The Speex Codec Manual
(version 1.0)

Jean-Marc Valin

23rd March 2003

Contents

List of Tables

1 Introduction to Speex

The Speex project (`http://www.speex.org/`) has been started because there was a need for a speech codec that was open-source and free from software patents. These are essential conditions for being used by any open-source software. There is already Vorbis that does general audio, but it is not really suitable for speech. Also, unlike many other speech codecs, Speex is not targeted at cell phones (not many open-source cell phones anyway :-)) but rather at voice over IP (VoIP) and file-based compression.

As design goals, we wanted to have a codec that would allow both very good quality speech and low bit-rate (unfortunately not at the same time!), which led us to developing a codec with multiple bit-rates. Of course very good quality also meant we had to do wideband (16 kHz sampling rate) in addition to narrowband (telephone quality, 8 kHz sampling rate).

Designing for VoIP instead of cell phone use means that Speex must be robust to lost packets, but not to corrupted ones since packets either arrive unaltered or don't arrive at all. Also, the idea was to have a reasonable complexity and memory requirement without compromising too much on the efficiency of the codec.

All this led us to the choice of CELP as the encoding technique to use for Speex. One of the main reasons is that CELP has long proved that it could do the job and scale well to both low bit-rates (think DoD CELP @ 4.8 kbps) and high bit-rates (think G.728 @ 16 kbps).

The main characteristics can be summarized as follows:

- Free software/open-source, patent and royalty-free
- Integration of narrowband and wideband in the same bit-stream
- Wide range of bit-rates available (from 2 kbps to 44 kbps)
- Dynamic bit-rate switching and Variable Bit-Rate (VBR)
- Voice Activity Detection (VAD, integrated with VBR)
- Variable complexity
- Ultra-wideband mode at 32 kHz (up to 48 kHz)
- Intensity stereo encoding option

This document is divided in the following way. Section 2 describes the different Speex features and defines some terms that will be used in later sections. Section 3 provides information about the standard command-line tools, while 4 contains information about programming using the Speex API. Section 5 has some information related to Speex and standards. The three last sections describe the internals of the codec and require some signal processing knowledge. Section 6 explains the general idea behind CELP, while sections 7 and 8 are specific to Speex. Note that if you are only interested in using Speex, those three last sections are not required.

2 Feature description

This section explains the main Speex features, as well as some concepts in speech coding that help better understand the next sections.

Sampling rate

Speex is mainly designed for 3 different sampling rates: 8 kHz, 16 kHz, and 32 kHz. These are respectively refered to as narrowband, wideband and ultra-wideband.

Quality

Speex encoding is controlled most of the time by a quality parameter that ranges from 0 to 10. In constant bit-rate (CBR) operation, the quality parameter is an integer, while for variable bit-rate (VBR), the parameter is a float.

Complexity (variable)

With Speex, it is possible to vary the complexity allowed for the encoder. This is done by controlling how the search is performed with an integer ranging from 1 to 10 in a way that's similar to the -1 to -9 options to *gzip* and *bzip2* compression utilities. For normal use, the noise level at complexity 1 is between 1 and 2 dB higher than at complexity 10, but the CPU requirements for complexity 10 is about 5 times higher than for complexity 1. In practice, the best trade-off is between complexity 2 and 4, though higher settings are often useful when encoding non-speech sounds like DTMF tones.

Variable Bit-Rate (VBR)

Variable bit-rate (VBR) allows a codec to change its bit-rate dynamically to adapt to the "difficulty" of the audio being encoded. In the example of Speex, sounds like vowels and high-energy transients require a higher bit-rate to achieve good quality, while fricatives (e.g. s,f sounds) can be coded adequately with less bits. For this reason, VBR can achive lower bit-rate for the same quality, or a better quality for a certain bit-rate. Despite its advantages, VBR has two main drawbacks: first, by only specifying quality, there's no guaranty about the final average bit-rate. Second, for some real-time applications like voice over IP (VoIP), what counts is the maximum bit-rate, which must be low enough for the communication channel.

Average Bit-Rate (ABR)

Average bit-rate solves one of the problems of VBR, as it dynamically adjusts VBR quality in order to meet a specific target bit-rate. Because the quality/bit-rate is adjusted in real-time (open-loop), the global quality will be slightly lower than that obtained by encoding in VBR with exactly the right quality setting to meet the target average bit-rate.

Voice Activity Detection (VAD)

When enabled, voice activity detection detects whether the audio being encoded is speech or silence/background noise. VAD is always implicitly activated when encoding in VBR, so the option is only useful in non-VBR operation. In this case, Speex detects non-speech periods and encode them with just enough bits to reproduce the background noise. This is called “comfort noise generation” (CNG).

Discontinuous Transmission (DTX)

Discontinuous transmission is an addition to VAD/VBR operation, that allows to stop transmitting completely when the background noise is stationary. In file-based operation, since we cannot just stop writing to the file, only 5 bits are used for such frames (corresponding to 250 bps).

Perceptual enhancement

Perceptual enhancement is a part of the decoder which, when turned on, tries to reduce (the perception of) the noise produced by the coding/decoding process. In most cases, perceptual enhancement make the sound further from the original *objectively* (if you use SNR), but in the end it still *sounds* better (subjective improvement).

Algorithmic delay

Every speech codec introduces a delay in the transmission. For Speex, this delay is equal to the frame size, plus some amount of “look-ahead” required to process each frame. In narrowband operation (8 kHz), the delay is 30 ms, while for wideband (16 kHz), the delay is 34 ms. These values don’t account for the CPU time it takes to encode or decode the frames.

3 Command-line encoder/decoder

The base Speex distribution includes a command-line encoder (*speexenc*) and decoder (*speexdec*). This section describes how to use these tools.

3.1 *speexenc*

The *speexenc* utility is used to create Speex files from raw PCM or wave files. It can be used by calling:

```
speexenc [options] input_file output_file
```

The value '-' for input_file or output_file corresponds respectively to stdin and stdout. The valid options are:

–narrowband (-n) Tell Speex to treat the input as narrowband (8 kHz). This is the default

–wideband (-w) Tell Speex to treat the input as wideband (16 kHz)

–ultra-wideband (-u) Tell Speex to treat the input as "ultra-wideband" (32 kHz)

–quality n Set the encoding quality (0-10), default is 8

–bitrate n Encoding bit-rate (use bit-rate n or lower)

–vbr Enable VBR (Variable Bit-Rate), disabled by default

–abr n Enable ABR (Average Bit-Rate) at n kbps, disabled by default

–vad Enable VAD (Voice Activity Detection), disabled by default

–dtx Enable DTX (Discontinuous Transmission), disabled by default

–nframes n Pack n frames in each Ogg packet (this saves space at low bit-rates)

–comp n Set encoding speed/quality tradeoff. The higher the value of n, the slower the encoding (default is 3)

-V Verbose operation, print bit-rate currently in use

–help (-h) Print the help

–version (-v) Print version information

Speex comments

–comment Add the given string as an extra comment. This may be used multiple times.

