss • Echo • Jitter
Service requirements after a VoIP session is complete	• Maintenance of a complete call log and call detail record (CDR).

Table 1-1 Service requirements that are taken into consideration when evaluating the performance of a VoIP network

Providing good QoS in VoIP networks is of major importance for the transition from the PSTN to VoIP networks. We are so used to the QoS provided by the PSTN that anything less than that would become a barrier to the deployment of VoIP systems and networks. The evaluation of the QoS for a VoIP system or network depends on a set of parameters and requirements that contain those described in Table 1-1. In this work we will be concerned only with a subset of the requirements described in Table 1-1. We will not analyze the requirements related to the signaling and call control protocols.

More specifically, in this work we will be interested in evaluating the service requirements during a VoIP session, that is, the voice quality over VoIP networks (here we are considering that the parameters that are associated to the requirements during a VoIP session in Table 1-1 are all grouped under what we are calling voice quality). One of the main components of the voice quality parameters is the amount of echo present in the conversation and we will discuss this to some detail and present an algorithm to evaluate how efficient is the echo cancellation (or how the echo signal is influencing the voice quality) in a VoIP call.

We can evaluate the quality of voice over IP networks in three different perspectives: the network quality, the objective quality, and the subjective quality, as illustrated by the next figure.

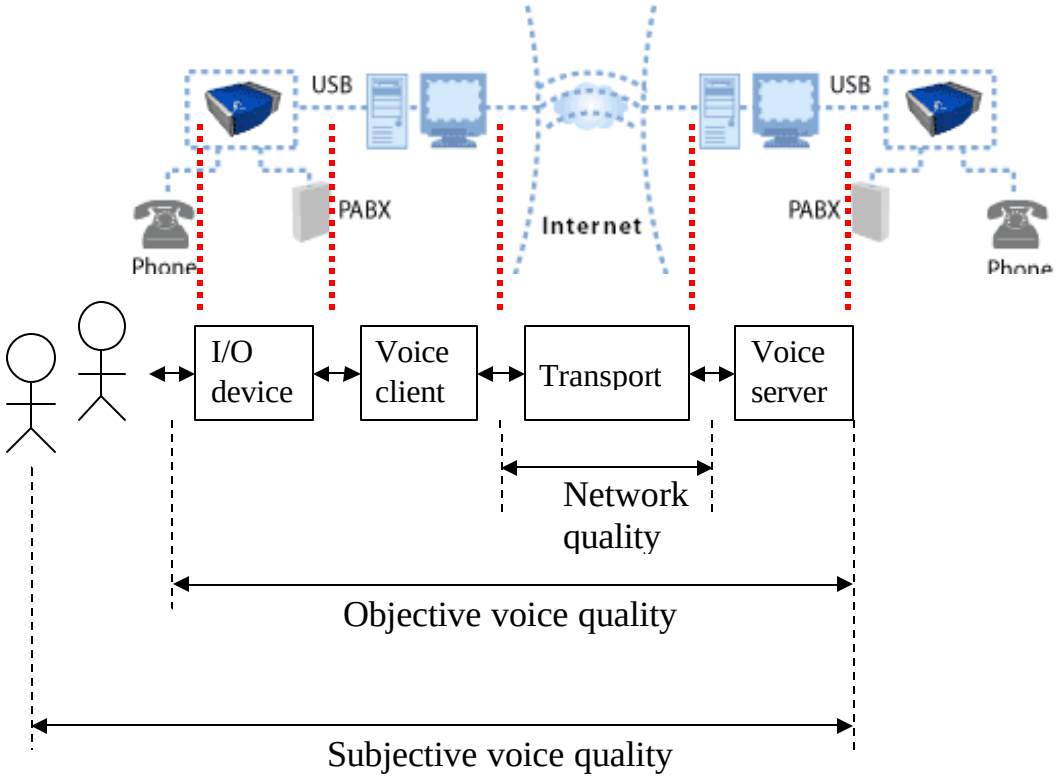


Figure 1-1 Different perspectives for voice quality evaluation in a VoIP network

The network quality reflects the provider's perspective. The objective and subjective quality reflect the customer's perspective. The network quality can be relatively easily measured by network parameters, such as the packet loss rate or packet delay or jitter. Subjective quality is generally more meaningful than network quality, as it relates directly to user-perceived quality. Assessing subjective voice quality, however, requires listening tests with a large number of test subjects. For this reason, objective quality measures that predict subjective quality are typically employed in the evaluation of voice transmission systems.

In the next sections we briefly describe the different perspectives for voice quality evaluation in a VoIP network.

1.1.1 Network Quality

In general, poor network quality decreases the performance of a VoIP system. In VoIP applications, delay, jitter and packet loss are the main network impairments that affect perceived voice quality. Jitter can be partially compensated for by using a playout buffer at the receiving end, but this introduces further delay and additional packet loss. There are several components (logical and physical) in the IP network that cause delay, jitter and packet loss. Here we briefly describe some of these

components that characterize the network quality and at the same time impact the quality of VoIP systems.

There are several components of the network that can result in delay, jitter and packet loss. Some of these components are

- Network protocols - routing protocols, traffic control protocols
- Router operation
- Bandwidth of the links
- Network reliability

Network reliability is an important component that introduces delay and packet loss, specially in the backbone of IP networks. There are two important scenarios that can directly influence the network reliability: routing reconfiguration and link failures.

- Link failure:

There are many reasons that can lead to link failures such as fiber cuts, linecard or router crashes and maintenance operations. In fact, long outage durations are typically attributed to a link failure in the IP network backbone [R 9].

- Routing reconfiguration:

It is typical for a routing protocol to require around 5 seconds to converge to a new configuration when a link goes down and around 15 seconds when a link goes up. During this reconfiguration period, forwarding may be disrupted and voice packets may be lost.

All the network behavior described above can influence the amount of delay and packet loss present in a VoIP system. When this happens, the IP network can exhibit undesirable characteristics, such as large delay spikes, periodic delay patterns and packet loss on one or more paths. All these lead to poor VoIP performance. There are experiments showing that calls using the G.711 (that is, PCM) encoder with high intrinsic quality, good echo cancellation but with some delay and packet loss are barely able to provide acceptable VoIP service (MOS > 3.6 – MOS is defined in the next section).

So, we can assess somehow the voice quality in a VoIP system by assessing the quality of the IP network, but this gives more a provider's perspective. In order to obtain a more precise evaluation of the voice quality (a user's evaluation) we need to go to the end points of the system as shown in Figure 1-1. The next sections describe the other two possible perspectives on quality in performance evaluation of a VoIP system.

1.1.2 Subjective Voice Quality

The MOS - mean opinion score - is a subjective voice quality assessment method. It is considered by many researches as the best evaluation method for assessing voice

quality because its result is based on the human direct ears. The MOS is a subjective rating system that is defined in ITU-T P.800. It is based on the opinions of several testing volunteers who listen to a sample of voice traffic and rate the quality of that transmission. The volunteers listen to a variety of voice samples and are asked to consider factor such as loss, noise and echo. The volunteers then rate the voice samples by giving a score in range from 1 to 5 as described in Table 1-2. The MOS score is calculated as an average of scores given by all listeners.

The MOS scores are defined as follows

MOS score	Description
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Table 1-2 Description of MOS scores

While MOS represents the true perceptual assessment of speech quality, it has obvious limitations. It is a time consuming process, it is not an automated method and it can not be applied to estimate the quality of a call in a real-time environment.

It is interesting to note that even using this time consuming MOS methodology, most experiments can only indicate the speech quality of unidirectional connections [R 7]. For instance, the MOS test does not indicate how the increased delay degrades the final QoS due to decreased interactivity when long transmission delays are introduced.

We will refer to the MOS score through out this work as a reference to quantify and compare the voice quality in different scenarios. This is what is normally used in research papers that assess voice quality.

1.1.3 Objective Voice Quality

One of the advantages of objective voice quality algorithms over subjective voice quality algorithms is that objective algorithms can be automated and may not require any human intervention at all. There are two main classes of objective voice quality algorithms: active and passive algorithms.

Objective voice quality monitoring, whether active or passive, has recently gained ground among VoIP providers. In active monitoring, a network analyzer injects traffic patterns that resemble a VoIP application into the network; the analyzer then observes the overall voice quality by comparing the impaired voice with the original voice sample using a perceptual model. Although this scheme can provide useful

input for optimization and network dimensioning, it uses network resources, provides non real-time results, and can't concretely determine the causes of degradation.

A passive monitoring scheme, on the other hand, can operate in real-time, and lets VoIP applications take corrective action when quality of service is unacceptable. For these reasons, the algorithm we propose in this work is a passive, objective voice quality algorithm.

Several active, objective algorithms have been proposed to automate the voice quality assessment of a call. The most successful two methods are the PAMS - Perceptual Analysis / Measurement System - and the PSQM - Perceptual Speech Quality Monitor.

In both methods a reference speech sample representing the transmitted speech signal is passed through degradation producing the degraded speech sample representing the received signal. Signal analysis is performed both on the time and frequency domains of the two speech samples and an estimate of the MOS score is provided.

The next figure is a high level view of the common concepts behind the PAMS and PSQM methods.

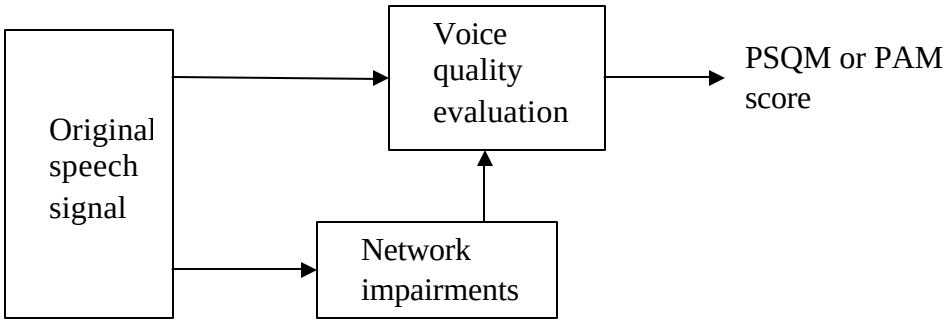


Figure 1-2 Common concepts for the PSQM and PAM methods

Despite the fact that the objective techniques described above are automated and they simplify the voice quality evaluation process compared to the MOS method, they are considered intrusive or active. This is due to the assumption that the reference speech sample representing the transmitted signal is available. This is, in general, not true in live communication calls.

It is worth noting that the ITU-T Recommendation P.861 specifies a model to map audio signals to their representation inside the head of a human. The basic idea of the modeling approach is to take measurements of the processed (compressed, encoded, etc) signal, perform an objective analysis between the original and the processed

version and offer an "opinion" as to the "goodness" of the signal. The result is an absolute number.

The challenges of measuring voice quality in real-life situations are a little more complex, and more data is often necessary than derived in Recommendation P.861. We can see this complexity in the block diagram description of the PSQM algorithm in Figure 1-3.

The PSQM algorithm derives objective numbers that are an estimate of the quality of the voice being delivered. The PSQM algorithm uses several steps in processing the input and output signals. The next figure shows a block diagram with the processing steps for the PSQM algorithm - extracted from [R 6]. In this block diagram, x[n] is the input signal (reference) and y[n] is a scaled version of the output signal. The PSQM also requires time-aligned input and output stream samples which maybe difficult to obtain in practice because this requires a precise knowledge of the delay that affected the output signal.

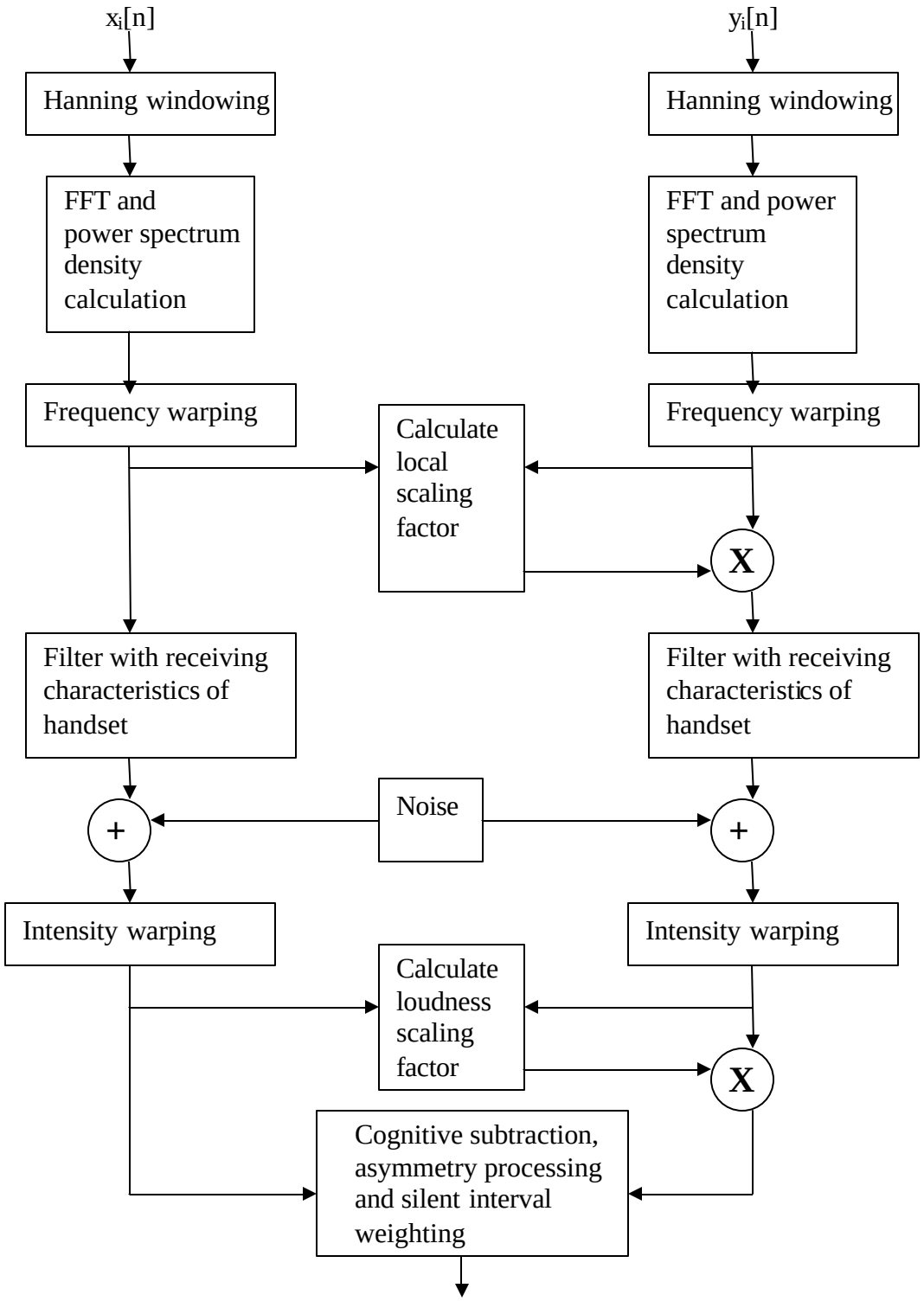


Figure 1-3 Block diagram of the PSQM algorithm

We note that the PSQM, besides requiring a reference signal and time-alignment of the signals, also requires a considerable computational power in order to perform operations such as FFTs and filtering. This is a heavy requirement to be posed to embedded systems that are monitoring live calls, especially systems with a high density of channels (which sometimes have hundreds of channels being processed by a few DSPs in a single platform). This algorithm is just not applicable in this scenario. On the other hand, our proposed algorithm is not as complete as the PSQM, but offers a tradeoff between computational requirements and accuracy of the estimation of the voice quality.

As a result of the combinations of some ideas of the PSQM and PAMS methods, recently the ITU-T created the PESQ - Perceptual Evaluation of Speech Quality - in Recommendation P.862.

In order to provide a voice quality measure in real-time for ongoing calls, non-intrusive (or passive), objective approaches have also been proposed. One such methodology relies on measuring various parameters that can affect the voice quality in the IP network. Those parameters are packet loss, latency and delay jitter. An example of this approach is the E-model of the ITU-T Recommendation G.107. In this method the two ends of a VoIP call can exchange information on these instantaneous parameters and the voice quality measure can be obtained. However, in this method the actual voice is not used and for this reason the results will not be accurate enough. Most current methods [R 33] use the E-model to measure voice quality, but the E-model requires subjective tests to derive model parameters which is time-consuming and often impractical. A real life VoIP network may cross several countries and in this case, the E-model will need to be trained for each country. As a result, the E-model is only applicable to a limited number of codecs and network conditions.

It is important to note here that all algorithms that were described (even the objective ones) have some kind of heuristic motivation. For instance, in the PSQM algorithm the "silent interval weighting" operation (Figure 1-3) is believed to allow a fitting of the cognitive processing to cultural differences. Changing the way this operation is done may result in more precise voice quality estimation in different cultures. In the same way, some of the reasoning behind our proposed algorithm is based on heuristics, for instance, on how humans perceive echo.