–author Author of this track.

–title Title for this track.

Raw input options

–rate n Sampling rate for raw input

–stereo Consider raw input as stereo

–le Raw input is little-endian

–be Raw input is big-endian

–8bit Raw input is 8-bit unsigned

–16bit Raw input is 16-bit signed

3.2 *speexdec*

The *speexdec* utility is used to decode Speex files and can be used by calling:

```
speexdec [options] speex_file [output_file]
```

The value '-' for input_file or output_file corresponds respectively to stdin and stdout. Also, when no output_file is specified, the file is played to the soundcard. The valid options are:

–enh enable post-filter (default)

–no-enh disable post-filter

–force-nb Force decoding in narrowband

–force-wb Force decoding in wideband

–force-uwb Force decoding in ultra-wideband

–mono Force decoding in mono

–stereo Force decoding in stereo

–rate n Force decoding at n Hz sampling rate

–packet-loss n Simulate n % random packet loss

-V Verbose operation, print bit-rate currently in use

–help (-h) Print the help

–version (-v) Print version information

4 Programming with Speex (the libspeex API)

This section explains how to use the Speex API. Examples of code can also be found in appendix B.

4.1 Encoding

In order to encode speech using Speex, you first need to:

```
#include <speex.h>
```

You then need to declare a Speex bit-packing struct

```
SpeexBits bits;
```

and a Speex encoder state

```
void *enc_state;
```

The two are initialized by:

```
speex_bits_init(&bits);
enc_state = speex_encoder_init(&speex_nb_mode);
```

For wideband coding, *speex_nb_mode* will be replaced by *speex_wb_mode*. In most cases, you will need to know the frame size used by the mode you are using. You can get that value in the *frame_size* variable with:

```
speex_encoder_ctl(enc_state,SPEEX_GET_FRAME_SIZE,&frame_size);
```

Once the initialization is done, for every input frame:

```
speex_bits_reset(&bits);
speex_encode(enc_state, input_frame, &bits);
nbBytes = speex_bits_write(&bits, byte_ptr, MAX_NB_BYTES);
```

where *input_frame* is a *(float *)* pointing to the beginning of a speech frame, *byte_ptr* is a *(char *)* where the encoded frame will be written, *MAX_NB_BYTES* is the maximum number of bytes that can be written to *byte_ptr* without causing an overflow and *nbBytes* is the number of bytes actually written to *byte_ptr* (the encoded size in bytes). Before calling speex_bits_write, it is possible to find the number of bytes that need to be written by calling `speex_bits_nbytes(&bits)`, which returns a number of bytes.

After you're done with the encoding, free all resources with:

```
speex_bits_destroy(&bits);
speex_encoder_destroy(enc_state);
```

That's about it for the encoder.

4.2 Decoding

In order to encode speech using Speex, you first need to:

```
#include <speex.h>
```

You also need to declare a Speex bit-packing struct

```
SpeexBits bits;
```

and a Speex encoder state

```
void *dec_state;
```

The two are initialized by:

```
speex_bits_init(&bits);
dec_state = speex_decoder_init(&speex_nb_mode);
```

For wideband decoding, *speex_nb_mode* will be replaced by *speex_wb_mode*. If you need to obtain the size of the frames that will be used by the decoder, you can get that value in the *frame_size* variable with:

```
speex_decoder_ctl(dec_state, SPEEX_GET_FRAME_SIZE, &frame_size);
```

There is also a parameter that can be set for the decoder: whether or not to use a perceptual post-filter. This can be set by:

```
speex_decoder_ctl(dec_state, SPEEX_SET_ENH, &enh);
```

where *enh* is an int that with value 0 to have the post-filter disabled and 1 to have it enabled.

Again, once the decoder initialization is done, for every input frame:

```
speex_bits_read_from(&bits, input_bytes, nbBytes);
speex_decode(st, &bits, output_frame);
```

where input_bytes is a *(char *)* containing the bit-stream data received for a frame, *nbBytes* is the size (in bytes) of that bit-stream, and *output_frame* is a *(float *)* and points to the area where the decoded speech frame will be written. A NULL value as the first argument indicates that we don't have the bits for the current frame. When a frame is lost, the Speex decoder will do its best to "guess" the correct signal.

After you're done with the decoding, free all resources with:

```
speex_bits_destroy(&bits);
speex_decoder_destroy(dec_state);
```

4.3 Codec Options (speex_*_ctl)

The Speex encoder and decoder support many options and requests that can be accessed through the *speex_encoder_ctl* and *speex_decoder_ctl* functions. These functions are similar to the *ioctl* system call and their prototypes are:

```
void speex_encoder_ctl(void *encoder, int request, void *ptr);
void speex_decoder_ctl(void *encoder, int request, void *ptr);
```

The different values of request allowed are (note that some only apply to the encoder or the decoder):

SPEEX_SET_ENH** Set perceptual enhancer to on (1) or off (0) (integer)

SPEEX_GET_ENH** Get perceptual enhancer status (integer)

SPEEX_GET_FRAME_SIZE Get the frame size used for the current mode (integer)

SPEEX_SET_QUALITY* Set the encoder speech quality (integer 0 to 10)

SPEEX_GET_QUALITY* Get the current encoder speech quality (integer 0 to 10)

SPEEX_SET_MODE*†

SPEEX_GET_MODE*†

SPEEX_SET_LOW_MODE*†

SPEEX_GET_LOW_MODE*†

SPEEX_SET_HIGH_MODE*†

SPEEX_GET_HIGH_MODE*†

SPEEX_SET_VBR* Set variable bit-rate (VBR) to on (1) or off (0) (integer)

SPEEX_GET_VBR* Get variable bit-rate (VBR) status (integer)

SPEEX_SET_VBR_QUALITY* Set the encoder VBR speech quality (float 0 to 10)

SPEEX_GET_VBR_QUALITY* Get the current encoder VBR speech quality (float 0 to 10)

SPEEX_SET_COMPLEXITY* Set the CPU resources allowed for the encoder (integer 1 to 10)

SPEEX_GET_COMPLEXITY* Get the CPU resources allowed for the encoder (integer 1 to 10)

SPEEX_SET_BITRATE* Set the bit-rate to use to the closest value not exceeding the parameter (integer in bps)

SPEEX_GET_BITRATE Get the current bit-rate in use (integer in bps)

SPEEX_SET_SAMPLING_RATE Set real sampling rate (integer in Hz)

SPEEX_GET_SAMPLING_RATE Get real sampling rate (integer in Hz)

SPEEX_RESET_STATE Reset the encoder/decoder state to its original state (zeros all memories)

SPEEX_SET_VAD* Set voice activity detection (VAD) to on (1) or off (0) (integer)

SPEEX_GET_VAD* Get voice activity detection (VAD) status (integer)

SPEEX_SET_DTX* Set discontinuous transmission (DTX) to on (1) or off (0) (integer)

SPEEX_GET_DTX* Get discontinuous transmission (DTX) status (integer)

SPEEX_SET_ABR* Set average bit-rate (ABR) to a value n in bits per second (integer in bps)

SPEEX_GET_ABR* Get average bit-rate (ABR) setting (integer in bps)

***** applies only to the encoder

****** applies only to the decoder

† normally only used internally

4.4 Mode queries

Speex modes have a query system similar to the speex_encoder_ctl and speex_decoder_ctl calls. Since modes are read-only, it is only possible to get information about a particular mode. The function used to do that is:

```
void speex_mode_query(SpeexMode *mode, int request, void *ptr);
```

The admissible values for request are (unless otherwise note, the values are returned through *ptr*):

SPEEX_MODE_FRAME_SIZE Get the frame size (in samples) for the mode

SPEEX_SUBMODE_BITRATE Get the bit-rate for a submode number specified through *ptr* (integer in bps).

4.5 Packing and in-band signalling

Sometimes it is desirable to pack more than one frame per packet (or other basic unit of storage). The proper way to do it is to call speex_encode *N* times before writing the stream with speex_bits_write. In cases where the number of frames is not determined by an out-of-band mechanism, it is possible to include a terminator code. That terminator consists of the code 15 (decimal) encoded with 5 bits, as shown in figure 4.

It is also possible to send in-band "messages" to the other side. All these messages are encoded as "pseudo-frames" of mode 14 which contain a 4-bit message type code, followed by the message. Table 1 lists the available codes, their meaning and the size of the message that follows. Most of these messages are requests that are sent to the encoder or decoder on the other end, which is free to comply or ignore them. By default, all in-band messages are ignored.

Code	Size (bits)	Content
0	1	Asks decoder to set perceptual enhancement off (0) or on(1)
1	1	Asks (if 1) the encoder to be less "agressive" due to high packet loss
2	4	Asks encoder to switch to mode N
3	4	Asks encoder to switch to mode N for low-band
4	4	Asks encoder to switch to mode N for high-band
5	4	Asks encoder to switch to quality N for VBR
6	4	Request acknowloedge (0=no, 1=all, 2=only for in-band data)
7	4	Asks encoder to set CBR (0), VAD(1), DTX(3), VBR(5), VBR+DTX(7)
8	8	Transmit (8-bit) character to the other end
9	8	Intensity stereo information
10	16	Announce maximum bit-rate acceptable (N in bytes/second)
11	16	reserved
12	32	Acknowledge receiving packet N
13	32	reserved
14	64	reserved
15	64	reserved

Table 1: In-band signalling codes

Finally, applications may define custom in-band messages using mode 13. The size of the message in bytes is encoded with 5 bits, so that the decoder can skip it if it doesn't know how to interpret it.

5 Formats and standards

Speex can encode speech in both narrowband and wideband and provides different bit-rates. However, not all features need to be supported by a certain implementation or device. In order to be called "Speex compatible" (whatever that means), an implementation must implement at least a basic set of features.

At the minimum, all narrowband modes of operation MUST be supported at the decoder. This includes the decoding of a wideband bit-stream by the narrowband decoder[1]. If present, a wideband decoder MUST be able to decode a narrowband stream, and MAY either be able to decode all wideband modes or be able to decode the embedded narrowband part of all modes (which includes ignoring the high-band bits).

For encoders, at least one narrowband or wideband mode MUST be supported. The main reason why all encoding modes do not have to be supported is that some platforms may not be able to handle the complexity of encoding in some modes.

5.1 RTP Payload Format

The RTP payload draft is included in appendix C and the latest version is available at `http://www.speex.org/drafts/latest`. This draft has been sent (2003/02/26) to the Internet Engineering Task Force (IETF) and will be discussed at the March 18th meeting in San Francisco.

5.2 MIME Type

For now, you should use the MIME type audio/x-speex for Speex. We will apply for type `audio/speex` in the near future.

5.3 Ogg file format

Speex bit-streams can be stored in Ogg files. In this case, the first packet of the Ogg file contains the Speex header described in table 2. All integer fields in the headers are stored as little-endian. The `speex_string` field must contain the "`Speex   `" (with 3 training spaces), which identifies the bit-stream. The next field, `speex_version` contains the version of Speex that encoded the file. For now, refer to speex_header.[ch] for more info. The *beginning of stream* (`b_o_s`) flag is set to 1 for the header. The header packet has `packetno=0` and `granulepos=0`.

The second packet contains the Speex comment header. The format used is the Vorbis comment format described here: http://www.xiph.org/ogg/vorbis/doc/v-comment.html . This packet has `packetno=1` and `granulepos=0`.

The third and subsequent packets each contain one or more (number found in header) Speex frames. These are identified with `packetno` starting from 2 and the `granulepos` is the number of the last sample encoded in that packet. The last of these packets has the *end of stream* (`e_o_s`) flag is set to 1.

[1] The wideband bit-stream contains an embedded narrowband bit-stream which can be decoded alone

Field	Type	Size
speex_string	char[]	8
speex_version	char[]	20
speex_version_id	int	4
header_size	int	4
rate	int	4
mode	int	4
mode_bitstream_version	int	4
nb_channels	int	4
bitrate	int	4
frame_size	int	4
vbr	int	4
frames_per_packet	int	4
extra_headers	int	4
reserved1	int	4
reserved2	int	4

Table 2: Ogg/Speex header packet

6 Introduction to CELP Coding

Speex is based on CELP, which stands for Code Excited Linear Prediction. This section attempts to introduce the principles behind CELP, so if you are already familiar with CELP, you can safely skip to section 7. The CELP technique is based on three ideas:

1. The use of a linear prediction (LP) model to model the vocal tract
2. The use of (adaptive and fixed) codebook entries as input (excitation) of the LP model
3. The search performed in closed-loop in a "perceptually weighted domain"

This section describes the basic ideas behind CELP. Note that it's still incomplete.