1.2 Contributions

As was stated in the beginning of this chapter, voice quality is essential in any communication system that is based on speech transmission. We also emphasized that providing good QoS in VoIP networks is of major importance for the transition from the PSTN to VoIP networks. As was seen in Table 1-1, there are many parameters that should be controlled in order to provide a good QoS in a VoIP network or

system. In this work we will focus on the voice quality portion of the QoS set of requirements. We will propose a real-time algorithm to evaluate the voice quality in a VoIP system or network. We will show the details and implementation results of one building block of such algorithm. Specifically, we will focus on the building block that evaluates the echo component of the voice quality in VoIP networks.

The state-of-the-art in the subject of evaluating the voice quality was briefly highlighted in this chapter. As is stated in [R 8], the objective assessment methods such as PSQM, PAM are mainly developed for the evaluation of the speech codec performance and are not fitted for the delayed and jittered speech signal. Another trend of algorithms to estimate voice quality is represented here by the E-model (Section 1.1.3), which requires parameters that depend on the telephone terminal and are difficult to be obtained. The conclusion is that "there is a need for simple speech objective evaluation methods" [R 8].

In this work we propose a real-time, low computational complexity fuzzy inference system to evaluate the echo component of the voice quality over VoIP networks (Figure 4-2). We also propose extensions to be incorporated in the algorithm in order to obtain the overall performance of the VoIP system or network (Figure 4-3).

We suggest a simpler (compared to PSQM, PESQ or the E-model) objective voice quality evaluation method which divides the voice quality assessment into three main engines that separately compute the contributions of the three main factors that affect the voice quality – delay, jitter and echo. It should be clear that the main developments in this work are done for the echo component of the voice quality.

The use of playout buffers at the receiving side of a VoIP call can be used to compensate for the effects of jitter based on a tradeoff between delay and packet loss [R 10]. However, as we can see in Figure 2-1, the effect of short delay (less than 300 ms) in the voice quality when echo is not present is minimal. If we can eliminate jitter (at the same time decreasing the rate of packet loss - due to jitter compensation) by increasing the delay and keeping the echo over control, the final voice quality of the call should be better than dealing with jitter in the conversation. This is one of the reasons why we emphasize the study of the echo quality parameter for estimating the voice quality in VoIP networks.

Of course these elements are interconnected in non-trivial, non-linear ways. For instance, jitter requires a jitter buffer (playout buffer), which causes delay and delay may amplify some existing echo that is generated by speech signal reflection somewhere in the network. Our algorithm doesn't try to evaluate precisely each separate contribution, nor does it try to precisely model how these components interact. We use fuzzy logic inference systems separately for each element (echo, jitter, delay) to try to approximate their contribution and also the final voice quality assessment.

Besides using fuzzy logic, which already requires low computational effort, we try to use as inputs for our fuzzy inference systems parameters that are "free" of computational effort once they are already available in the system for some other purposes (e.g. the echo canceller has to estimate the loss in the echo path and we use this estimation as an input to our fuzzy inference system that estimates the echo component of the voice quality).

While most of the existing voice quality assessment techniques are active and can not be applied to real-time analysis in live calls, our proposed algorithm does not interfere with the call or the signals present in the network. This characteristic of being able to analyze live calls with low computational complexity is the main advantage of our proposed method over the methods described in this chapter. The main disadvantages of our proposed algorithm, as we will show in Chapter 4, are the lack of accuracy (usual fuzziness introduced by the fuzzy engine) and the need for calibration of thresholds of the algorithm for each specific VoIP system or network.

Finally, we would like to give an idea of where our algorithm fits in the bigger picture of evaluating the performance of VoIP networks, that is, not only the voice quality, but the performance of the whole VoIP system or network. As was described in Table 1-1, we can split the analysis of the performance of a VoIP system with respect to the various stages of a call over IP networks. We have performance requirements before the call is set up, during the setup, during the VoIP session and after the VoIP session. For each one of these steps that define a VoIP call, we have a set of parameters that are used to evaluate the performance of the specific step. In the next diagram we show how our proposed algorithm fits in the larger view of analyzing the performance of a VoIP system.

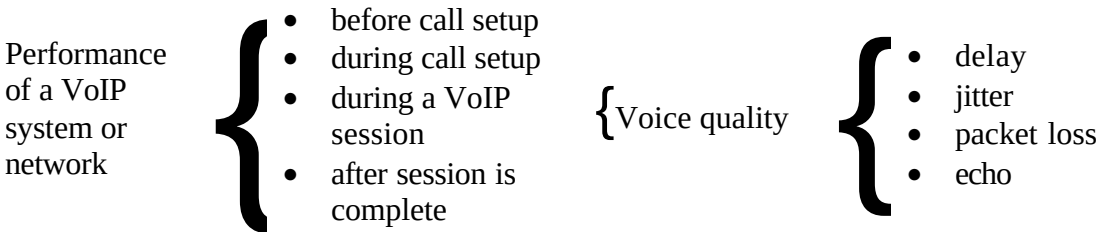


Figure 1-4 Components required for evaluating the performance of a VoIP network, highlighting where the VoIP voice quality fits in the bigger picture

1.3 Thesis Outline

In Chapter 2 we describe how echo is created in VoIP networks and the relationship between echo control and voice quality. The description of the echo problem given in this chapter is important because the main result of this work is an algorithm to

evaluate the echo component of the voice quality in VoIP systems or networks. In Chapter 3 we give an overview of fuzzy logic and soft computing, which was the methodology used in the development and implementation of our proposed algorithm. Chapter 4 is our original contribution and the main result of this work. It describes the ideas, implementation and simulation results of our proposed algorithm. We present our conclusions and some directions for further work in Chapter 5.

2 Echo Control and Voice Quality

2.1 Echo in Telecommunications Networks

In most cases our everyday conversations take place in the presence of echoes. We hear echoes of our speech waves as they are reflected for instance from the floor and the walls. However, if the reflected waves arrive shortly after we speak them, we do not perceive them as echo but as some reverberation. On the other hand if the reflected wave takes 20 or 30 milliseconds (ms) to come back to us, we will identify it as an annoying echo.

Similarly, in telecommunications networks echo can also be quite annoying and, if left uncontrolled, can make it impossible to carry on a conversation. Hearing your own voice in the receiver while you are talking is common and reassuring to the speaker. Hearing your own voice in the receiver after a delay of more than about 25 ms, however, can cause interruptions and can break the cadence in a conversation.

Whether a caller hears echo is chiefly dependent on the amount of delay present in the circuit or network. Most callers will hear echo of their own voice if the circuit contains as little as 30 milliseconds of round-trip delay. If the round-trip delay approaches 50 ms, virtually all callers will complain of echo if it is left uncontrolled.

2.1.1 Network Delay Makes Echo Noticeable

Delay is introduced into the telecommunications network primarily by transmission facilities and transmission equipment. Negligible delay is introduced into the telecommunications network by some types of transmission equipment, such as a digital switch. Other transmission equipment, such as low bit rate voice encoders, often introduces significant delays. Depending on the network topology, and the type of transmission equipment used in the network, 30 ms of roundtrip delay can occur in connections that are across country or just across town.

The next table depicts some typical transmission facility delays.

Transmission facility	Delay per 100 miles
T1 carrier over copper	1 ms
Fiber optic cable	1 ms
Microwave radio	0.7 ms

Table 2-1 Some typical transmission facility delays

Even when echo control is present, there is a limit for the amount of delay that will be tolerated by an average user. The ITU-T Recommendation G.114 provides limits for one-way transmission time (delay) on connections with adequately controlled echo.

One-way transmission time	User acceptance
0 to 150 ms	Acceptable for most users
150 to 400 ms	Acceptable but has impact
400ms and above	Unacceptable

Table 2-2 ITU-T limits for one-way transmission time (delay) with echo control

The next figure shows how for a specific network configuration the quality of a call degrades with increasing delay. The figure also shows how this degradation is further affected by the presence of echo.

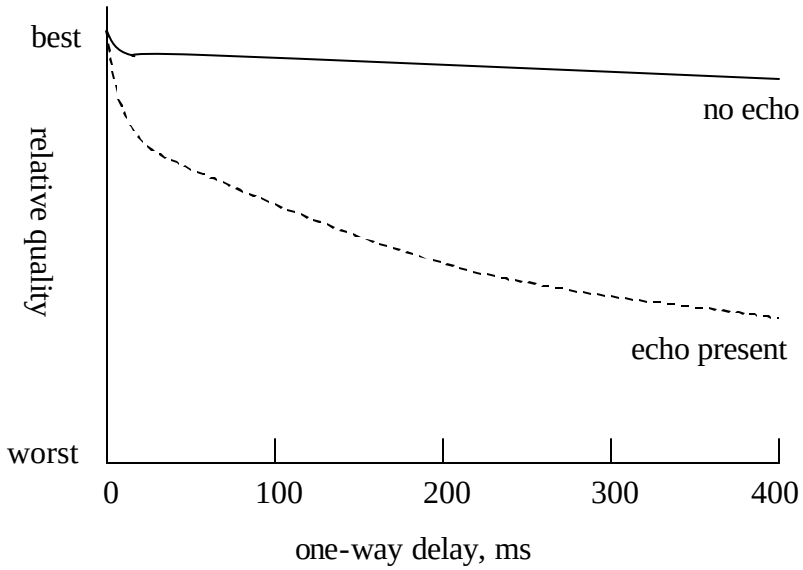


Figure 2-1 Impact of delay on call quality with and without echo

It is clear from the figure above that echo is a determinant component in decreasing the quality of the call. More specifically, echo has two drawbacks: it can be loud and it can be long. The louder and longer the echo, the more annoying it becomes.

So far, we have analyzed the phenomenon of echo in telecommunications networks in general. The next sections discuss in more detail what causes delay in VoIP networks (which is important because echo is noticeable only when delay is present) and we also discuss the echo problem specifics for VoIP networks.

2.1.2 Delay in VoIP Networks

Differently from PSTN networks, where delay is in large part due to the propagation delay in the transmission facilities, which means that short distance calls rarely suffer from delays, VoIP networks have delay injected by several reasons as we describe next.

Traditionally used in the PSTN network, the pulse code modulation (PCM) method of encoding voice signal (as defined in ITU-T's G.711 standard) generates a bit stream of 64Kbps. On the other hand, in VoIP applications low bit rate voice encoding algorithms are widely used. For instance, the most popular frame-based vocoders that utilize linear prediction are the G.723 standard, generating a bit stream of 5.3 Kbps, and the G.729 standard, producing a bitstream of 8Kbps.

However, this reduction in rate using vocoders does not come for free. There is a coding delay associated to each vocoder, for instance, in G.723 systems there is approximately a 37.5 ms delay due to the algorithmic portion of codec delay (the coder process the voice signal in 30 ms frames). As we will see, this delay when added to other delays introduced by the network will result in an end-to-end delay that greatly increases the perception of echo in VoIP networks.

Buffers are another cause of delay in VoIP networks. IP based networks employ buffers for several reasons. At the access domain, a buffer provides temporary storage for packets before they are routed to the appropriate transport network. The amount of delay suffered by packets at this level of the network depends on buffer size, traffic density and packet priority. At the transport domain, buffering is needed to support proper routing and multiplexing of packets. In this domain, the total amount of delay depends on several aspects such as propagation time, transmission capacity and header processing delays. Finally, at the packet delivery domain, the packets that arrive earlier than the expected time need to be stored temporarily before being delivered. For VoIP applications, delayed packets may become useless after a specified amount of time. The delay jitter buffer holds these packets that arrived earlier and also delayed packets in an attempt to neutralize the effects of packet inter-arrival jitter. This helps maintaining the liveliness of real-time communication over IP networks, increasing the voice quality. These playout buffers at the receiving side of a VoIP call can be used to compensate for the effects of jitter based on a tradeoff between delay and packet loss.

Besides these "designed" delays described above, IP networks are susceptible to several network scenarios that can drastically increase the amount of delay in one or more paths of the network. These scenarios, such as link failure and routing reconfiguration, were described in Section 1.1.1.

In an ideal VoIP network we would have a one-way delay that would be less than 150 ms.

2.1.3 Echo in VoIP Networks

As described in the previous section, in VoIP systems the delays introduced by coding the speech into packets and removing network jitter are long enough to make the system susceptible to echo problems even for short distance calls. Echo cancellation is therefore likely to be needed in most VoIP systems. This is in contrast to the PSTN where echo cancellation is only necessary on long-haul connections.

In general, short-delay echoes are rarely distinguished from side-tone unless either the round-trip delay exceeds 30 ms or the echo level is extremely high. For this reason echo cancellation is not required on short PSTN connections. However, round-trip delays of VoIP systems are unlikely to be less than 30 ms, ensuring that some form of echo cancellation is invariably required.

If a VoIP system connects to a local PSTN, echo cancellation is probably needed to cancel the local hybrid reflections. If the system does not connect to a local PSTN, echo cancellation should still be included to remove any acoustic echo.

As was mentioned in the last paragraph, in general there are two possible origins for echo in telecommunications networks. Echo can originate from hybrid reflections in the network or from poor acoustic isolation. Depending on how the echo originated it is called line echo or acoustic echo. These two types of echo are described in Sections 2.2 and 2.3 respectively.

2.2 Line Echo

In a traditional network, line echo is caused by a mismatch in impedance from the four-wire network switch conversion to the two-wire local loop. Echo in the PSTN is regulated with echo cancellers and a tight control on impedance mismatches at the common reflection points.

The 2-wire local loop consists of a single pair of wires that carry both directions of the conversation. At the local telephone exchange, this 2-wire pair is connected to a 4-wire trunk by using a device called a hybrid. The hybrid splits the 2-wire local loop into two separate pairs of wires, one for the send path and one for the receive path as described by the following figure.

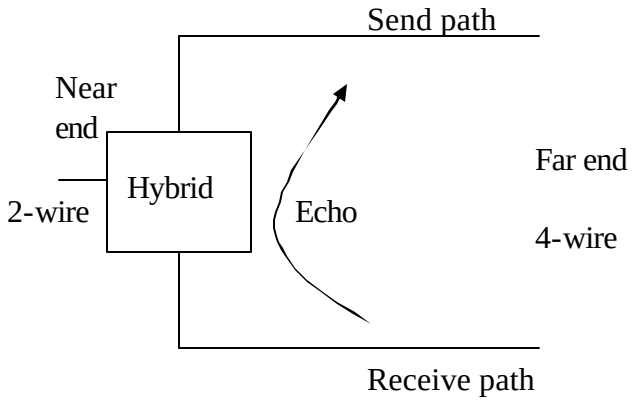


Figure 2-2 The hybrid device and line echo generation

Because the hybrid cannot be made to split the 2-wire loop perfectly, some of the receive signal is erroneously leaked into the send path and is called echo.

Telephony networks in parts of the world where analog voice is primarily used employ echo suppressors. This is not the best mechanism to use to remove echo. For instance, a line that has an echo suppressor can not use ISDN because the echo suppressor cuts off the frequency range that is used by the ISDN. Our proposed algorithm does not deal with echo suppressors.

On the other hand, in IP networks, echo cancellers can be built into the codecs and operate on each DSP. In our proposed algorithm we take advantage of the measurements made by those echo cancellers present in the DSP to draw conclusions about the echo quality in the call and more generally the voice quality of the call. Note that once the echo canceller has already computed such measurements (that will be describe in more detail on Chapter 4) there is no extra computational effort required by the algorithm for the DSP.

2.2.1 Line Echo in VoIP Networks

In VoIP networks line echo is generated from the telephone network (PSTN) toward the packet network. Normally the IP portion of the VoIP solution adds more than 50 ms of round trip delay and for this reason line echo cancellers are essential for VoIP networks when they interface with the PSTN. The echo-cancellation tail length varies among different VoIP applications. The tail-length requirement is determined by the distance between the gateway equipment and the four-to-two line hybrid. Typically this ranges from an 8 ms tail length for residential applications to 128 ms tail length for carrier applications.

The following figure is a very simplified block diagram of a TDM-IP gateway with a line echo canceller.

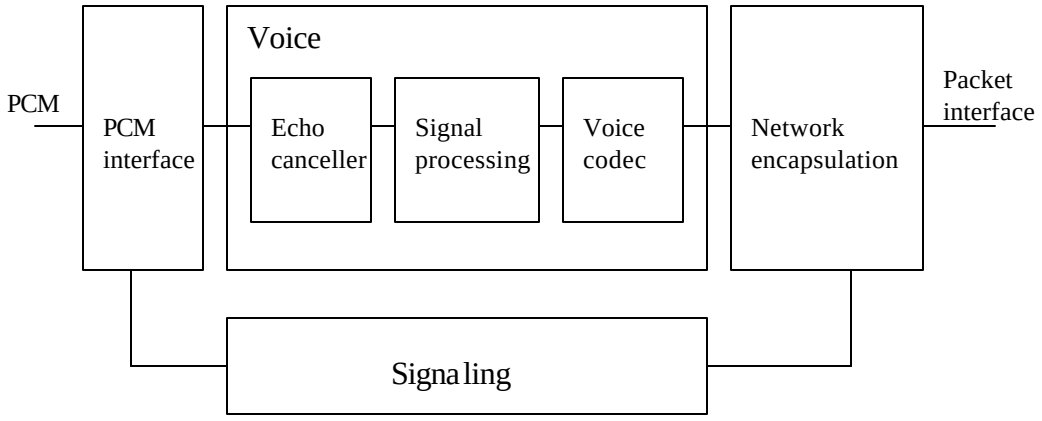


Figure 2-3 Simplified block diagram of a TDM-IP gateway

2.3 Acoustic Echo

While not as prevalent as echo caused by the hybrid (line echo), acoustical echo can also be encountered in the telecommunications networks. Acoustical echo is caused by poor isolation between the microphone and speaker of some telephone sets. Most hands free speakerphone systems incorporate special echo control circuitry to ensure that echo is not a problem. Another example is the need for acoustic echo cancellation to protect the landline subscriber from acoustic echo originating from digital wireless networks.