6.1 Linear Prediction (LPC)

Linear prediction is at the base of many speech coding techniques, including CELP. The idea behind it is to predict the signal $x[n]$ using a linear combination of its past samples:

$$y[n] = \sum_{i=1}^{N} a_i x[n-i]$$

where $y[n]$ is the linear prediction of $x[n]$. The prediction error is thus given by:

$$e[n] = x[n] - y[n] = x[n] - \sum_{i=1}^{N} a_i x[n-i]$$

The goal of the LPC analysis is to find the best prediction coefficients a_i which minimize the quadratic error function:

$$E = \sum_{n=0}^{L-1} [e[n]]^2 = \sum_{n=0}^{L-1} \left[x[n] - \sum_{i=1}^{N} a_i x[n-i] \right]^2$$

That can be done by making all derivatives $\frac{\partial E}{\partial a_i}$ equal to zero:

$$\frac{\partial E}{\partial a_i} = \frac{\partial}{\partial a_i} \sum_{n=0}^{L-1} \left[x[n] - \sum_{i=1}^{N} a_i x[n-i] \right]^2 = 0$$

The a_i filter coefficients are computed using the Levinson-Durbin algorithm, which starts from the auto-correlation $R(m)$ of the signal $x[n]$.

$$R(m) = \sum_{i=0}^{N-1} x[i]x[i-m]$$

For an order N filter, we have:

$$\mathbf{R} = \begin{bmatrix} R(0) & R(1) & \cdots & R(N-1) \\ R(1) & R(0) & \cdots & R(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & \cdots & R(0) \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(N) \end{bmatrix}$$

The filter coefficients a_i are found by solving the system $\mathbf{Ra} = \mathbf{r}$. What the Levinson-Durbin algorithm does here is making the solution to the problem $O\left(N^2\right)$ instead of $O\left(N^3\right)$ by exploiting the fact that matrix $\mathbf{R}$ is toeplitz hermitian. Also, it can be proven that all the roots of $A(z)$ are within the unit circle, which means that $1/A(z)$ is always stable. This is in theory; in practice because of finite precision, there are two commonly used techniques to make sure we have a stable filter. First, we multiply $R(0)$ by a number slightly above one (such as 1.0001), which is equivalent to adding noise to the signal. Also, we can apply a window to the auto-correlation, which is equivalent to filtering in the frequency domain, reducing sharp resonances.

The linear prediction model represents each speech sample as a linear combination of past samples, plus an error signal called the excitation (or residual).

$$x[n] = \sum_{i=1}^{N} a_i x[n-i] + e[n]$$

In the z-domain, this can be expressed as

$$x(z) = \frac{1}{A(z)} e(z)$$

where $A(z)$ is defined as

$$A(z) = 1 - \sum_{i=1}^{N} a_i z^{-i}$$

We usually refer to $A(z)$ as the analysis filter and $1/A(z)$ as the synthesis filter. The whole process is called short-term prediction as it predicts the signal $x[n]$ using a prediction using only the N past samples, where N is usually around 10.

Because LPC coefficients have very little robustness to quantization, they are converted to Line Spectral Pair (LSP) coefficients which have a much better behaviour with quantization, one of them being that it's easy to keep the filter stable.

6.2 Pitch Prediction

During voiced segments, the speech signal is periodic, so it is possible to take advantage of that property by approximating the excitation signal $e[n]$ by a gain times the past of the excitation:

$$e[n] \simeq p[n] = \beta e[n-T]$$

where T is the pitch period, β is the pitch gain. We call that long-term prediction since the excitation is predicted from $e[n-T]$ with $T \gg N$.

6.3 Innovation Codebook

The final excitation $e[n]$ will be the sum of the pitch prediction and an *innovation* signal $c[n]$ taken from a fixed codebook, hence the name *Code* Excited Linear Prediction. The final excitation is given by:

$$e[n] = p[n] + c[n] = \beta e[n-T] + c[n]$$

The quantization of $c[n]$ is where most of the bits in a CELP codec are allocated. It represents the information that couldn't be obtained either from linear prediction or pitch prediction. In the z-domain we can represent the final signal $X(z)$ as

$$X(z) = \frac{C(z)}{A(z)\left(1 - \beta z^{-T}\right)}$$

6.4 Analysis-by-Synthesis and Error Weighting

Most (if not all) modern audio codecs attempt to "shape" the noise so that it appears mostly in the frequency regions where the ear cannot detect it. For example, the ear is more tolerant to noise in parts of the spectrum that are louder and *vice versa*. That's why instead of minimizing the simple quadratic error

$$E = \sum_n (x[n] - \bar{x}[n])^2$$

where $\bar{x}[n]$ is the encoder signal, we minimize the error for the perceptually weighted signal

$$X_w(z) = W(z)X(z)$$

where $W(z)$ is the weighting filter, usually of the form

$$W(z) = \frac{A\left(\frac{z}{\gamma_1}\right)}{A\left(\frac{z}{\gamma_2}\right)} \tag{1}$$

with control parameters $\gamma_1 > \gamma_2$. If the noise is white in the perceptually weighted domain, then in the signal domain its spectral shape will be of the form

$$A_{noise}(z) = \frac{1}{W(z)} = \frac{A\left(\frac{z}{\gamma_2}\right)}{A\left(\frac{z}{\gamma_1}\right)}$$

If a filter $A(z)$ has (complex) poles at p_i in the z-plane, the filter $A(z/\gamma)$ will have its poles at $p'_i = \gamma p_i$, making it a flatter version of $A(z)$.

Analysis-by-synthesis refers to the fact that when trying to find the best pitch parameters (T, β) and innovation signal $c[n]$, we do not work by making the excitation $e[n]$ as close as the original one (which would be simpler), but apply the synthesis (and weighting) filter and try making $X_w(z)$ as close to the original as possible.

7 Speex narrowband mode

This section looks at how Speex works for narrowband (8 kHz sampling rate) operation. The frame size for this mode is 20 ms, corresponding to 160 samples. Each frame is also subdivided into 4 sub-frames of 40 samples each.

Also many design decisions were based on the original goals and assumptions:

- Minimizing the amount of information extracted from past frames (for robustness to packet loss)
- Dynamically-selectable codebooks (LSP, pitch and innovation)
- sub-vector fixed (innovation) codebooks

7.1 LPC Analysis

An LPC analysis is first performed on a (asymetric Hamming) window that spans all of the current frame and half a frame in advance. The LPC coefficients are then converted to Line Spectral Pair (LSP), a representation that is more robust to quantization. The LSP's are considered to be associated to the 4^{th} sub-frames and the LSP's associated to the first 3 sub-frames are linearly interpolated using the current and previous LSP's.

The LSP's are encoded using 30 bits for higher quality modes and 18 bits for lower quality, through the use of a multi-stage split-vector quantizer. For the lower quality modes, the 10 coefficients are first quantized with 6 bits and the error is then divided in two 5-coefficient sub-vectors. Each of them is quantized with 6 bits, for a total of 18 bits. For the higher quality modes, the remaining error on both sub-vectors is further quantized with 6 bits each, for a total of 30 bits.

The perceptual weighting filter $W(z)$ used by Speex is derived from the LPC filter $A(z)$ and corresponds to the one described by eq. 1 with $\gamma_1 = 0.9$ and $\gamma_2 = 0.6$. We can use the unquantized $A(z)$ filter since the weighting filter is only used in the encoder.

7.2 Pitch Prediction (adaptive codebook)

Speex uses a 3-tap prediction for pitch. That is, the pitch prediction signal $p[n]$ is obtained by the past of the excitation by:

$$p[n] = \beta_0 e[n-T-1] + \beta_1 e[n-T] + \beta_2 e[n-T+1]$$

where T is the pitch period and the β_i are the prediction (filter) taps. It is worth noting that when the pitch is smaller than the sub-frame size, we repeat the excitation at a period T. For example, when $n-T+1$, we use $n-2T+1$ instead. The period and quantized gains are determined in closed loop (analysis-by-synthesis). In most modes, the pitch period is encoded with 7 bits in the $[17, 144]$ range and the β_i coefficients are vector-quantized using 7 bits (15 kbps narrowband and above) at higher bit-rates and 5 bits at lower bit-rates (11 kbps narrowband and below).

7.3 Innovation Codebook

In Speex, the innovation signal is quantized using sub-vector shape-only vector quantization (VQ). That means that the innovation signal is divided in sub-vectors (of size 5 to 20) and quantized using a codebook that represents both the shape and the gain at the same time. This saves many bits that would otherwise be allocated for a separate gain at the price of a slight increase in complexity.

7.4 Bit allocation

There are 7 different narrowband bit-rates defined for Speex, ranging from 250 bps to 24.6 kbps, although the modes below 5.9 kbps should not be used for speech. The bit-allocation for each mode is detailed in table 3. Each frame starts with the mode ID encoded with 4 bits which allows a range from 0 to 15, though only the first 7 values are used (the others are reserved). The parameters are listed in the table in the order they are packed in the bit-stream. All frame-based parameters are packed before sub-frame parameters. The parameters for a certain sub-frame are all packed before the following sub-frame is packed. Note that the "OL" in the parameter description means that the parameter is an open loop estimation based on the whole frame.

Parameter	Update rate	0	1	2	3	4	5	6	7	8
Wideband bit	frame	1	1	1	1	1	1	1	1	1
Mode ID	frame	4	4	4	4	4	4	4	4	4
LSP	frame	0	18	18	18	18	30	30	30	18
OL pitch	frame	0	7	7	0	0	0	0	0	7
OL pitch gain	frame	0	4	0	0	0	0	0	0	4
OL Exc gain	frame	0	5	5	5	5	5	5	5	5
Fine pitch	sub-frame	0	0	0	7	7	7	7	7	0
Pitch gain	sub-frame	0	0	5	5	5	7	7	7	0
Innovation gain	sub-frame	0	1	0	1	1	3	3	3	0
Innovation VQ	sub-frame	0	0	16	20	35	48	64	96	10
Total	frame	5	43	119	160	220	300	364	492	79

Table 3: Bit allocation for narrowband modes

So far, no MOS (Mean Opinion Score) subjective evaluation has been performed for Speex. In order to give an idea of the quality achivable with it, table 4 presents my own subjective opinion on it. It sould be noted that different people will perceive the quality differently and that the person that designed the codec often has a bias (one way or another) when it comes to subjective evaluation. Last thing, it should be noted that for most codecs (including Speex) encoding quality sometimes varies depending on the input. Note that the complexity is only approximate (within 0.5 mflops and using the lowest complexity setting). Decoding requires approximately 0.5 mflops in most modes (1 mflops with perceptual enhancement).

Mode	Bit-rate (bps)	mflops	Quality/description
0	250	N/A	No transmission (DTX)
1	2,150	6	Vocoder (mostly for comfort noise)
2	5,950	9	Very noticeable artifacts/noise, good intelligibility
3	8,000	10	Artifacts/noise sometimes noticeable
4	11,000	14	Artifacts usually noticeable only with headphones
5	15,000	11	Need good headphones to tell the difference
6	18,200	17.5	Hard to tell the difference even with good headphones
7	24,600	14.5	Completely transparent for voice, good quality music
8	3,950	10.5	Very noticeable artifacts/noise, good intelligibility
9	N/A	N/A	reserved
10	N/A	N/A	reserved
11	N/A	N/A	reserved
12	N/A	N/A	reserved
13	N/A	N/A	Application-defined, interpreted by callback or skipped
14	N/A	N/A	Speex in-band signaling
15	N/A	N/A	Terminator code

Table 4: Quality versus bit-rate

7.5 Perceptual enhancement

This part of the codec only applies to the decoder and can even be changed without affecting inter-operability. For that reason, the implementation provided and described here should only be considered as a reference implementation. The enhancement system is divided into two parts. First, the synthesis filter $S(z) = 1/A(z)$ is replaced by an enhanced filter

$$S'(z) = \frac{A(z/a_2)A(z/a_3)}{A(z)A(z/a_1)}$$

where a_1 and a_2 depend on the mode in use and $a_3 = \frac{1}{r}\left(1 - \frac{1-ra_1}{1-ra_2}\right)$ with $r = .9$. The second part of the enhancement consists of using a comb filter to enhance the pitch in the excitation domain.

8 Speex wideband mode (sub-band CELP)

For wideband, the Speex approach uses a *q*uadrature *m*irror *f*ilter (QMF) to split the band in two. The 16 kHz signal is thus divided into two 8 kHz signals, one representing the low band (0-4 kHz), the other the high band (4-8 kHz). The low band is encoded with the narrowband mode described in section 7 in such a way that the resulting "embedded narrowband bit-stream" can also be decoded with the narrowband decoder. Since the low band encoding has already been described, only the high band encoding is described in this section.

8.1 Linear Prediction

The linear prediction part used for the high-band is very similar to what is done for narrowband. The only difference is that we use only 12 bits to encode the high-band LSP's using a multi-stage vector quantizer (MSVQ). The first level quantizes the 10 coefficients with 6 bits and the error is then quantized using 6 bits, too.

8.2 Pitch Prediction

That part is easy: there's no pitch prediction for the high-band. There are two reasons for that. First, there is usually little harmonic structure in this band (above 4 kHz). Second, it would be very hard to implement since the QMF folds the 4-8 kHz band into 4-0 kHz (reversing the frequency axis), which means that the location of the harmonics is no longer at multiples of the fundamental (pitch).

8.3 Excitation Quantization

The high-band excitation is coded in the same way as for narrowband.

8.4 Bit allocation

For the wideband mode, the entire narrowband frame is packed before the high-band is encoded. The narrowband part of the bit-stream is as defined in table 3. The high-band follows, as described in table 5. This also means that a wideband frame may be correctly decoded by a narrowband decoder with the only caveat that if more than one frame is packed in the same packet, the decoder will need to skip the high-band parts in order to sync with the bit-stream.

Parameter	Update rate	0	1	2	3	4
Wideband bit	frame	1	1	1	1	1
Mode ID	frame	3	3	3	3	3
LSP	frame	0	12	12	12	12
Excitation gain	sub-frame	0	5	4	4	4
Excitation VQ	sub-frame	0	0	20	40	80
Total	frame	4	36	112	192	352

Table 5: Bit allocation for high-band in wideband mode

A FAQ

Vorbis is open-source and patent-free; why do we need Speex?