In the case of VoIP networks, acoustic echo is normally present when at least one of the callers is using a computer with a loudspeaker and a microphone.

As is the case for line echo, acoustic echo becomes audible when there is long delay. On the other hand, differently from line echo, acoustic echo usually is not severe enough to make the conversation impossible.

The methodology for canceling acoustic echo differs in many aspects from the methodology used for canceling line echo. In this work we will not be dealing with acoustic echo in VoIP networks. Our proposed algorithm is valid only for line echo signals and in the next section we describe with more details the ideas behind the line echo cancellation.

2.4 Line Echo Cancellation

In this section we describe an overview of the building blocks of a line echo canceller. Our proposed algorithm to evaluate the echo component of the voice quality is based in measurements realized by the echo canceller. This section also

defines the notation and some parameters that will be used when we describe our algorithm.

In this work we will adopt the notation used by the ITU-T Recommendation G.165 [R 35] and by most books and articles on echo cancellation. A line echo canceller has four ports, two on the near end side and two on the far end side. The four ports are described in the next figure, which was again extracted from ITU-T Recommendation G.165.

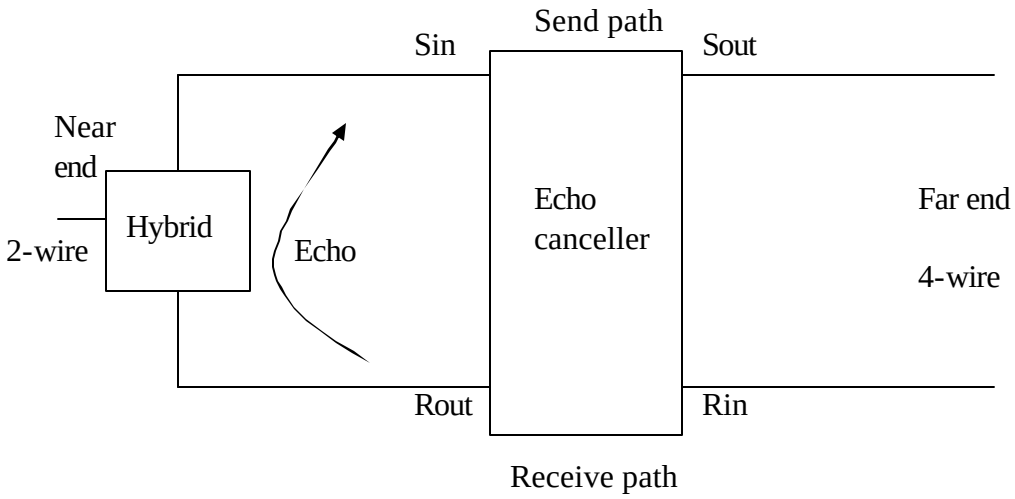


Figure 2-4 Circuit with a line echo canceller

The four ports of the echo canceller are denoted as follows:

- Receive-in (Rin)
- Receive-out (Rout)
- Send-in (Sin)
- Send-out (Sout).

An echo canceller monitors speech from the far end that passes through its receive path and uses this information to compute an estimate of the echo that is then subtracted from its send path. If the estimation is good, the echo is cancelled and only the near end speech is sent to the far end. Good echo cancellation is essential for the quality of the voice in the network.

Echo cancellation occurs between the send-in and send-out ports, reducing the echo present in the send path. The total amount of echo attenuation that an echo canceller provides is called echo return loss enhancement (ERLE). ERLE is the difference in the echo level between the send-in and send-out ports and it is measured in dB.

An echo canceller normally consists of three major building blocks:

• Adaptive filter
• Double-talk detector
• Nonlinear processor

In the next figure, we expand the echo canceller that was represented in Figure 2-4 with its major building blocks listed above.

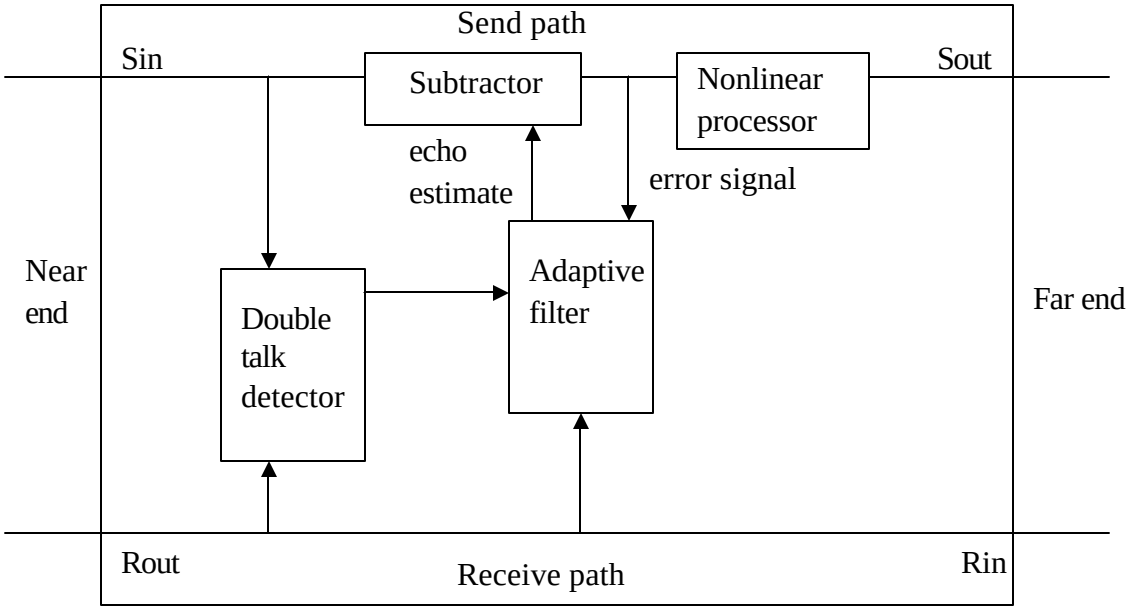


Figure 2-5 Block diagram of a line echo canceller

We now introduce two parameters that are commonly defined and used for echo cancellers as described by Figure 2-5. These parameters will also be used as inputs of our proposed algorithm. They are

- Echo return loss (ERL):

The amount of echo attenuation provided by the hybrid (Figure 2-4). That is, the attenuation of the signal from the Rout port to the Sin port of the echo canceller. The ERL is measured in dB.

- Combined loss (ACOM):

It is defined by the sum (in dB) of the ERL, the attenuation provided by the adaptive filter (cancellation loss) and the attenuation provided by the nonlinear processor (nonlinear processing loss) Figure 2-5.

We now give a high level description on how the blocks shown in Figure 2-5 interact to realize the echo cancellation.

The first step in echo cancellation occurs when the signal coming from the Rin port is sampled and given to the adaptive filter. The signal then travels from the Rout port of the echo canceller, to the hybrid, where most of the signal is transferred to the 2-wire loop connected to the near-end telephone.

A portion of the signal is leaked by the hybrid to the Sin port of the echo canceller. This is the echo that needs to be cancelled by the adaptive filter.

The echo path (Figure 2-4) is highly variable, so the filter that is required to realize the echo cancellation can not be a fixed filter. In fact, the echo path must be estimated for the particular local loop to which the hybrid gets connected. One option to derive the filter is to measure the impulse response of the echo path and then approximate it by a tapped delay line. However, in general the echo path is not stationary. Therefore, such measurements would have to be made repeatedly during a conversation. To eliminate the need of such measurements the filter is made adaptive. An algorithm is implemented which uses the residual error to adapt the filter to the characteristics of the local loop (Figure 2-5). The adaptive filter computes an estimate of the echo. The resulting estimation of the echo is then subtracted from the signal coming from the Sin port, which is composed by the echo and possibly some near end speech and noise.

The resulting output is residual echo that is passed on to the nonlinear processor and is also fed back to the adaptive filter as the error signal. However, this error signal is truly an error signal only when there is no near end speech. If there is near end speech, the "error signal" does not accurately indicates the degree of success of the cancellation and the adaptation algorithm will not converge, resulting in a failed attempt to cancel the echo. For this reason, there is a need to have double talk detection, so that the adaptation would only occur when there is no double talk (both callers speaking simultaneously).

When the echo canceller's double talk detector senses that both the near end and far end callers are speaking at the same time, it informs the adaptive filter so that the filter can ignore the error signal that comes from the subtractor, freezing the filter adaptation. As we said before, near end speech can distort the error signal and confuse the adaptation process, for this reason adaptation is halted when double talk is detected. Of course, the echo canceller still continues to cancel echo during double-talk. As soon as the double talk detector senses that double talk is no longer present, it informs the adaptive filter so that it can, once again, use the error signal to adapt to the impulse response of the hybrid.

The quantization noise introduced by the PCM representation of speech samples and nonlinear echoes make it difficult for the adaptive filter to develop an absolutely perfect echo estimate. Nonlinear echoes can be caused by clipped speech signals, speech compression or poor quality speakerphones. It is extremely difficult to develop an accurate echo estimate of these nonlinear echoes because the echo canceller's linear impulse response model cannot be correlated with these nonlinear

echoes. Consequently, residual echo from the subtractor is reduced to an inaudible level by some nonlinear processing. The nonlinear processor has a suppression threshold that is typically adaptive, based on the Rin and Sin signal levels. The threshold is made adaptive because, if the nonlinear processor simply blocked all signals in the send path, there would be noticeable clipping of speech. For a more detailed description about a nonlinear processor see [R 35].

3 Fuzzy Logic

3.1 Introduction to Fuzzy Logic

"A fuzzy design is an attempt to systematize the natural variations in human perception of truth and to imitate rudimentary skills of approximation" [R 29]. In other words, a fuzzy model of a system is a set of fuzzy rules (Section 3.4.4) by which the behavior of the system is approximately emulated. We will discuss with more details the principles of fuzzy logic and how we can use it in the following sections.

The next table was extracted from reference [R 29] and it shows some of the existing applications that use fuzzy logic.

Application	Product
Automatic train operation (Sendai subway system, Japan)	Industrial
Nuclear reactor control (Art Fugen, Japan)	Industrial
Home heating system (Viessmann-INFORM, Germany)	Commercial
Fingerprint classification (NIST, USA)	Research
Camera tracking (NASA, USA)	Industrial
Target tracker in Patriot missile (MMES, USA)	Industrial
Autofocus still camera (Sanyo, Japan)	Commercial
Fire detector (Cerberus, Switzerland)	Industrial

Table 3-1 Some successful fuzzy systems

3.2 Advantages of Using Fuzzy Logic

Representing a solution with fuzzy sets generally reduces the computational requirements of the system. Approximating a group of related data points by a few fuzzy categories serves this purpose. In some cases, fuzzy methodology makes a solution possible that would otherwise be unthinkable due to cost of computing every single crisp data point.

By selecting the number of fuzzy representative sets, there is a way of adjusting the precision level of a solution. If more fuzzy sets are used in design, systems will require more memory and faster CPUs. At the limit, the number of fuzzy sets becomes equal to the number of crisp data points. That represents the most precise and costly solution.

Two important characteristics of successful fuzzy systems are:

1. The fuzzy systems are simple in terms of their objective and structure - which we call a fuzzy inference system (Section 3.4.8).

2. The fuzzy systems employ solutions articulated in daily language by means of IF-THEN fuzzy rules (Section 3.4.4).

A successful fuzzy system is robust, has adjustable precision and when compared with traditional systems of computation they are more practical and cost effective.

We have seen that the measure of how much the echo is contributing to decrease the quality of a call is a subjective measure. We will see in Chapter 4 that our proposed algorithm to evaluate the echo component of the voice quality in a VoIP system (Figure 4-2) is based on several parameters that are not precise values, but estimated values. Finally, as described in Section 3.4.2, fuzzy logic is useful to model approximate reasoning and is tolerant to imprecise data. In light of the two observations in the beginning of this paragraph, it seems that fuzzy logic would be an adequate tool for the implementation of our proposed algorithm.

3.3 Fuzzy Logic and Embedded Systems

An algorithm to evaluate the voice quality in a VoIP system or network should be a real-time algorithm in order to give operators the precise current voice quality in their network and a chance to react as fast as possible when the quality drops. There are two distinct approaches for were such algorithm should run:

1. It can run inside the embedded system that processes the call.
2. It can run in a network server.

In approach one, there is a disadvantage that embedded systems normally have limited processing power, which is used for high priority tasks like call control, speech compression and echo cancellation. It is common to have a situation that such embedded systems are working very close to their processing capacity. An advantage of this approach is that each embedded system can take action based on the real-time results of the algorithm and try to improve its performance without having to rely in decisions based by a remote server that may even be offline for some reason.

On the other hand, approach two seems to relieve the embedded system of such high processing requirement, once the algorithm would be running in a server somewhere in the network. This approach has a tremendous disadvantage of requiring all the embedded systems in the network to send information about each of their calls to this centralized server (which can be one or more servers). In this case, the bandwidth of the network is compromised. It also has the disadvantage of removing from the network device the ability of monitoring its own voice quality, generate alarms or even try some self-fixing action. It should also be noticed that even in this approach there is some extra processing required from the embedded systems, once they will have to code the required information and access the network in order to send it to the server.

Of course we can mix the ideas of these two approaches and try to find a compromise between network usage and required processing power from the embedded systems.

In this work, we focus on the first approach and we use the ideas of fuzzy logic to develop an algorithm that requires low processing power from the embedded system that carries the VoIP application.

3.4 Fundamentals of Fuzzy Logic

This section briefly describes some fundamental ideas of fuzzy logic that will be used in the development of our proposed algorithm. The description here of the fuzzy logic tools is far from complete. In general, every step of the fuzzy algorithm (also called fuzzy inference system) has several variants. For each step of the fuzzy algorithm, such as fuzzy rules (Section 3.4.4), fuzzy operations (Section 3.4.3), fuzzy implication (Section 3.4.5) and defuzzification (Section 3.4.7), we will describe only the variant that will be used in our proposed algorithm (Chapter 4). For instance, there are several proposed methods of defuzzification, but in Section 3.4.7 we will describe only the method used in our proposed algorithm.

Before going into the details of each step of a generic fuzzy inference system, we will give an overview of such a system. Generally, in introductory books on fuzzy logic such description is given only after all steps have been described. We think there might be some gain in having a first look of the whole system before the parts are explained.

The next figure, extracted from [R 29], depicts a generic fuzzy inference system. The figure will later be analyzed and explained in more details in Section 3.4.8.

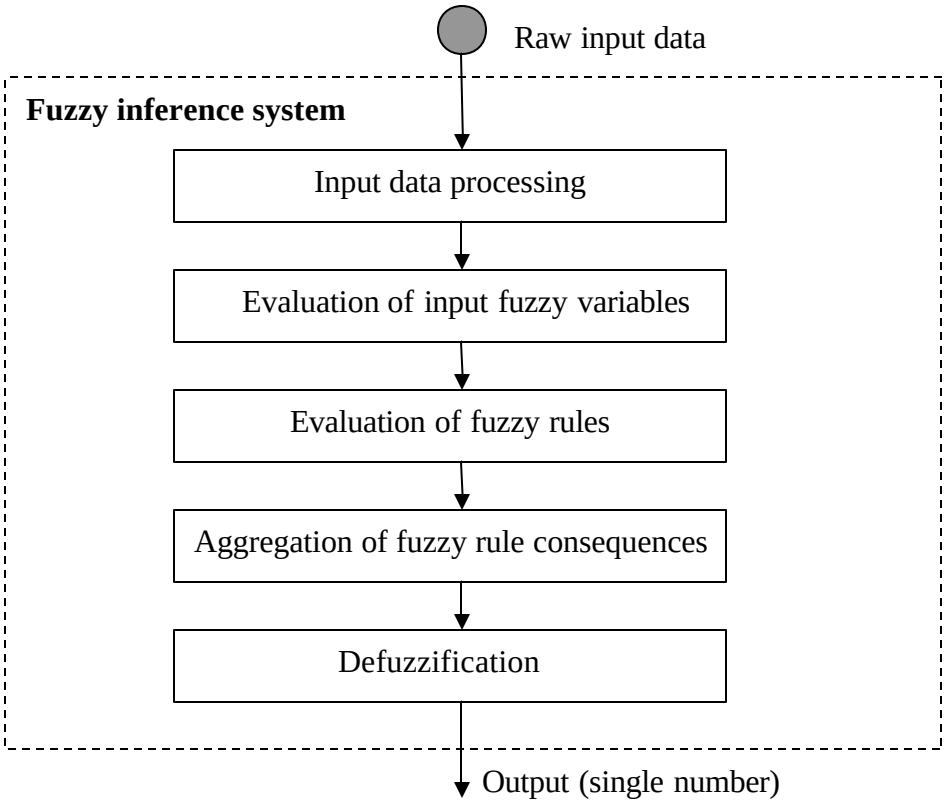


Figure 3-1 Block diagram of a generic fuzzy inference system

In the next sections we give a brief introduction to fuzzy logic and describe with some detail the blocks present in Figure 3-1.