Vorbis is a great project but its goals are not the same as Speex. Vorbis is mostly aimed at compressing music and audio in general, while Speex targets speech only. For that reason Speex can achieve much better results than Vorbis on speech, typically 2-4 times higher compression at equal quality.

Isn't there a GPL implementation of the GSM-FR codec? Why is Speex necessary?

First of all, it's not clear whether GSM-FR is covered by a Phillips patent (see http://kbs.cs.tu-berlin.de/~jutta/toast.html). Also, GSM-FR offers mediocre quality at a relatively high bit-rate, while Speex can offer equivalent quality at almost half the bit-rate. Last but not least, Speex offers a wide range of bit-rates and sampling rates, while GSM-FR is limited to 8 kHz speech at 13 kbps.

Under what license is Speex released?

As of version 1.0 beta 1, Speex is released under Xiph's BSD-like license. This license is the most permissive of the open-source licenses.

Ogg, Speex, Vorbis, what's the difference?

Ogg is a container format for holding multimedia data. Vorbis is an audio codec that uses Ogg to store its bit-streams as files, hence the name Ogg Vorbis. Speex also uses the Ogg format to store its bit-streams as files, so technically they would be "Ogg Speex" files (I prefer to call them just Speex files). One difference with Vorbis however, is that Speex is less tied with Ogg. Actually, if what you do is Voice of IP (VoIP), you don't need Ogg at all.

What's the extension for Speex?

Speex files have the .spx extension. Note, however that the Speex tools (speexenc, speexdec) do not rely on the extension at all, so any extension will work.

Can I use Speex for compressing music?

Just like Vorbis is not really adapted to speech, Speex is really not adapted for music. In most cases, you'll be better of with Vorbis when it comes to music.

I converted some MP3's to Speex and the quality is bad. What's wrong?

This is called transcoding and it will always result in much poorer quality than the original MP3. Unless you have a really good (size) reason to do so, never transcode speech. This is even valid for self transcoding (tandeming), i.e. If you decode a Speex file and re-encode it again at the same bit-rate, you will lose quality.

Does Speex run on Windows?

Compilation on Windows has been supported since version 0.8.0. There are also several front-ends available from the website.

Why is encoding so slow compared to decoding?

For most kinds of compression, encoding is inherently slower than decoding. In the case of Speex, encoding consists of finding, for each vector of 5 to 10 samples, the entry that matches the best within a codebook consisting of 16 to 256 entries. On the other hand, at decoding all that needs to be done is look up the right entry in the codebook using the encoded index. Since a lookup is much faster than a search, the decoder works much faster than the encoder.

Why is Speex so slow on my iPaq (or insert any platform without an FPU)?

Well, the parenthesis provides the answer: no FPU (floating-point unit). The Speex code makes heavy use of floating-point operations. On devices with no FPU, all floating-point instructions need to be emulated. This is a very time consuming operation.

I'm getting unusual background noise (hiss) when using libspeex in my application. How do I fix that?

One of the causes could be scaling of the input speech. Speex expects signals to have a $\pm 2^{15}$ (signed short) dynamic range. If the dynamic range of your signals is too small (e.g. ± 1.0), you will suffer important quantization noise. A good target is to have a dynamic range around ± 8000 which is large enough, but small enough to make sure there's no clipping when converting back to signed short.

I get very distorted speech when using libspeex in my application. What's wrong?

There are many possible causes for that. One of them is errors in the way the bits are manipulated. Another possible cause is the use of the same encoder or decoder state for more than one audio stream (channel), which produces strange effects with the filter memories. If the input speech has an amplitude close to $\pm 2^{15}$, it is possible that at decoding, the amplitude be a bit higher than that, causing clipping when saving as 16-bit PCM.

How does Speex compare to other proprietary codecs?

It's hard to give precise figures since no formal listening tests have been performed yet. All I can say is that in terms of quality, Speex competes on the same ground as other proprietary codecs (not necessarily the best, but not the worst either). Speex also has many features that are not present in most other codecs. These include variable bit-rate (VBR), integration of narrowband and wideband, as well as stereo support. Of course, another area where Speex is really hard to beat is the quality/price ratio. Unlike many very expensive codecs, Speex is free and anyone may distribute/modify it at will.

Can Speex pass DTMF?

I guess it all depends on the bit-rate used. Though no formal testing has yet been performed, I'd say don't go below the 15 kbps mode if you want DTMF to be transmitted correctly. DTMF at 8 kbps may work but your mileage may vary. Also, make sure you don't use the lowest complexity (see SPEEX_SET_COMPLEXITY or –comp option), as it causes significant noise.

Can Speex pass V.9x modem signals correctly?

If I could do that I'd be very rich by now :-)

What is your (Jean-Marc) relationship with the University of Sherbrooke and how does Speex fit into that?

Currently (2003/03/09), I'm doing a Ph.D. at the University of Sherbrooke in mobile robotics. Although I did my master with the Sherbrooke speech coding group (in speech enhancement, not coding), I am not associated with them anymore. It should **not** be understood that they or the University of Sherbrooke endorse the Speex project in any way. Furthermore, Speex does not make use of any code or proprietary technology developed in the Sherbrooke speech coding group.

CELP, ACELP, what's the difference?

CELP stands for "Code Excited Linear Prediction", while ACELP stands for "*Algebraic* Code Excited Linear Prediction". That means ACELP is a CELP technique that uses an algebraic codebook represented as a sum of unit pulses, thus making the codebook search much more efficient. This technique was invented at the University of Sherbrooke and is now one of the most widely used form of CELP. Unfortunately, since it is patented, it cannot be used in Speex.

B Sample code

This section shows sample code for encoding and decoding speech using the Speex API. The commands can be used to encode and decode a file by calling:
`% sampleenc in_file.sw | sampledec out_file.sw`
where both files are raw (no header) files encoded at 16 bits per sample (in the machine natural endianness).

B.1 sampleenc.c

sampleenc takes a raw 16 bits/sample file, encodes it and outputs a Speex stream to stdout. Note that the packing used is NOT compatible with that of speexenc/speexdec.

```
#include <speex.h>
#include <stdio.h>

/*The frame size in hardcoded for this sample code but it doesn't have to be*/
#define FRAME_SIZE 160
int main(int argc, char **argv)
{
   char *inFile;
   FILE *fin;
   short in[FRAME_SIZE];
   float input[FRAME_SIZE];
   char cbits[200];
   int nbBytes;
   /*Holds the state of the encoder*/
   void *state;
   /*Holds bits so they can be read and written to by the Speex routines*/
   SpeexBits bits;
   int i, tmp;

   /*Create a new encoder state in narrowband mode*/
   state = speex_encoder_init(&speex_nb_mode);

   /*Set the quality to 8 (15 kbps)*/
   tmp=8;
   speex_encoder_ctl(state, SPEEX_SET_QUALITY, &tmp);

   inFile = argv[1];
   fin = fopen(inFile, "r");

   /*Initialization of the structure that holds the bits*/
   speex_bits_init(&bits);
   while (1)
   {
```

```
      /*Read a 16 bits/sample audio frame*/
      fread(in, sizeof(short), FRAME_SIZE, fin);
      if (feof(fin))
         break;
      /*Copy the 16 bits values to float so Speex can work on them*/
      for (i=0;i<FRAME_SIZE;i++)
         input[i]=in[i];

      /*Flush all the bits in the struct so we can encode a new frame*/
      speex_bits_reset(&bits);

      /*Encode the frame*/
      speex_encode(state, input, &bits);
      /*Copy the bits to an array of char that can be written*/
      nbBytes = speex_bits_write(&bits, cbits, 200);

      /*Write the size of the frame first. This is what sampledec expects but
       it's likely to be different in your own application*/
      fwrite(&nbBytes, sizeof(int), 1, stdout);
      /*Write the compressed data*/
      fwrite(cbits, 1, nbBytes, stdout);

   }

   /*Destroy the encoder state*/
   speex_encoder_destroy(state);
   /*Destroy the bit-packing struct*/
   speex_bits_destroy(&bits);
   fclose(fin);
   return 0;
}
```

B.2 sampledec.c

sampledec reads a Speex stream from stdin, decodes it and outputs it to a raw 16 bits/sample file. Note that the packing used is NOT compatible with that of speexenc/speexdec.

```
#include <speex.h>
#include <stdio.h>

/*The frame size in hardcoded for this sample code but it doesn't have to be*/
#define FRAME_SIZE 160
int main(int argc, char **argv)
{
   char *outFile;
```

```
FILE *fout;
/*Holds the audio that will be written to file (16 bits per sample)*/
short out[FRAME_SIZE];
/*Speex handle samples as float, so we need an array of floats*/
float output[FRAME_SIZE];
char cbits[200];
int nbBytes;
/*Holds the state of the decoder*/
void *state;
/*Holds bits so they can be read and written to by the Speex routines*/
SpeexBits bits;
int i, tmp;

/*Create a new decoder state in narrowband mode*/
state = speex_decoder_init(&speex_nb_mode);

/*Set the perceptual enhancement on*/
tmp=1;
speex_decoder_ctl(state, SPEEX_SET_ENH, &tmp);

outFile = argv[1];
fout = fopen(outFile, "w");

/*Initialization of the structure that holds the bits*/
speex_bits_init(&bits);
while (1)
{
   /*Read the size encoded by sampleenc, this part will likely be
     different in your application*/
   fread(&nbBytes, sizeof(int), 1, stdin);
   fprintf (stderr, "nbBytes: %d\n", nbBytes);
   if (feof(stdin))
      break;

   /*Read the "packet" encoded by sampleenc*/
   fread(cbits, 1, nbBytes, stdin);
   /*Copy the data into the bit-stream struct*/
   speex_bits_read_from(&bits, cbits, nbBytes);

   /*Decode the data*/
   speex_decode(state, &bits, output);

   /*Copy from float to short (16 bits) for output*/
   for (i=0;i<FRAME_SIZE;i++)
      out[i]=output[i];
```

```
      /*Write the decoded audio to file*/
      fwrite(out, sizeof(short), FRAME_SIZE, fout);
   }

   /*Destroy the decoder state*/
   speex_encoder_destroy(state);
   /*Destroy the bit-stream truct*/
   speex_bits_destroy(&bits);
   fclose(fout);
   return 0;
}
```

C IETF RTP Profile

```
Internet Engineering Task Force                           Greg Herlein
Internet Draft                                         Jean-Marc Valin
draft-herlein-speex-rtp-profile-00                        Simon Morlat
February, 2002
Expires: July, 2003


             RTP Payload Format for the Speex Codec

Status of this Memo

   This document is an Internet-Draft and is in full conformance with
   all provisions of Section 10 of RFC2026.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups.  Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   Internet-Drafts are draft documents valid for a maximum of six
   months and may be updated, replaced, or obsoleted by other
   documents at any time.  It is inappropriate to use Internet-Drafts
   as reference material or to cite them other than as "work in
   progress".

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt

   To view the list Internet-Draft Shadow Directories, see
   http://www.ietf.org/shadow.html.

Copyright Notice

   Copyright (C) The Internet Society (2002).  All Rights Reserved.


Abstract
```

```
Speex is an open-source, patent-free voice codec suitable for use in
Voice over IP (VoIP) type applications.  The Speex codec supports three
modes of operation: narrowband at a nominal 8kHz sample rate,
wideband at a nominal 16kHz sample rate, and ultra-wideband at a
nominal 32kHz sample rate.  Speex supports Voice Activity Detection
(VAD)  and Variable Bit Rate (VBR).  This document describes the
payload format for Speex generated bit streams within an RTP packet.
Also included here are the necessary details for the use
of Speex with the Session Description Protocol (SDP) [4] and a preliminary
method of using Speex within H.323 applications.  Use of Speex with
MIME will be covered as part of the Ogg Vorbis MIME definitions and is
covered only minimally here.

Herlein, Valin, etc                                          [Page 1]
^L
Internet-Draft    RTP Payload Format for the Speex Codec    Nov 2002


1. Conventions used in this document

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC-2119 [5].

2. Overview of the Speex Codec

Speex is based on the CELP encoding technique with support for either
narrowband (nominal 8kHz), wideband (nominal 16kHz) or ultra-wideband
(nominal 32kHz) sampling.  The main characteristics can be summerized
as follows:

   o  Free software/open-source, royalty-free
   o  Integration of wideband and narrowband in the same bit-stream
   o  Wide range of bit-rates available
   o  Dynamic bit-rate switching and variable bit-rate (VBR)
   o  Voice Activity Detection (VAD, integrated with VBR)
   o  Variable complexity


3. RTP payload format for Speex

   Speex uses 20 ms frames and a variable sampling rate clock.  The
   RTP timestamp MUST be in units of 1/X of a second where X is the
   sample rate used.  Speex uses a nominal 8kHz sampling rate for
   narrowband use, a nominal 16kHz sampling rate for wideband use, and
   a nominal 32kHz sampling rate for ultra-wideband use.
```

The RTP payload for Speex has the format shown in Figure 1. No additional header specific to this payload format is required.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      RTP Header [2]                           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                                                               |
+                 one or more frames of Speex                   |
|                             ....                            |p|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: RTP payload for Speex

The encoding and decoding algorithm can change the bit rate at any 20ms frame boundary but the bit rate change notification is provided in-band with the bit stream. Each frame contains both "mode" (narrowband,wideband or ultra-wideband) and "sub-mode" (bit-rate) information in the bit stream. No out-of-band notification is required for the decoder to process changes in the bit rate sent by the encoder.