3.4.1 Fuzzy Logic Versus Boolean Logic

Some books describe fuzzy logic as an extension of Boolean logic. In this section we briefly draw some comparisons between Boolean logic and fuzzy logic.

Boolean logic consists of three elements: truth values, linguistic connectors and reasoning types. In Boolean logic, truth values are either 1 or 0. In fuzzy logic, truth is a matter of degree and truth values can range between 1 and 0 in a continuous manner. In fact, this idea of continuum variation of truth values constitutes the most outstanding difference between Boolean logic and fuzzy logic. This will be discussed in more details in Section 3.4.2.

The other two elements that compound the theory of Boolean logic - linguistic connectors (union, intersection, negation) and modes of reasoning (such as syllogism)

- function in the same way for fuzzy logic and Boolean logic. However, their properties and interpretation are affected and they are analyzed with more details in Sections 3.4.3 and 3.4.4 respectively.

The first important definition in fuzzy logic is what is called a fuzzy set and it describes exactly this idea of degree that makes fuzzy logic different from Boolean logic.

3.4.2 Fuzzy Sets and Membership Functions

In classical set theory, an element either belongs to a set or not. The characteristic function of a subset A of a set X is the indicator function

$$\chi_A(x) : X \rightarrow \{0, 1\},$$

with domain X which has value 1 at points of A and 0 at points of X - A, that is

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases},$$

Using the characteristic function defined above, we can also express the set A as

$$A = \{ x \mid \chi_A(x) = 1\} = \{ x \mid x \in A \}$$

As our proposed algorithm only takes real values as inputs, we will from now on consider only the case where the universe $X = \Re$, the set of real numbers.

Motivated by the ideas of classical set theory described above, we can define a fuzzy set as collection of elements with a varying degree of inclusion.

In classical set theory the characteristic function defines a crisp boundary between the elements that belong to a set A and the elements that do not belong to A. In fuzzy theory a function that plays a similar role to the characteristic function is called membership function. A membership function can take values in the interval [0, 1]. For a fuzzy set A, the membership function is defined as

$$\mu_A(x): \Re \rightarrow [0, 1]$$

That is, a membership function is a curve that defines how each point in $\Re$ is mapped to a membership value (or degree of membership) between 0 and 1. If $\mu_A(x_1) > \mu_A(x_2)$, then "x_1 belongs more to A than x_2".

In a similar way as described before for the characteristic function, we can express the fuzzy set A as

$$A = \{ (x, \mu_A(x)) \mid x \in \Re \},$$

where $(x, \mu_A(x))$ is a singleton.

As an example, suppose that all possible temperatures cover the real line $\Re$ (imagine that there is no limit for negative temperatures). The following figure depicts a possible membership function $\mu_A(x)$ for the fuzzy set A = "good temperature to have the ECE annual barbecue".

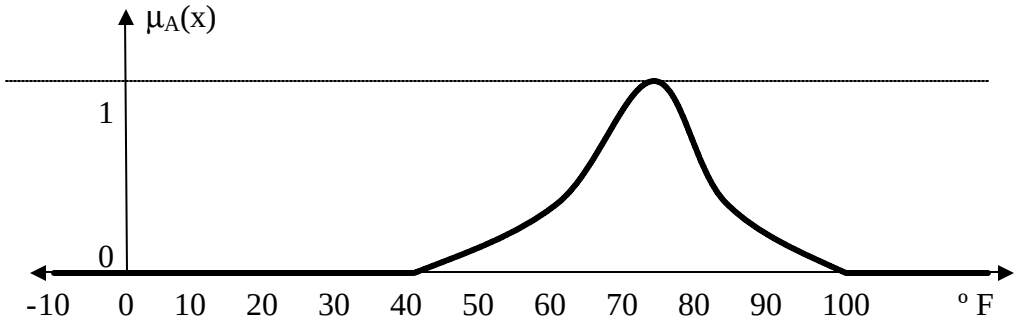


Figure 3-2 Example of a fuzzy member function

For instance, we can say that any temperature between 70º F and 80º F is very good for a barbecue. We can also say that if we have a temperature between 50º F and 60º F it won't be so good, some people will complaint that it's cold. That is why temperatures between 70º F and 80º F have a high value for $\mu_A(x)$ and temperatures between 50º F and 60º F have lower values for $\mu_A(x)$. However, it doesn't make too much sense to say that 60º F is a good temperature to have a barbecue, but 59º F is not. So it makes more sense to define the set A = "good temperature to have the ECE annual barbecue" as a fuzzy set instead of a classical, crisp set.

The difference between crisp (or classical) sets and fuzzy sets is exemplified by the difference between a characteristic function and a membership function as described in the next figure. The membership function $\mu_A(x)$ describes the membership values for the fuzzy set A = "good temperature to have the ECE annual barbecue". The characteristic function $\chi_B(x)$ describes the set B = "temperature is equal or larger than 30º F and equal or smaller than 90º F".

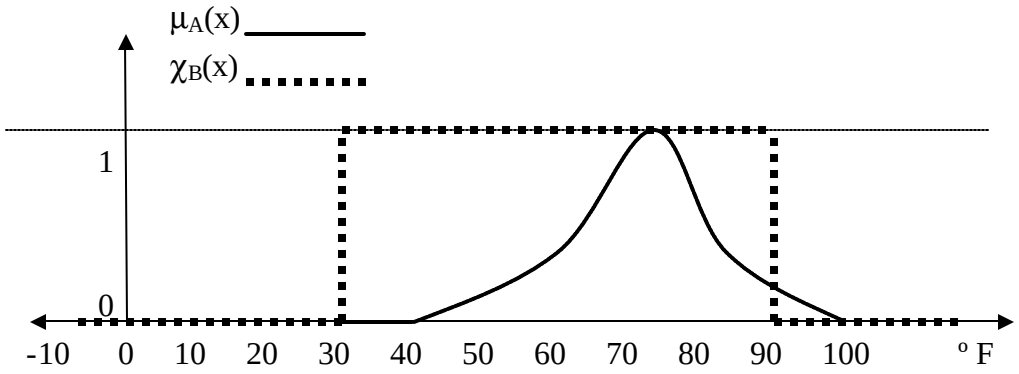


Figure 3-3 Comparing a characteristic function to a membership function

It is clear that the fuzzy set A is imprecise, subjective and it is strongly based in human interpretation. Different temperatures $x \in \Re$ belong to A to certain extent. On the other hand, the set B is completely determined and precise, with a sharp boundary. Some temperatures are in B and all remaining temperatures are not.

The geometrical shape of the membership function should reflect the uncertainty in the corresponding fuzzy variable. For this reason we should not design such membership functions with a high level of detail. In our proposed algorithm we used only membership functions that are piece-wise-linear.

In our work, we explore these ideas of fuzzy sets as a model of approximate reasoning for our proposed algorithm.

3.4.3 Fuzzy Set Operations

Compared to classical set theory, fuzzy set theory offers a family of set operations due to the nature of fuzzy sets. For example, the intersection between two crisp sets such as "ECE students with GPA greater than 3.0" and "ECE students that live in college park" will be a deterministic set. On the other hand, the intersection between two fuzzy sets such as "ECE students with good GPA" and "ECE students who live close to the campus" will be a matter of degree that should be determined from the individual membership functions of the fuzzy sets.

In this section we define the intersection and union of fuzzy sets. These operations (together with fuzzy implication - Section 3.4.5) will be used to compose fuzzy rules, which will be the main logic behind our proposed algorithm.

a) Union of fuzzy sets

There are many possible ways to define the union operator in fuzzy logic and each definition potentially produces a different outcome.

In our work we used the following definition for the union of fuzzy sets:

Suppose we have N fuzzy sets that are represented by the membership functions

$$\mu_1(x) , \mu_2(x) , \dots , \mu_N(x) \quad x \in \Re$$

then, the union of these fuzzy sets is a new fuzzy set with a membership function $\mu_U(x)$ given by

$$\mu_U(x) = \max (\mu_1(x) , \mu_2(x) , \dots , \mu_N(x)) \text{ for all } x \in \Re$$

b) Intersection of fuzzy sets

Again, there are many possible ways to define the intersection operator in fuzzy logic and each definition potentially produces a different outcome.

In our work we used the following definition for the intersection of fuzzy sets:

Suppose we have N fuzzy sets that are represented by the membership functions

$$\mu_1(x) , \mu_2(x) , \dots , \mu_N(x) \quad x \in \Re$$

then, the intersection of these fuzzy sets is a new fuzzy set with a membership function $\mu_I(x)$ given by

$$\mu_I(x) = \min (\mu_1(x) , \mu_2(x) , \dots , \mu_N(x)) \text{ for all } x \in \Re$$

The fuzzy operations described above are the main blocks that build what is called composite fuzzy rules. A fuzzy system has its behavior dictated by the fuzzy rules, which are described in the next section.

3.4.4 Fuzzy Rules

The basic principles of inference in fuzzy logic are adaptations of the classical inference principles to the fuzzy domain.

Fuzzy reasoning is based on inference rules of the form

IF <premise>, THEN <consequence>

similarly to classical logic, but now we use fuzzy sets instead of classical sets. As fuzzy sets define linguistic variables, fuzzy inference rules can model a system linguistically. In fact, our proposed algorithm does just that, modeling linguistically

for instance, what is a good and a bad echo signal in terms of the overall voice quality as we will see in Chapter 4.

An example of a simple fuzzy rule is:

IF x is A, THEN y is B

where A and B are fuzzy sets.

Fuzzy algorithms are just a set of fuzzy rules (which are also called IF/THEN rules). In Chapter 4, the main logic behind our proposed algorithm is described in terms of fuzzy rules and as we will see, those fuzzy rules allow degrees of association between the inputs of the algorithm to better reflect what would be, for instance, a good and bad echo signal with respect to the voice quality of the call.

Interpreting an IF/THEN rule involves two distinct parts:

- Evaluate the premise, that is, use membership functions to map the input values into fuzzy sets
- Apply that result to the consequence - also known as implication.

In the next section we discuss how to compute this implication. However, before proceeding to the next section we will show in a block diagram how a set of fuzzy rules are used by the fuzzy algorithm to produce an output (extracted from [R 29]). This will help to clarify the discussions in the next sections (fuzzy implication, aggregation and defuzzification).

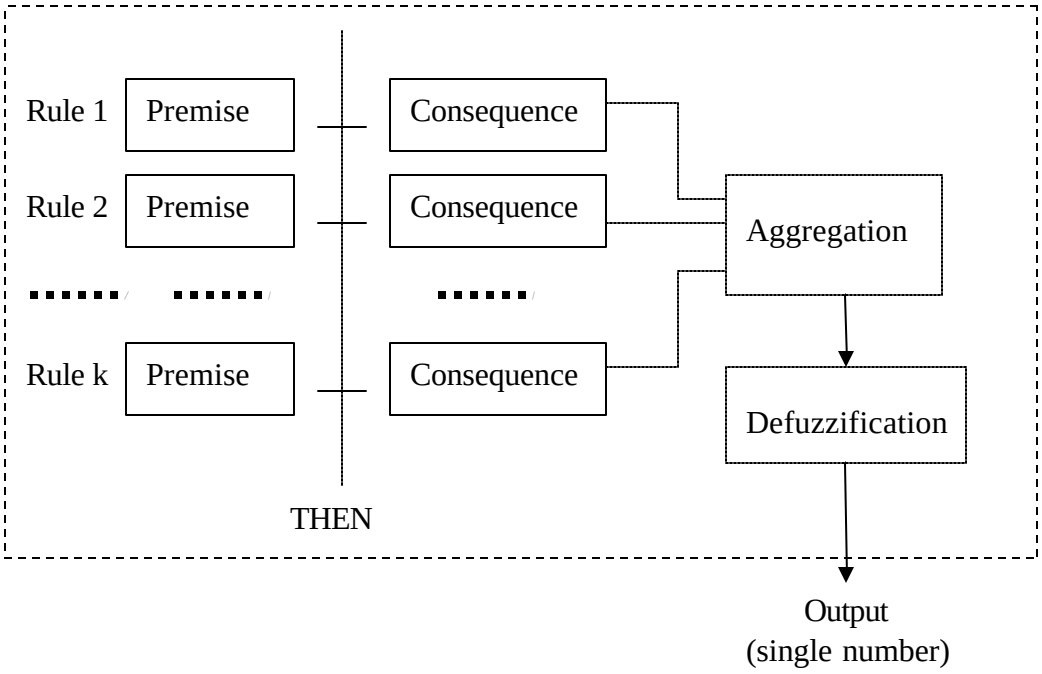


Figure 3-4 Set of rules being aggregated and the final defuzzification

If we now compare Figure 3-4 with the figure that shows a block diagram of a fuzzy inference system (Figure 3-1 in the beginning of Section 3.4) we see that Figure 3-4 shows with some more details what is described by the last (bottom) three blocks of Figure 3-1. Later, in Section 3.4.8, we will come back to these figures to put all the blocks together.

3.4.5 Fuzzy Implication

The fuzzy implication is a mechanism that performs the inference of a fuzzy rule. In the conditional proposition (fuzzy rule) described in the previous section we need to define how the consequence is affected by the premise. The idea here is that the consequence specifies a fuzzy set to be assigned to the output. The fuzzy implication then modifies that fuzzy set to the degree specified by the premise.

As shown in Figure 3-4, the implication process is the first step towards computing the output of the fuzzy inference system. That is, after we have computed the consequences of all fuzzy rules, which is done by the implication method, we aggregate them and defuzzify.

There are several ways to specify the fuzzy implication operator. Some of the most used implication operators are known as Lukasiewicz, Zadeh, Larsen, Mamdani, standard and drastic product implication operator. There is an almost exhaustive list of such operators and their mathematical definitions in terms of membership functions in reference [R 29]. Also, this same reference [R 29] analyzes which set of

operators is probably best adjusted to the characteristics of a problem. In our proposed algorithm we first tried the Mamdani implication operator. The simulation results, after the tuning of the membership functions, were quite good. However, we got even better results using the Larsen implication operator, which acts like a scaling operator, that is, the output membership function is scaled by some factor that is determined by the premise. The Larsen implication operator is the one chosen for our proposed algorithm.

In order to give an example of how this operator works, suppose we have a fuzzy rule like

IF x is A, THEN y is B

where the fuzzy sets A and B are respectively described by the fuzzy membership functions $\mu_A(x)$ and $\mu_B(x)$ as shown in the next figure. Then, for the Larsen implication method, if we have the input variable x set to the value 3 then the output fuzzy set of this rule is given by the membership function $\mu_C(x)$ as shown in the next figure.

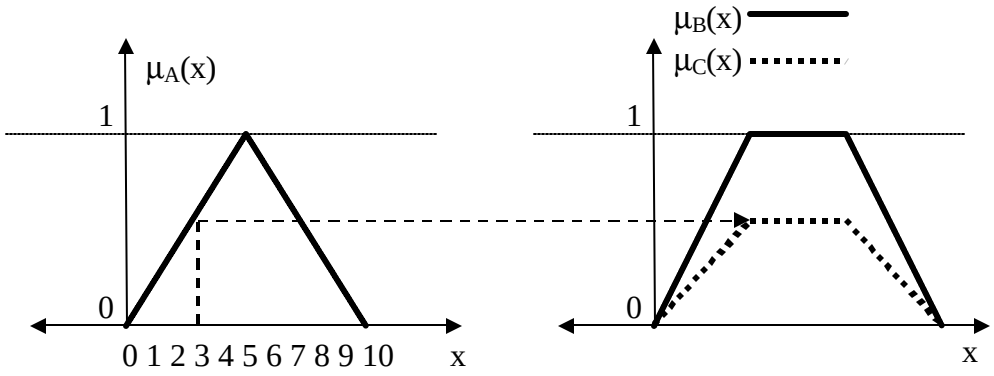


Figure 3-5 Example of the fuzzy implication method chosen for our proposed algorithm

That is, the degree of which the particular value of the input variable x belongs to the fuzzy set A is used to scale the membership function of the output (consequence) of the rule.

3.4.6 Aggregation Operator

For a set of inputs to the fuzzy algorithm, several fuzzy rules may be used to provide the final output of the algorithm. However, in an intermediate step of the algorithm (Figure 3-4) we are required to aggregate the few fuzzy output sets (or membership functions) that are the result of the few fuzzy rules that were used. Aggregating two or more fuzzy output sets yields a new fuzzy set (or membership function) in the fuzzy algorithm.

There are few different ways to specify the fuzzy aggregation operator. In our proposed algorithm we will use the fuzzy union (Section 3.4.3) as the aggregation operator.

In order to exemplify how the aggregation of membership functions work using the fuzzy union operator, suppose that two fuzzy rules generate two membership functions $\mu_A(x)$ and $\mu_B(x)$ as described in the Figure 3-6 below. Then the aggregation of these two outputs using the union operator is given by $\mu_C(x)$ as shown in the figure below.

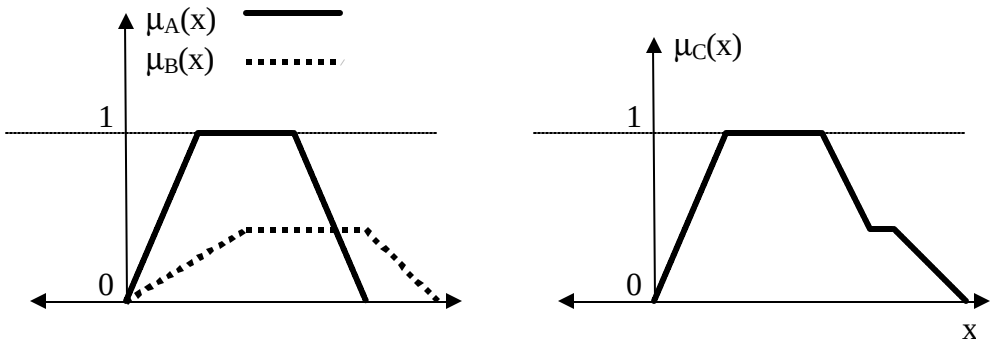


Figure 3-6 Example of the fuzzy aggregation method used in our proposed algorithm

3.4.7 Defuzzification

In the proposed fuzzy system for our algorithm we require a final crisp output. That is, the output of the algorithm is a number between zero and one that informs how good was the echo signal with respect to the voice quality in the call. In order to convert a result from a fuzzy set to a crisp result we use a process that is called defuzzification.

Defuzzification is the process that selects a single value to represent the information contained within a fuzzy set. It is the final treatment to obtain a scalar that is often suitable for the operation of systems in practice (Figure 3-4). There are several methods suggested in the literature for the defuzzification process. Different methods may lead to different results. For our proposed algorithm we used what is called the centroid method (also called center of mass). In fact this is probably the most commonly used defuzzification method.

In the centroid method, the defuzzified output x_c is defined by

$$x_c = \frac{\int_{-\infty}^{\infty} m(x)\,x\,dx}{\int_{-\infty}^{\infty} m(x)\,dx}$$

where $\mu(x)$ is the output membership function after the aggregation of individual IF/THEN rules.

So, for instance, in the example described in Figure 3-6, the output membership function after aggregation is $\mu_C(x)$. The defuzzification in that example is then computed as the center of mass of $\mu_C(x)$ as shown in the figure below.

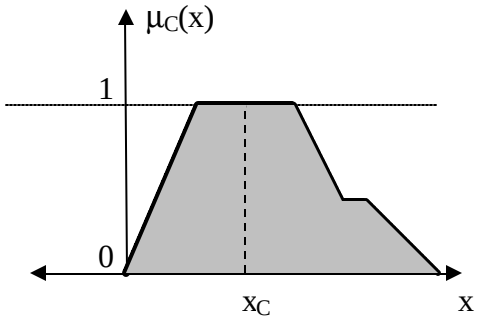


Figure 3-7 Example of a defuzzification by center of mass

3.4.8 Fuzzy Inference System

In this section we put together all the ideas that we developed in the previous sections of this chapter. Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made. A fuzzy inference system has a simple input-output relationship as was shown in Figure 3-1. Input data is collected from the external world. Then it is processed by the fuzzy inference system to produce the output data to be used back in the external world.

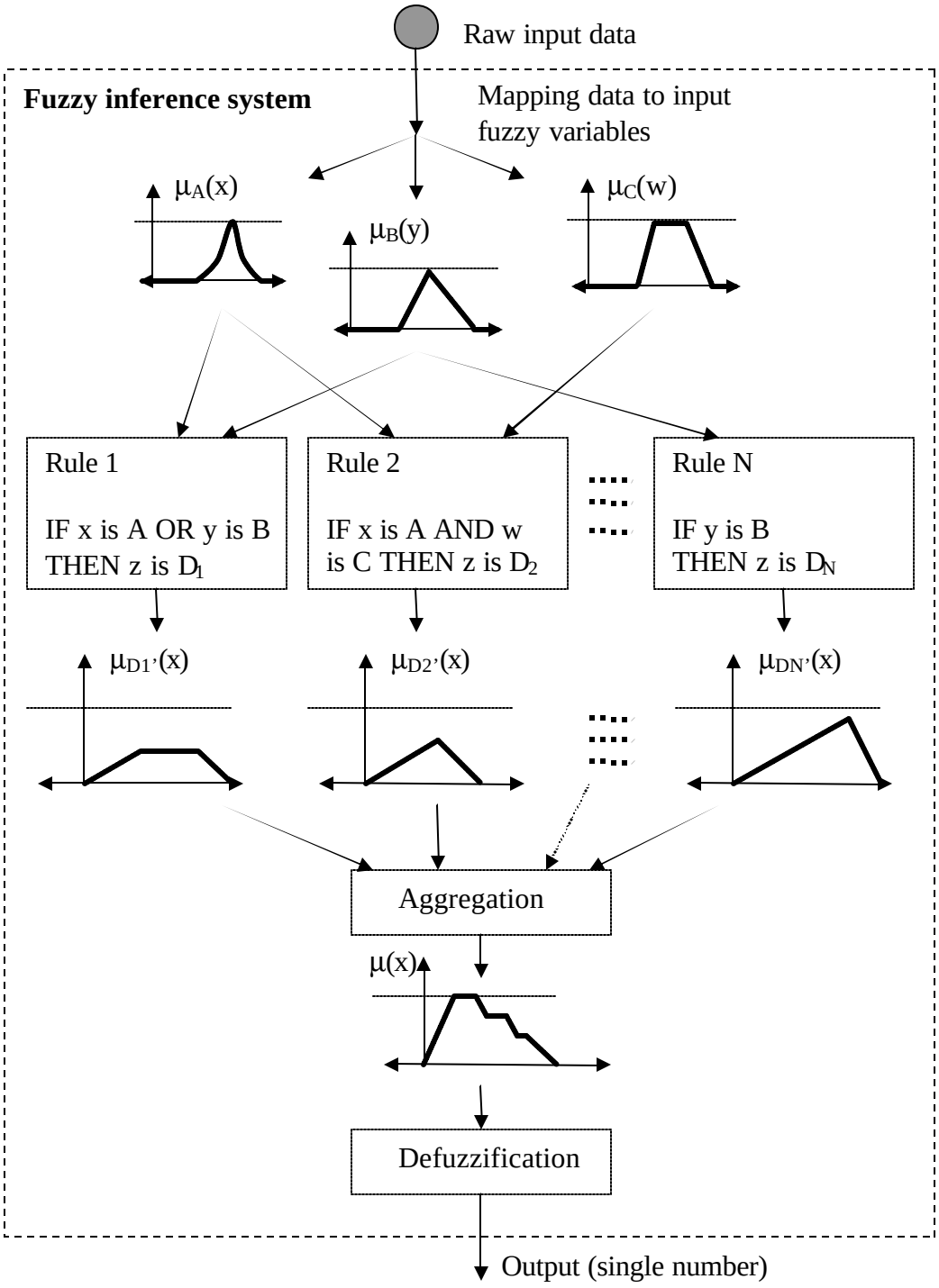


Figure 3-8 Detailed fuzzy inference system

4 A Fuzzy Algorithm to Evaluate the Echo Component of the Voice Quality in a VoIP Network

4.1 Objective Evaluation of the Voice Quality

In Section 1.1 we described the several parameters that can be used to evaluate the performance of a VoIP system or network (Table 1-1). In this work we are focusing on measuring the voice quality in a VoIP system or network and in Section 1.1 we showed three different perspectives for voice quality evaluation in a VoIP network. We decided to adopt the objective evaluation of the voice quality (Section 1.1.3) for our proposed algorithm for two reasons:

- Differently from the subjective quality method (Section 1.1.2), an objective method can be automated once it doesn't require human intervention or feedback. If well designed, such a method can even estimate the quality in real-time.

- Differently from the network quality method (Section 1.1.1), an objective method takes into consideration the user's perspective of the call and not only parameters that qualify the performance of the IP network.

As is described in [R 5], there are three classes of objective voice quality evaluation metrics: network-parameter based metrics, psycho-acoustic metrics, and elementary metrics.

- Parameter-based metrics do not consider the actual voice signal. Instead, these metrics sum impairment factors that characterize the individual components of the communication system. For instance, in the E-model (Section 1.1.3) the packet loss and delay in a VoIP system are translated into impairment factors. Parameter-based metrics such as the E-model hold promise for predicting subjective voice quality but still require extensive refinements and verifications.

- Psycho-acoustic metrics transform voice signals to a reduced representation to retain only perceptually significant aspects. These metrics aim to predict the subjective quality over a wide range of voice signal distortions. One example of such metric is the PESQ algorithm (Section 1.1.3).

- Elementary objective voice quality metrics rely on low-complexity signal processing parameters and techniques to predict subjective voice quality. Elementary metrics generally have smaller correlations with subjective voice quality than highly complex psycho-acoustic metrics and do not provide the perception modeling needed for psycho-acoustic coder algorithm

development. However, elementary metrics represent a good engineering tradeoff for communication and networking system researchers and developers in that they allow for fairly detailed conclusions about voice quality while having low computational complexity.

The algorithm that we propose in this chapter is a fuzzy algorithm that estimates the amount of echo present in a VoIP call after echo cancellation. Based on the descriptions given in Section 1.1 about the different perspectives for evaluating the voice quality in VoIP networks and based on the objective voice quality metrics described above we have the following diagram:

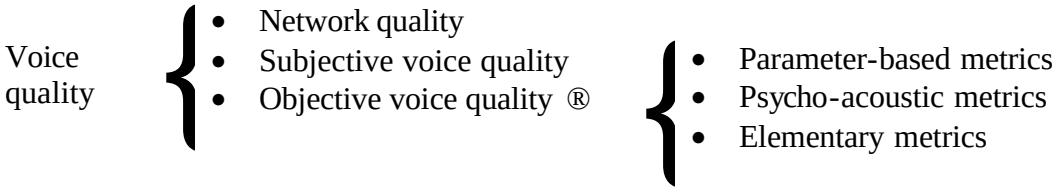


Figure 4-1 Classification of voice quality algorithms for VoIP systems

So, the fuzzy algorithm proposed in this chapter estimates the echo quality factor of the voice quality and it is a building block of an objective, passive, voice quality algorithm based on elementary metrics that can run in real-time and estimate the voice quality for live calls in a VoIP system or network.

As we mentioned in Chapter 1, the critical issues in delivering good voice quality over IP networks are: packet loss, delay, echo and jitter. These issues are all correlated, but there is a stronger correlation between jitter, delay and packet loss. Jitter in VoIP systems is normally compensated for by using a playout buffer at the receiving end, which introduces delay and additional packet loss. So we can imagine a fuzzy inference system that evaluates the voice quality in a VoIP network described by the following blocks:

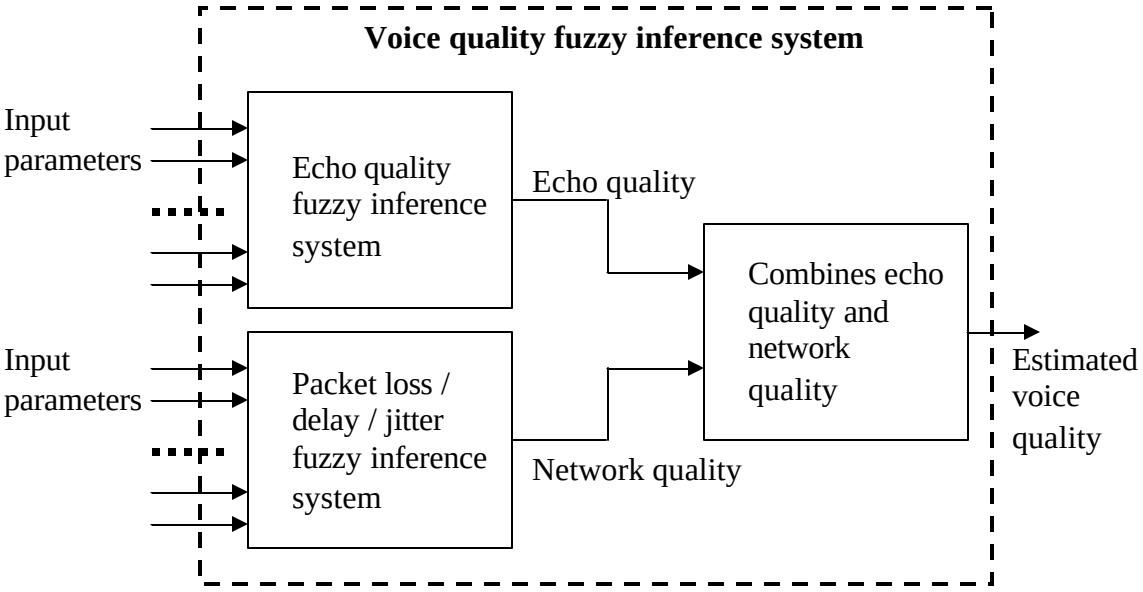


Figure 4-2 Fuzzy inference system to estimate the voice quality in an VoIP network

We note that in the above figure, the block that combines the echo quality and network quality into a final voice quality estimative can be a fuzzy inference system or not.

We can go even further and imagine a fuzzy inference system that estimates the performance of a VoIP network (not only the voice quality - see Table 1-1) for a call. Based on Table 1-1 and Figure 1-4 we can have such fuzzy inference system described as follows

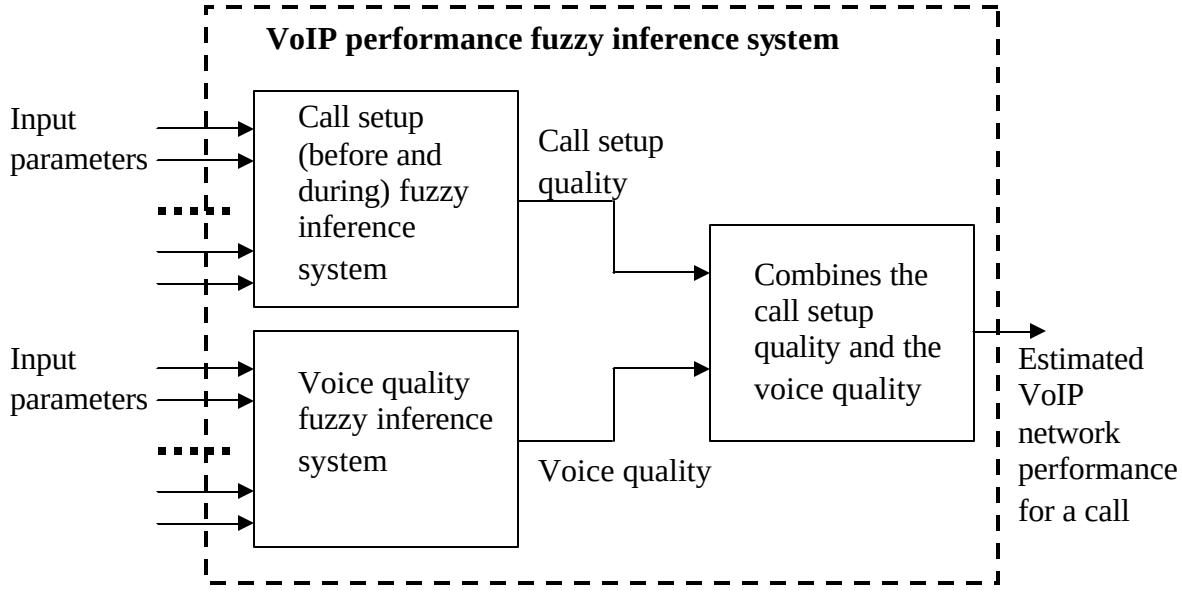


Figure 4-3 Fuzzy inference system that estimates the performance of a VoIP network for a call

That is, the fuzzy inference system that estimates the voice quality in a VoIP network (Figure 4-2) is a building block of the larger system described in Figure 4-3.

However, this seems to be a long shot and in this chapter we have a much simpler objective. In the next section we propose an echo quality fuzzy inference system, which is a component of a larger system (Figure 4-2) that can estimate the voice quality in a VoIP system or network.

4.2 Channel Based Algorithm

A VoIP system or network may carry a huge number of VoIP calls simultaneously. In fact there are high density platforms available in the market that can handle dozens of VoIP calls per card. In this section we propose an algorithm to evaluate the quality of the echo signal for a single VoIP channel. There are many possible ways of aggregating the echo component of several channels in the network to provide an evaluation of the echo in the network as whole and this will be discussed later.

The objectives of the proposed algorithm have already been described in the previous sections and chapters. The following list is a summary of such objectives.

- Obtain the echo quality component of an objective voice quality algorithm based on elementary metrics to estimate the voice quality in a VoIP system (Section 4.1 and Figure 4-2).
- The algorithm must have a computational complexity low enough such that it can run in an embedded module inside every VoIP channel in the VoIP system or network (Section 1.1).
- The algorithm must be able to run for live VoIP calls without the need of a reference signal, which is one of the limitations of several objective voice quality methodologies including the PSQM (Section 1.1.3).
- The algorithm must give a real-time estimation of the echo signal by outputting a few parameters (or scores).
- The algorithm's output scores will not be as reliable and precise as the MOS (Section 1.1.2) or the PSQM / PESQ (Section 1.1.3) scores. As is common for elementary metrics methodologies of evaluating objective voice quality (Section 4.1), there will be a tradeoff between precision and computational complexity that we should be willing to accept.

In order to achieve the objectives listed above we propose the following design characteristics for the algorithm.

- Use a fuzzy inference system to estimate the echo component of the voice quality. This should give the algorithm a low computational complexity, the ability to run in real-time for live calls embedded in every VoIP channel. The usage of fuzzy logic will also result in some imprecision for the final scores.
- In order to obtain a real-time, low computational complexity algorithm we chose to use as inputs to our fuzzy inference system parameters that are already being

computed or estimated by the echo canceller, such as estimates for the ERL and ACOM (Section 2.4), speech powers estimations and noise powers estimations.

We should note that although the fuzzy logic implementation results in low computational complexity it has the disadvantage of not being precise. As we said before, it reflects approximate human reasoning and it will never be as good as the subjective MOS (Section 1.1.2) and it won't be as precise as the PSQM or PESQ.

Based on the ideas described in Chapter 3, we will now design a fuzzy inference system to evaluate the echo component of the voice quality. The first step is to define the input parameters that will be used by the fuzzy inference system. In order to achieve minimum computational power, we used only parameters that are already estimated and used by the echo canceller (Figure 2-5). Of course, different echo cancellers may estimate a different set of parameters and in this case we should need some extra computations to estimate the required parameters for the algorithm proposed here. The input parameters that we will use are:

- Echo return loss (ERL) - Section 2.4
- Combined loss (ACOM) - Section 2.4
- Receive speech power - an estimate of the speech power in the receive path (Figure 2-5)
- Receive noise power - an estimate of the noise in the receive path (Figure 2-5)
- Transmit speech power - an estimate of the speech power in the send path (Figure 2-5) after the echo cancellation
- Transmit noise power - an estimate of the noise in the send path (Figure 2-5)

Then we need to define the fuzzy sets and membership functions for each input so that we can associate the raw input parameters to a fuzzy set (Section 3.4.2).

Based on the input parameters define above, we will use the fuzzy sets described in the next table.

Fuzzy set	Description
Good ERL	Represents values of ERL that will help the echo canceller to realize a good echo cancellation.
Bad receive speech power	The receive speech powers in this set are either to low or to high, making it difficult for the echo canceller to generate the signal that must be subtracted in the send path.
Bad transmit noise power	Represents values of the transmit noise that may disrupt the convergence of the adaptive filter.
Bad ACOM	With high probability, VoIP systems with ACOM values in this set will have echo problems and the voice quality will be bad.

Moderate ACOM	Represents values of ACOM that may indicate that the echo cancellation was not good enough and some echo may be leaked to the far end.
Good ACOM	VoIP systems with ACOM values in this set are able to cancel most of the echo in the calls.

Table 4-1 Fuzzy sets associated to the input parameters

The output of the fuzzy inference system will be an estimate of the echo component of the voice quality. We defined the following three output membership functions:

- The membership function for the fuzzy set "bad echo (be)"

$$\mu_{be}(x) = \begin{cases} 1-2x, 0 \le x \le \frac{1}{2} \\ 0, otherwise \end{cases}$$

- The membership function for the fuzzy set "moderate echo (me)"

$$\mu_{me}(x) = \begin{cases} 2x, 0 \le x \le \frac{1}{2} \\ 2(1-x), \frac{1}{2} < x \le 1 \\ 0, otherwise \end{cases}$$

- The membership function for the fuzzy set "good echo (ge)"

$$\mu_{ge}(x) = \begin{cases} 2x-1, \frac{1}{2} \le x \le 1 \\ 0, otherwise \end{cases}$$

Graphically we have

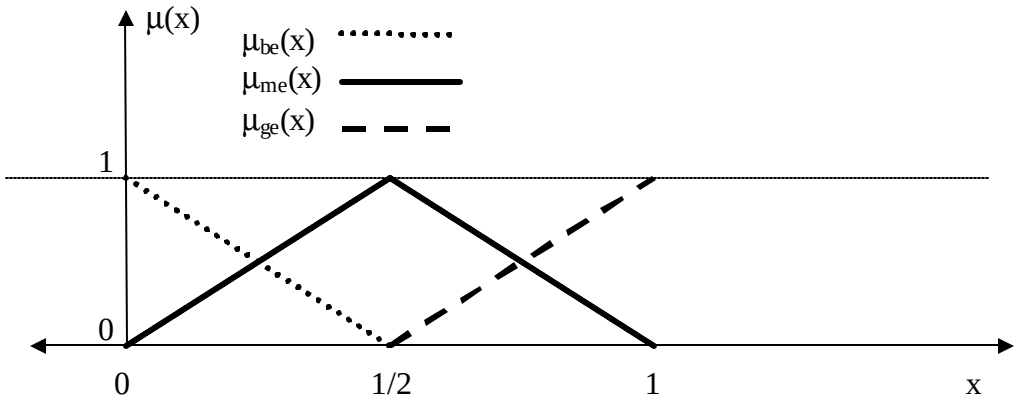


Figure 4-4 Output membership functions for the echo component of the voice quality

Now it is time to define the input membership functions (Section 3.4.2). The approach we used is to define the input membership functions based in empirical reasoning and then we spent some time tuning those membership functions. But the tuning is done after the fuzzy inference system is designed because we need to use the output of the algorithm as a feedback for tuning. For this reason, we will describe the input membership functions later in this section. First we will describe our proposed fuzzy inference system with the complete set of fuzzy rules, operations, and defuzzification.

The fuzzy rules (Section 3.4.4) together with the fuzzy membership functions are the main elements that reflect the empirical reasoning behind the proposed fuzzy inference system. The next table describes the fuzzy rules that we adopted for our proposed algorithm and the empirical reasoning behind each rule.

Fuzzy rule	Empirical reasoning
IF ACOM is bad THEN echo is bad.	The ACOM is a major parameter for estimating the quality of the echo signal (Section 2.4). If the ACOM is bad, most probably the user is perceiving echo.
IF ACOM is good, THEN echo is good.	With a good ACOM, some echo is being cancelled successfully, independently of the other parameters.
IF ACOM is moderate AND ERL is good THEN echo is moderate.	If ACOM is moderate, there is some uncertainty about the quality of the echo signal. So we use the ERL to better estimate it.
IF receive speech power is bad AND transmit noise is bad THEN	The signal levels for transmit and receive speech as well as for transmit noise are all contributing

echo is bad.	to a bad echo signal.

Table 4-2 Fuzzy rules to evaluate the echo component of the voice quality

The fuzzy implication operator that we chose for our proposed algorithm is Larsen operator as described in Section 3.4.5. The defuzzification method (Section 3.4.7) that we chose is the center of mass method.

An advantage of using fuzzy logic is that we can first define the fuzzy input variables and elaborate the fuzzy rules and then we can tune the membership functions by running the algorithm for calls for which we know the MOS. That is exactly what we did in order to define the following membership functions for each one of the fuzzy inference input variables.

We used Matlab and its fuzzy logic toolbox to implement our proposed algorithm and run a set of 32 calls. We implemented it in a way that we would give to Matlab 5 different fuzzy inference systems at a time. The difference between the fuzzy systems was only in terms of the membership functions. All systems had the same input variables, fuzzy operations, and fuzzy rules, but different membership functions for the fuzzy variables. Then, after running all 32 calls for each one of the 5 fuzzy inference systems we could compare the result of the fuzzy algorithm to the expected MOS scores (after the echo cancellation). We chose to use 5 different fuzzy inference systems in each tuning step because there are so many parameters that you can change in a membership function that you easily get lost if you try to change several parameters at once. So, for instance, if we are tuning a specific triangular membership function that has a positive and a negative slope we would first tune the positive slope of the triangle and try it with say 3 to 5 different positive slopes. As we followed this tuning methodology for each parameter of the membership functions, we never found it necessary to use more than 5 different versions of such parameter (the positive slope in this case).

As a result of the tuning process described above we derived the following membership functions for the fuzzy sets described in Table 4-1.

Echo return loss (ERL)

- The membership function for the fuzzy set "good ERL (gerl)"

$$\mu_{gerl}(x) = \begin{cases} \dfrac{x-20}{10}, 20 \le x \le 30 \\ 1, x > 30 \\ 0, otherwise \end{cases}$$

Graphically we have

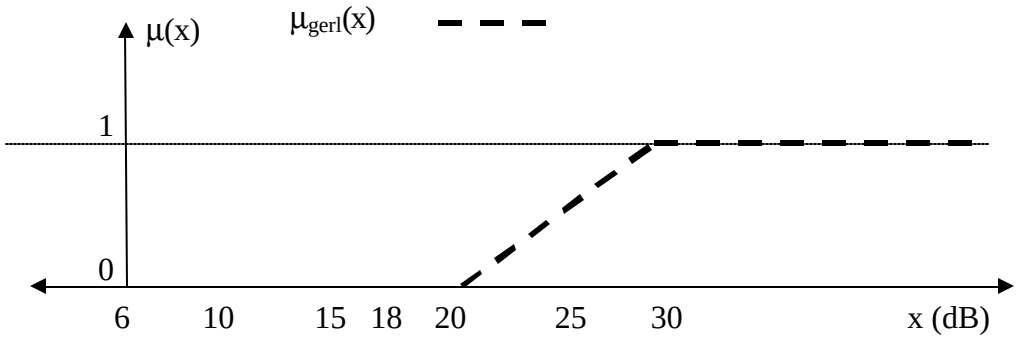


Figure 4-5 ERL fuzzy membership function

Combined Loss (ACOM)

- The membership function for the fuzzy set "bad ACOM (bacom)"

$$\mu_{bacom}(x) = \begin{cases} \frac{23-x}{17}, 6 \le x \le 23 \\ 1, x < 6 \\ 0, otherwise \end{cases}$$

- The membership function for the fuzzy set "moderate ACOM (macom)"

$$\mu_{macom}(x) = \begin{cases} \frac{x-12}{11}, 12 \le x \le 23 \\ \frac{36-x}{13}, 23 < x \le 36 \\ 0, otherwise \end{cases}$$

- The membership function for the fuzzy set "good ACOM (gacom)"

$$\mu_{gacom}(x) = \begin{cases} \frac{x-23}{17}, 23 \le x \le 40 \\ 1, x > 40 \\ 0, otherwise \end{cases}$$

Graphically we have

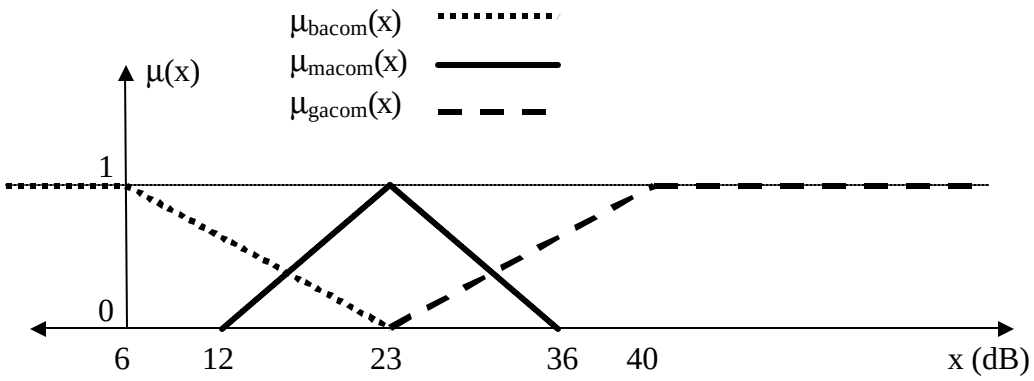


Figure 4-6 ACOM fuzzy membership functions

Receive speech power

- The membership function for the fuzzy set "bad receive speech power (brsp)"

$$\mu_{brsp}(x) = \begin{cases} \frac{-25-x}{5}, -30 \le x \le -25 \\ 0, -25 < x < -15 \\ \frac{x+15}{10}, -15 \le x \le -5 \\ 1, otherwise \end{cases}$$

Graphically we have

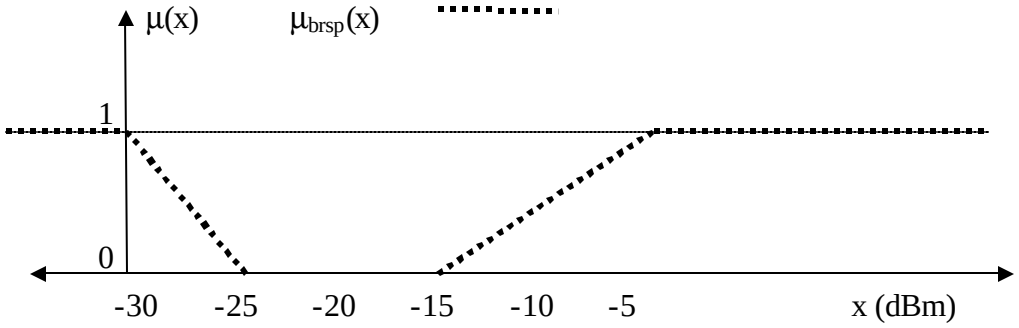


Figure 4-7 Receive speech fuzzy membership function

Transmit noise power

- The membership function for the fuzzy set "bad transmit noise power (btnp)"

$$\mu_{btnp}(x) = \begin{cases} \frac{x+45}{9}, -45 \le x \le -36 \\ 1, x > -36 \\ 0, otherwise \end{cases}$$

Graphically we have

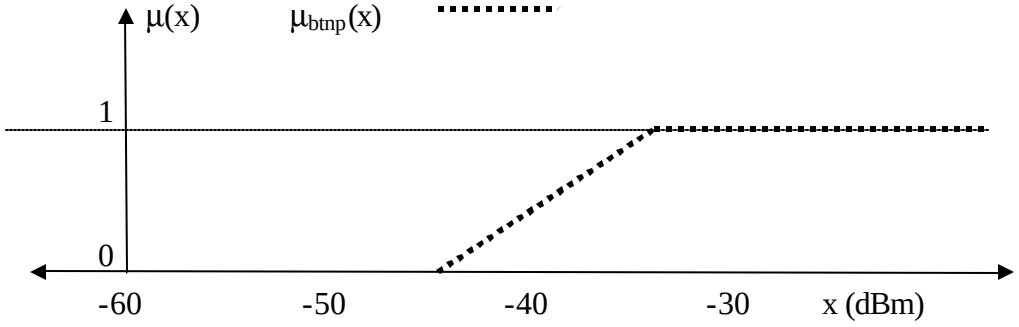


Figure 4-8 Transmit noise fuzzy membership function

Now we show an example of a fuzzy rule computation carried by the fuzzy inference system proposed above. Suppose we have the following set of inputs (raw data)

ERL = 23 dB
ACOM = 28dB
Receive speech power = -27dBm

Transmit noise power = -50dBm

Suppose we want to compute the fuzzy rule

"IF ACOM is moderate AND ERL is good THEN echo is moderate"

as described in Table 4-2.

The fuzzy set "moderate ACOM" is described by the membership function $\mu_{macom}(x)$, the fuzzy set "good ERL" is described by the membership function $\mu_{gerl}(x)$ and finally, the fuzzy set "moderate echo" is described by the membership function $\mu_{me}(x)$. All these membership functions were previously defined in this section.

Now we map the input data into the fuzzy sets:

$$\mu_{macom}(28) = \frac{36-28}{13} = \frac{8}{13}$$

$$\mu_{gerl}(23) = \frac{23-20}{10} = \frac{3}{10}$$

The fuzzy AND operator (intersection of fuzzy sets) that we are using is defined in Section 3.4.3., the fuzzy implication operator that we are using is the Larsen operator, so the output of this rule is given by:

$$\mu(x) = \frac{3}{10}\mu_{me}(x) = \begin{cases} \frac{3}{5}x, 0 \le x \le \frac{1}{2} \\ \frac{3(1-x)}{5}, \frac{1}{2} < x \le 1 \\ 0, otherwise \end{cases}$$

The next figure is a graphical representation of the computation described above - this kind of graphical analysis helps in the tuning of the membership functions and the fuzzy rules.

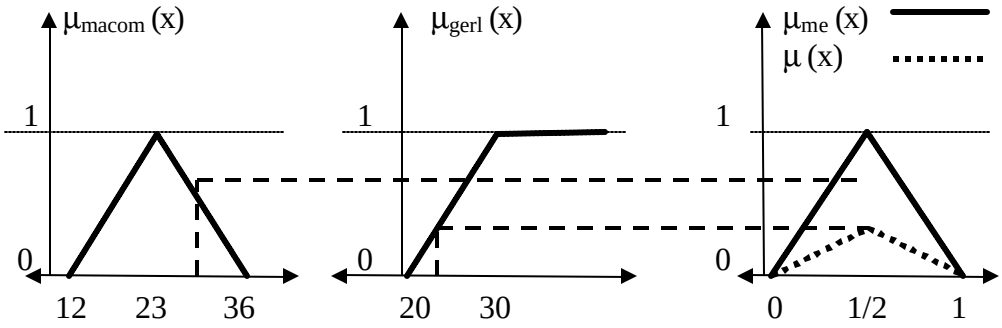


Figure 4-9 Graphical interpretation of the fuzzy rule

Once all the rules described in Table 4-2 are computed as shown above, the output membership functions ('consequences' of the fuzzy rules - which is μ(x) in the example above) are then aggregated (Section 3.4.6) and the output of the fuzzy inference system is computed using the center of mass defuzzification method (Section 3.4.7) - as described in Figure 3-8.

The idea is to run such computation periodically throughout a VoIP call, getting instantaneous values of the quality of the echo signal in such calls. This was done in our simulations and the next figure shows the output of the fuzzy algorithm for a call with a relatively bad echo signal. More simulation results are described in Section 4.4.

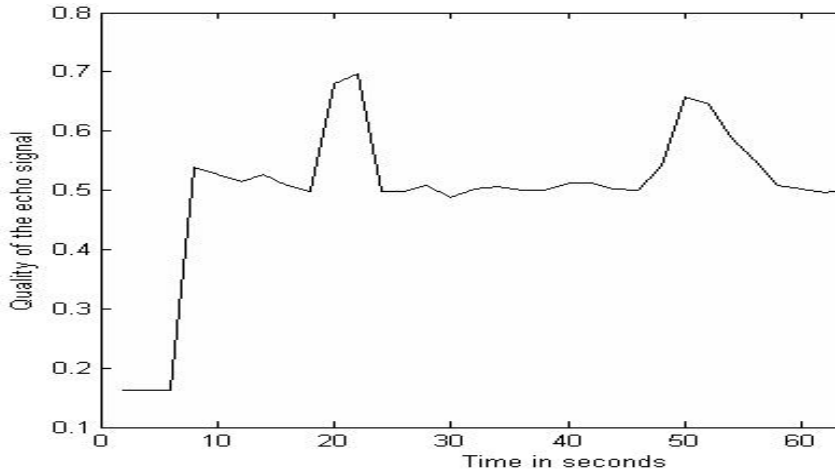


Figure 4-10 Echo quality estimation for a simulated call

This was a call with duration of about one minute. We computed the quality of the echo signal every two seconds. The higher the number for the estimative of echo quality, the better it is as described by the membership functions on Figure 4-4.

Figure 4-11 and Figure 4-12 below show the signals used to estimate the quality of the echo signal. Although we really need to listen to this signal in order to evaluate the voice quality, we can clearly see the echo component embedded in the signal.

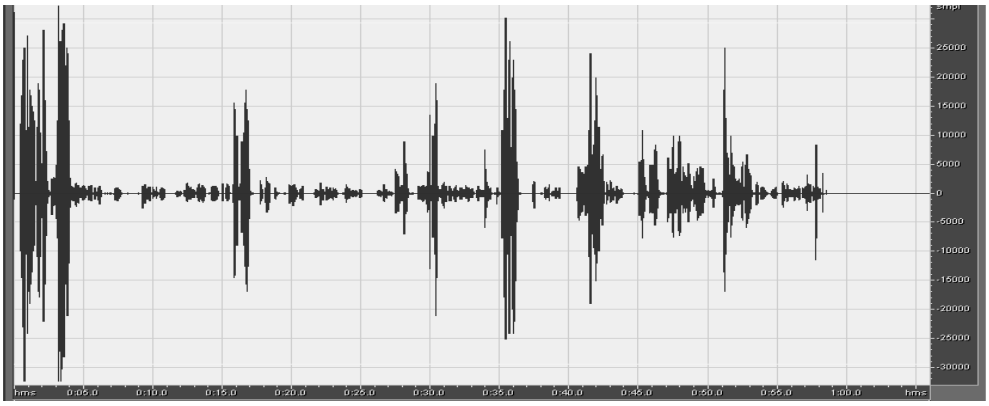


Figure 4-11 Speech and echo signals in the send path

Below we have the speech signal in the receive path. A portion of this signal will be reflected into the send path to compose the echo signal (Figure 2-2).

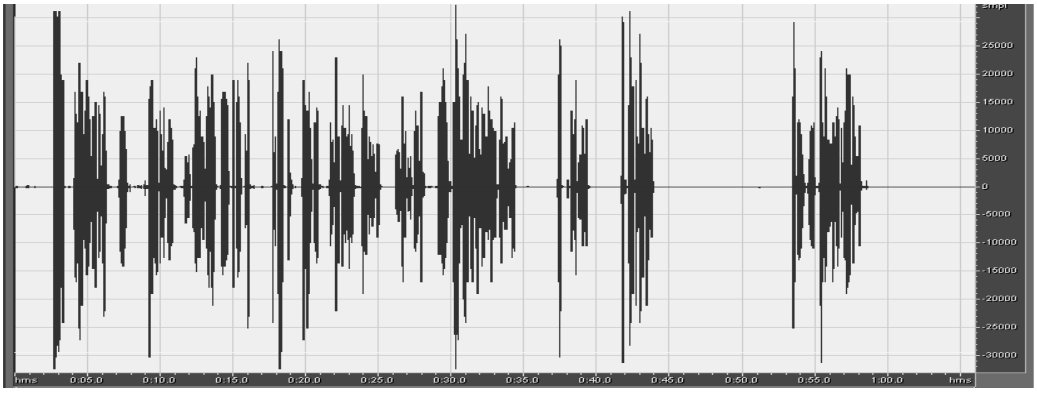


Figure 4-12 Speech signal in the receive path

4.3 Network Based Algorithm

So far we have analyzed the echo signal present in a single channel in a VoIP network. But it is interesting to have voice quality performance or, in the case of this chapter, echo quality for a group of channels in the network. In a practical scenario, we can imagine a group of VoIP channels that are in the same subnetwork, so it makes sense for the network provider to have an aggregate estimate of that subnetwork echo quality besides having the echo quality for each individual channel.

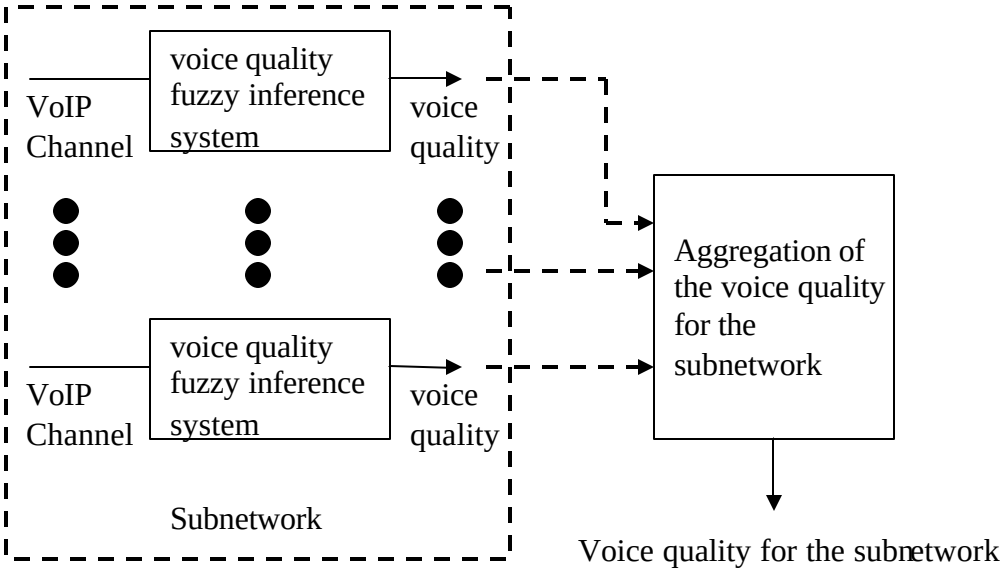


Figure 4-13 Estimating the voice quality for a subnetwork of VoIP channels

In a similar way, we can also have a hierarchy of subnetworks that ends with the provider's complete IP network and we can evaluate the echo quality for each level of these subnetworks hierarchies.

4.4 Simulation Results

We ran the proposed channel based algorithm for 32 calls with different levels of voice quality. These calls were all previously generated in the lab and recorded in PCM format. We split the calls into two groups. The first group contains 16 calls where echo was effectively cancelled or no echo was created in that call. The second group contains calls with various levels of echo signals and background noise. Then, we ran these calls through a Texas Instruments DSP platform with a line echo canceller. We then collected the measurements from this echo canceller, parsed these measurements and finally used them as inputs to our fuzzy algorithm. As described in the previous section, the collected measurements from the echo canceller were the ERL, ACOM, transmit speech power, receive speech power, transmit noise power and receive noise power.

The fuzzy algorithm was implemented in Matlab, using the fuzzy inference toolbox provided by Matlab. The setup parameters that are necessary to implement our proposed algorithm using this toolbox are listed in the appendix. For each simulated VoIP call we had the near end signal, the far end signal and the measurements provided by the echo canceller. As was described, the proposed algorithm gives an instantaneous estimation of the echo signal for the call. During our simulations we chose to compute this estimation every two seconds.

There are several ways to report the results of the algorithm for each call. One possible way is to report every instantaneous output of the fuzzy algorithm and in this case we can have a picture of the evolution of the echo component of the voice quality throughout the call as described in Figure 4-10. However, from a network provider perspective this would require a lot of bandwidth, data storage and possibly unnecessary processing. For this reason, we also developed some alternative ways to try to consolidate all these temporal measurements (Figure 4-10) into a small set of measurements for the whole call.

One approach that we considered was to provide an average of the outputs of the fuzzy inference system throughout the call. We obtained even better results if before computing the average we discard the 5% highest and lowest outputs of the fuzzy inference system. Using this idea, we can provide a single number that reflects the overall echo signal quality for the observed VoIP call.

The next figures show the simulation results for the approach described above. We ran all the 32 calls and computed an estimate of the echo component of the voice quality for each call by computing the average output of the fuzzy inference system. The following figure depicts the result of the simulation for the group of 16 calls with good echo signal. Each bar represents the average quality for one VoIP call.

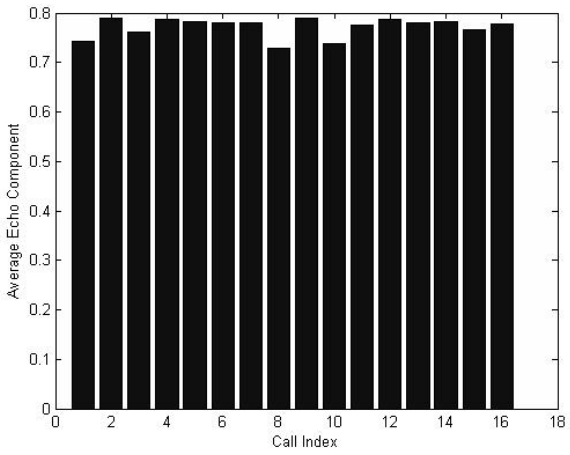


Figure 4-14 Average quality for calls with good echo component

Analyzing the figure above, we note that the algorithm was really effective in estimating the echo component for these calls. There is a little variation (less than 15%) between the scores for this group of 16 calls and in fact they have very similar MOS scores. We should note that due to the design of the output functions of the fuzzy rules, the defuzzification method and the fuzzy implication method chosen for the algorithm, the maximum value for the quality of the echo signal is 5/6, or 0.8333. So, the figure above really shows calls with good echo signal.

The second group contains calls where echo was not effectively cancelled or there was some high noise or high difference level between the receive path signal and the send path signal. The results for these calls are shown below. Again, each bar represents the average quality for a VoIP call. There are a total of 16 calls in this group.

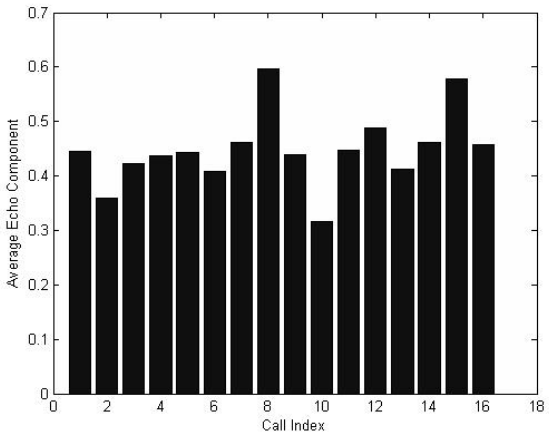


Figure 4-15 Average quality for calls with bad echo component

We see from this figure that the algorithm was able to declare several calls as having bad echo quality. However, some calls (like call number 8 and number 15) obtained a much higher score than they really deserved.

Of course, using the approach described above, we lose the detailed information about the call. For instance, if we get an average score for the voice quality in a given call we do not know if the call had a good quality for approximately half of the time and a bad quality in the remaining time or if the call quality was average during the whole call. So, another possible way of aggregation that is in between the detailed temporal results and the average proposed above is to provide a histogram of the levels of echo and voice quality for the call.

For instance, for the call described by Figure 4-10 we have the following histogram

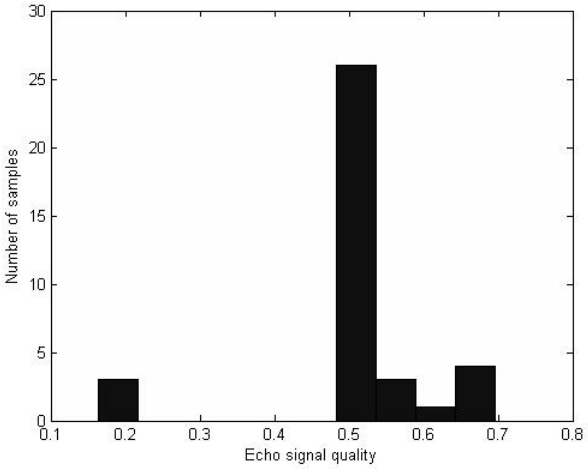


Figure 4-16 Histogram showing the echo signal quality

The next figure shows a histogram for a simulated call that had a good estimated echo quality for most of the call.

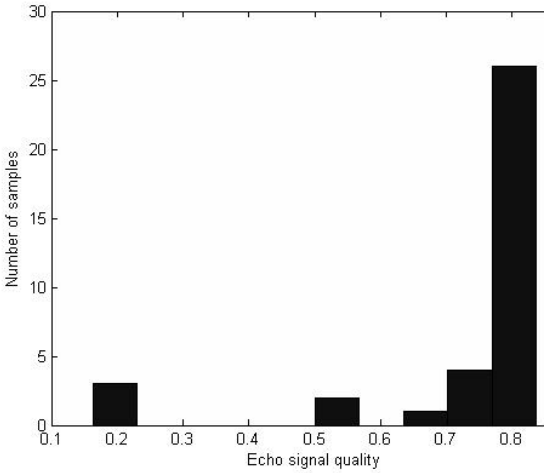


Figure 4-17 Histogram for a call with a good estimated echo signal quality

We also used the complete set of instantaneous outputs of the fuzzy inference system to analyze the influence of the different input data into the output provided by the algorithm. For instance in the example provided in Section 4.2 we could compare the influence of the receive and transmit speech powers (Figure 4-12 and Figure 4-11 respectively) with the output of the proposed algorithm (Figure 4-10).

In the next figure we use another simulated call with undesirable echo signal and unsatisfactory echo cancellation to compare the output of the proposed fuzzy algorithm to the estimated level of combined loss (ACOM) for that call. The call has an approximate duration of one minute.

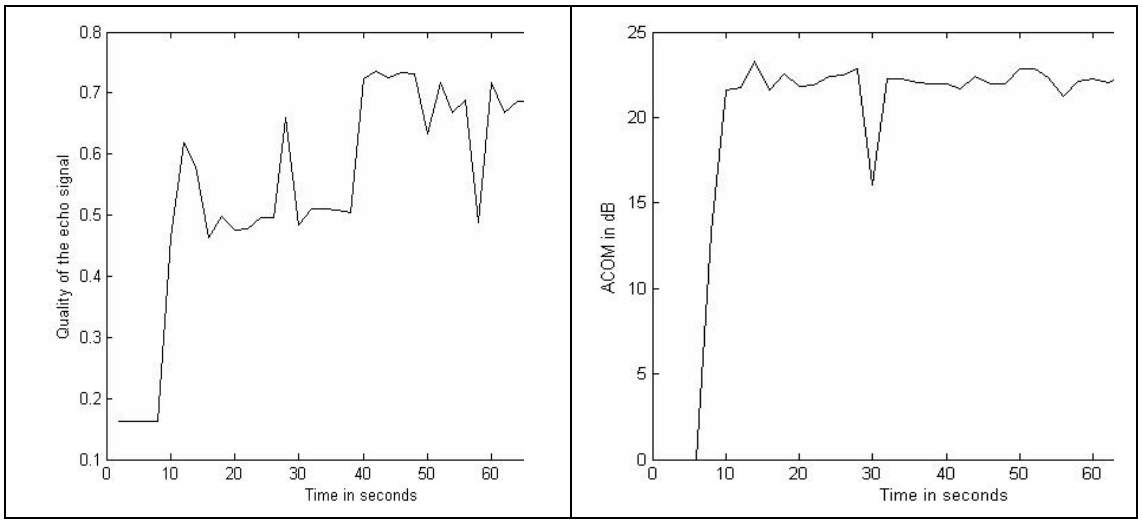


Figure 4-18 Comparing the estimated echo quality with the estimated ACOM

We conclude from these simulations that our proposed algorithm to evaluate the echo component of the voice quality in a VoIP call gave very good results for calls that had little or no echo. That is, the proposed algorithm was able to estimate with good accuracy that the final voice quality was not affected by echo in those calls (Figure 4-14). We should note that due to our choices of defuzzification method and the output membership functions, even for calls with perfect quality the output wouldn't be 1.0 . So the outputs shown in Figure 4-14 really reflect very good voice quality as we expected.

On the other hand, for calls where the voice quality was not so good due either to some high background noise, transmit/receive speech signal level disparity or presence of echo, the algorithm had a fair performance. As we can see in Figure 4-15 the algorithm was able to point out calls with voice quality problem but we still think that some calls got higher scores than expected considering our subjective analysis of the call. For instance, again in Figure 4-15 we got a few calls with scores around 0.6 which fails to point out the real subjective quality of the call which was lower than that. There is not a single reason why these calls got scores higher than expected, once different calls, with different issues (sometime echo, sometimes high background noise) presented high scores. We tried to tune the membership functions and modify or add fuzzy rules in order to improve the output of the algorithm but the results shown in this section are the best we could obtain.

Finally, we suggest that before using this algorithm in a deployed VoIP network, the network provider should run several testing calls in order to calibrate the output values of the algorithm, not the fuzzy rules or the fuzzy membership functions. That is, after running several calls with known voice quality, the provider can generate thresholds to compare against the output of the algorithm in order to determine the voice quality. For instance, in our simulations we could suggest that any output of the

algorithm that is below 0.5 indicates bad voice quality and anything above 0.7 indicates good voice quality. Any call with quality in between these numbers have some moderate voice quality. Which shows that, as designed, the algorithm's output is fuzzy, imprecise and it only gives us a degree of truth.

5 Summary and Future Work

5.1 Summary

VoIP service is bound to grow considerably in the years to come. Even so, VoIP reliability and voice quality remain important factors, specially when compared against the PSTN, that limit the widespread adoption of VoIP in consumer markets. In this work we propose a fuzzy algorithm to estimate the echo quality component of the voice quality in VoIP networks.

The proposed algorithm is a building block (Figure 4-2) of an objective, passive, voice quality algorithm that can run in real-time and estimate the voice quality for live calls in a VoIP system or network. The use of fuzzy logic was motivated by the low computational complexity normally required by such methods, also called soft computing methods. The proposed algorithm can run in real-time in the embedded system that processes the VoIP call, giving operators an almost instantaneous estimation of the quality of their network. Another advantage of the proposed algorithm is that it doesn't require a reference signal and this is another reason why it can be run for real life calls. On the other hand, the algorithm carries a tradeoff between precision and complexity. The main disadvantage of the proposed algorithm is that it is not as precise in its estimates as the PESQ, MOS or the E-model methods. As designed, the algorithm is based on a fuzzy engine and the results of the fuzzy inference system for real calls must be analyzed against results of the algorithm for calls that have known MOS scores.

We simulated the algorithm using Matlab and the simulation results presented in Section 4.4 show that it estimated with very good precision the quality of the echo component for calls with no or very little echo (Figure 4-14). On the other hand, although the algorithm was able to detect and estimate the bad quality of calls with some echo, it also missed to point out other calls that presented bad quality (Figure 4-15).

In this work, we also presented some extensions for the proposed algorithm in order to estimate the voice quality of a VoIP call taking into consideration also the effects of jitter, packet loss and delay (Figure 4-2). In the same way, we proposed building blocks of a more complete algorithm to estimate the performance of a VoIP system or network (Figure 4-3). These and a few other points are described with more detail in the next section, which proposes topics for future work related to this subject.

5.2 Further Work

In this section we discuss two main directions of future developments related to the algorithm proposed in this work.

- We can improve the accuracy of the proposed algorithm.
- We can modify the proposed algorithm so that it can be used in different scenarios.

One possible way of improving the accuracy of a fuzzy inference system is to add more input fuzzy variables and more fuzzy rules. As was said before, a new version of the algorithm with a larger set of fuzzy rules that uses the information of packet loss, delay and jitter may be considered as a good contribution in a future work. It would give a more precise estimate of the voice quality than taking only the echo component into consideration. Also, we should consider the possibilities of including fuzzy rules that incorporate knowledge of other speech parameters that can be affected by the IP network such as speech clarity and loudness. Of course, we will need to add new fuzzy membership functions for the new parameters, new fuzzy rules will be created and some existing fuzzy rules will need to be modified. All these should be followed by an extensive tuning of the membership functions and possibly the rules, using calls with known MOS.

Regarding different scenarios, there are few directions that can be followed in order to increase the range of scenarios that can take advantage of a similar algorithm. For instance, the main results of this work were developed for line echo present in a VoIP call. So, a modified algorithm could be used for similar purposes in a scenario where the echo present in the call has acoustic nature. It would be very useful for network operators to have a similar algorithm to evaluate the quality of an acoustic echo signal. The characteristics of the acoustics echo and acoustic echo cancellers are different from the characteristics of the line echo and line echo cancellers. For instance, the echo path for acoustic echo can be much longer and can vary much more (in length) than for line echo. The acoustic echo generated by a handsfree phone for example will vary based on different room sizes. Also, the adaptive filter algorithm in general is different between the acoustic and the line echo canceller implementations. The bottom line is that the algorithm proposed here is not valid for scenarios where the echo is of acoustic nature and modifying the algorithm to handle this scenario is an important extension of this work.

Still in this context of modifying the proposed algorithm for different scenarios, another interesting direction is to adapt the algorithm for 802.11 or Wi-Fi networks, which are also becoming very popular. In fact, the use of VoIP over such networks is already being called Vo802.11 [R 34]. It seems that the chief challenge to Vo802.11 is that, relative to wired IP networks, packets are dropped at an excessive rate - in general 20% more packets are dropped. This can lead to distortion of the voice to the extent that the conversation is unintelligible and this must be taken into consideration when adapting our proposed algorithm to the Vo802.11 environment. Again, we may

need to add new fuzzy variables and fuzzy rules to deal with the problem of packet loss in wireless IP networks. We should note here that this may not be a trivial task once in the Vo802.11 environment there are new speech processing algorithms that provide diversity, which allow for some speech segment recovery even when a few packets are lost. Of course, this diversity leads to increased delays and that is the price to be paid. So, this trade-off between diversity, delay and packet loss should be included in the fuzzy rules of the modified algorithm.

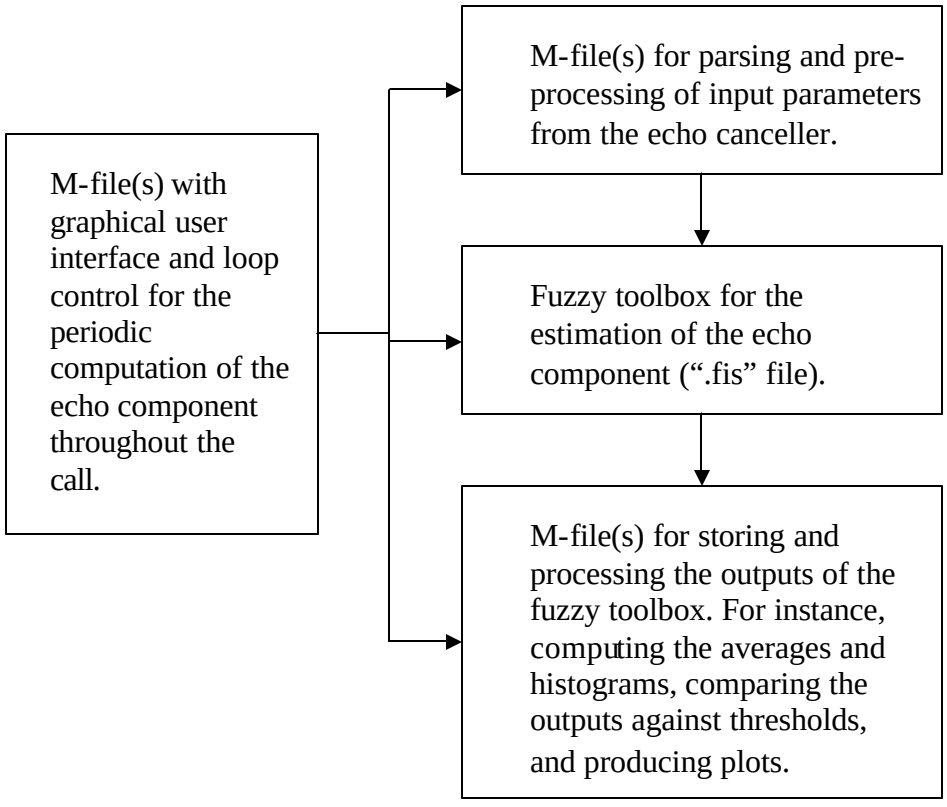
Appendix

The implementation of the proposed fuzzy algorithm was done using the Fuzzy Logic Toolbox provided by Matlab. The toolbox uses files with extension ".fis". We list below the ".fis" file with the necessary parameters to simulate our algorithm. In order to use it in Matlab, we suggest to copy and paste it in a notepad and save it as a "txt" file. Then the user should rename it with a ".fis" extension and load it using the toolbox.

```
/*****************************************/
[System]
Name='Echo_Component_Algorithm'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=4
AndMethod='min'
OrMethod='max'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='ERL'
Range=[6 30]
NumMFs=1
MF1='Good':'trimf',[20 30 30]

[Input2]
Name='ACOM'
Range=[6 40]
NumMFs=3
MF1='Bad':'trimf',[6 6 23]
MF2='Moderate':'trimf',[12 23 36]
MF3='Good':'trimf',[23 40 40]

[Input3]
Name='Transmit_Noise_Power'
Range=[-60 -36]
NumMFs=1
MF2='Bad':'trimf',[-45 -36 -36]

[Input4]
Name='Receive_Speech_Power'
Range=[-30 -5]
NumMFs=2
MF1='Bad1':'trimf',[-30 -30 -25]
MF2='Bad2':'trimf',[-15 -5 -5]

[Output1]
Name='Echo'
```

```
Range=[0 1]
NumMFs=3
MF1='Bad':'trimf',[0 0 0.5]
MF2='Moderate':'trimf',[0 0.5 1]
MF3='Good':'trimf',[0.5 1 1]

[Rules]
0 3 0 0 0, 3 (1) : 1
0 1 0 0 0, 1 (1) : 1
3 2 0 0 0, 2 (1) : 1
0 0 2 1 3, 1 (1) : 1

/*****************************************/
```

This will take care of the main computations of the algorithm, but it will be necessary to use other Matlab M-files that are implementation dependent in order to implement the algorithm. In our simulations we used the following logic diagram of M-files.



Bibliography

[R 1] B. Khasnabish, Implementing Voice over IP, Wiley & Sons, 2003

[R 2] R. Lau, R. Khare, W. Chang, Service Assurance for Voice over WiFi and 3G Networks, Artech House, 2005

[R 3] D. Minoli, E. Minoli, Delivering Voice over IP Networks, Wiley, 2002.

[R 4] M. Manousos, S. Apostolacos, I. Grammatikakis, D. Mexis, D. Kagklis, E. Sykas, "Voice-Quality Monitoring and Control for VoIP," IEEE Internet Computing, July 2005.

[R 5] S. Rein, F. Fitzek, M. Reisslein, "Voice Quality Evaluation in Wireless Packet Communication Systems: A Tutorial and Performance Results for ROHC," IEEE Wireless Communications, Feb. 2005

[R 6] B. Douskalis, IP Telephony: The integration of Robust VoIP Services, Prentice Hall, 2000.

[R 7] A. Lakaniemi, J. Rosti, V. Raisanen, "Subjective VoIP speech quality evaluation based on network measurements," IEEE International Conference on Communications, June 2001.

[R 8] T. Wakahara, Y. Meng, K. Kazaura, M. Matsumoto, T. Shimizu, "A Voice Quality Measurement Method for PHS over IP Communication," IEEE 18th International Conference on Advanced Information Networking and Application, 2004.

[R 9] A. Markopoulou, F. Tobagi, M. Karam, "Assessing the Quality of Voice Communications Over Internet Backbones", IEEE Transactions on Networking, Oct. 2003.

[R 10] L. Sun, E. Ifeachor, "New Models for Perceived Voice Quality Prediction and their Applications in Playout Buffer Optimization for VoIP Networks," IEEE International Conference on Communications, June 2004.

[R 11] F. Hammer, P. Reichl, T. Ziegler, "Where Packet Traces Meet Speech Samples: An Instrumental Approach to Perceptual QoS Evaluation of VoIP," IEEE Twelfth International Workshop on Quality of Service, June 2004.

[R 12] S. El-Hennawey, D. Lee, "Embedded Real-Time Voice Quality Analysis System," Global Signal Processing Conference, Sep. 2004.

[R 13] W. Zha, W. Chan, "Voice Quality Assessment Using Classification Trees," IEEE Asilomar Conference on Signals, Systems and Computers, Nov. 2003.

[R 14] J. James, B. Chen, L. Garrison, "Implementing VoIP: A Voice Transmission Performance Progress Report," IEEE Communications Magazine, July 2004.

[R 15] S. El-Kader, H. Eissa, H. Baraka, "Real-Time Voice Quality Guarantee on the IP Based Networks," ICCS, Nov. 2002.

[R 16] D. G. Messerschmitt, "Echo Cancellation in Speech and Data Transmission," IEEE Journal on Selected Areas in Communications, March 1984.

[R 17] M. Sondhi and D. A. Berkley, "Silencing Echos on the Telephone Network," IEEE Proceedings, Aug. 1980.

[R 18] L. Carvalho, E. Mota, A. Lima, J. Souza, A. Barreto, "An E-Model Implementation for Speech Quality Evaluation in VoIP Systems", 10th IEEE Symposium on Computers and Communications, 2005.

[R 19] O. Hersent, D Gurle, J. Petit, IP Telephony: Packet-based multimedia communications systems, Addison-Wesley 2000.

[R 20] R. Swale, Voice Over IP: Systems and Solutions, The Institution of Electrical Engineers, London, 2001.

[R 21] T. Chan, D. Greenstreet, Building Residencial VoIP Gateways: A Tutorial, Texas Instruments, 2000.

[R 22] D. Armstrong, C. Endicott, "Controlling Echo in the Worldwide Telecommunications Network", Tellabs.

[R 23] J. Davidson, J. Peters, Voice Over IP Fundamentals, Cisco Press, 2000.

[R 24] J. Benesty, T. Gansler, D. Morgan, M. Sondhi, S. Gay, Advances in Network and Acoustic Echo Cancellation, Springer-Verlag, 2001.

[R 25] D. Schobben, Real-time Adaptive Concepts in Acoustics, Kluwer Academic Publishers, 2001.

[R 26] D. Armstrong, "Acoustic Echo Control for Digital Wireless Networks," A Telecommunications Industry Report, Tellabs.

[R 27] T. Ross, Fuzzy Logic with Engineering Applications, Wiley, 2004.

[R 28]A. Ibrahim, Fuzzy Logic for Embedded Systems Applications, Newnes, 2004.

[R 29] R. Berkan, S. Trubatch, Fuzzy Systems Design Principles, IEEE Press, 1997.

[R 30] V. Loia, M. Nikravesh, L. Zadeh, Fuzzy Logic and the Internet, Springer-Verlag, 2004.

[R 31] K. Y. Cai, M. Ying, G. Chen, Fuzzy Logic and Soft Computing, Kluwer Academic Publishers, 1999.

[R 32] S. G. Tzafestas, Soft Computing in Systems and Control Technoloty, World Scientific, 1999.

[R 33] C. Boutremans, J. Le Boudec, "Adaptive Joint Playout Buffer and FEC Adjustment for Internet Telephony," Proceedings of IEEE INFOCOM'2003, April 2003.

[R 34] F. Ohrtman, Voice over 802.11, Artech House, 2004

[R 35] ITU-T Recommendation G.165 for Echo Cancellers, International Telecommunication Union, 1994

[R 36] ITU-T Recommendation G.168 for Digital Network Echo Cancellers