^L

For the purposes of packetizing the bit stream in RTP, it is only necessary to consider the sequence of bits as output by the Speex encoder, and present the same sequence to the decoder. The payload format described here maintains this sequence.

An RTP packet MAY contain Speex frames of the same bit rate or of varying bit rates, since the bit-rate for a frame is conveyed in band with the signal.

It is RECOMMENDED that values of 8000 or 16000 be used for normal internet telephony applications, though the sample rate is supported at rates as low as 6000 Hz and as high as 32 kHz.

The RTP payload MUST be padded to provide an integer number of octets as the payload length. These padding bits MUST be all zero. This padding is only required for the last frame in the packet, and only to ensure the packet contents ends on an octet boundary.

```
3.1 RTP Payload Type Codes

   The RTP Audio-Visual Working Group will no longer issue static
   payload type codes for RTP (beyond those already assigned).
   Dynamic payload type codes MUST be negotiated 'out-of-band'
   for the assignment of a dynamic payload type from the
   range of 96-127.  Examples of this are shown in the section
   discussing the Session Description Protocol (SDP) below.


3.2 Multiple Speex frames in a RTP packet

   By default only one Speex frame is permitted in a single RTP
   packet.  When operating with multiple frames per packet then the
   end points MUST use out-of-band negotiation to determine the number
   of frames per packet.  See section 5 below for an example of how to
   do this with SDP [4].


3.3 Computing the number of Speex frames

   If using SDP [4] (see section 5 below for an example) this can be
   done using the "ptime" variable to denote the packetization
   interval (ie, how many milliseconds of audio is encoded in a single
   RTP packet).  Since Speex uses 20ms frames, ptime values of
   multiples of 20 denote multiple Speex frames per packet.  Values of
   ptime in other than multiples of 20 SHOULD be ignored and SHOULD
   use the default value of one instead.


4. MIME registration of Speex

   Full definition of the MIME type for Speex will be part of the Ogg
   Vorbis MIME type definition application.

   MIME media type name: audio

   MIME subtype: speex

   Required parameters: to be included in the Ogg MIME specification.

Herlein, Valin, etc                                         [Page 3]
^L
Internet-Draft    RTP Payload Format for the Speex Codec    Nov 2002
```

```
   Optional parameters:

   Encoding considerations:

   Security Considerations:
         See Section 6 of RFC 3047.

   Interoperability considerations: none

   Published specification:

   Applications which use this media type:

   Additional information: none

   Person & email address to contact for further information:
 Greg Herlein <gherlein@herlein.com>
 Jean-Marc Valin <jean-marc.valin@hermes.usherb.ca>

   Intended usage: COMMON

   Author/Change controller:
         Author:  Greg Herlein <gherlein@herlein.com>
         Change controller: Greg Herlein <gherlein@herlein.com>


Herlein, Valin, etc                                     [Page 4]
^L
Internet-Draft    RTP Payload Format for the Speex Codec    Nov 2002


5. SDP usage of Speex

   When conveying information by SDP [4], the encoding name SHALL be
   "speex".  An example of the media representation in SDP for
   offering a single channel of Speex at 8000 samples per second might
   be:

m=audio 8088 RTP/AVP 97
a=rtpmap:97 speex/8000

   Note that the RTP payload type code of 97 is defined in this media
   definition to be 'mapped' to the speex codec at an 8kHz sampling
   frequency using the 'a=rtpmap' line.  Any number from 96 to 127
```

could have been chosen (the allowed range for dynamic types). The value of the sampling frequency is typically 8000 for narrow band operation, 16000 for wide band operation, and 32000 for ultra-wide band operation.

If for some reason the offerer has bandwith limitations, he may use the "b=" header, as explained in SDP [4]. The following example illustrates the case where the offerer cannot receive more than 10 kbit/s.

```
    m=audio 8088 RTP/AVP 97
b=AS:10
a=rtmap:97 speex/8000
```

In this case, if the remote part agrees, it should configure its speex encoder so that it does not use modes that produce more than 10 kbit/s. Note that the "b=" constraint also applies on all payload types that may be proposed in the media line ("m=").

An other way to make recommendations to the remote speex encoder is to use its specific parameters via the a=fmtp: directive. The following parameters are defined for use in this way:

ptime: duration of each packet in milliseconds.

sr: actual sample rate in Hz.

ebw: encoding bandwidth - either 'narrow' or 'wide' or 'ultra' (corresponds to nominal 8000, 16000, and 32000 Hz sampling rates).

vbr: variable bit rate - either 'on' 'off' or 'vad' (defaults to off). If on, variable bit rate is enabled. If off, disabled. If set to 'vad' then constant bit rate is used but silence will be encoded with special short frames to indicate a lack of voice for that period.

cng: comfort noise generation - either 'on' or 'off'. If off then silence frames will be silent; if 'on' then those frames will be filled with comfort noise.

mode: speex encoding mode. Can be {1,2,3,4,5,6,any} defaults to 3 in narrowband, 6 in wide and ultra-wide.

penh: use of perceptual enhancement. 1 indicates

```
  to the decoder that perceptual enhancement is recommended,
0 indicates that it is not. Defaults to on (1).


Herlein, Valin, etc                                     [Page 5]
^L
Internet-Draft    RTP Payload Format for the Speex Codec    Nov 2002


   Examples:

    m=audio 8008 RTP/AVP 97
a=rtpmap:97 speex/8000
a=fmtp:97 mode=4

   This examples illustrate an offerer that wishes to receive
   a speex stream at 8000Hz, but only using speex mode 3.

   The offerer may suggest to the remote decoder to activate
   its perceptual enhancement filter like this:

m=audio 8088 RTP/AVP 97
a=rtmap:97 speex/8000
a=fmtp:97 penh=1

   Several speex specific parameters can be given in a single
   a=fmtp line provided that they are separated by a semi-colon:

    a=fmtp:97 mode=any;penh=1

   The offerer may indicate that it wishes to send variable bit rate
   frames with comfort noise:

m=audio 8088 RTP/AVP 97
a=rtmap:97 speex/8000
a=fmtp:97 vbr=on;cng=on


   The use of a particular packetization interval may be
   suggested to the remote encoder using the ptime parameter:

    m=audio 8008 RTP/AVP 97
a=rtpmap:97 speex/8000
a=ptime:40

   Note that the ptime parameter applies to all payloads listed
```

```
   in the media line and is not used as part of an a=fmtp directive.

   Speex can encode frames of 20 ms. Values of ptime not multiple
   of 20 ms are meaningless, so the receiver of such ptime values
   SHOULD ignore them.


Herlein, Valin, etc                                         [Page 6]
^L
Internet-Draft    RTP Payload Format for the Speex Codec    Nov 2002

6. ITU H.323/H.245 Use of Speex

   Application is underway to make Speex a standard ITU codec.
   However, until that is finalized, Speex MAY be used in H.323 [6] by
   using a non-standard codec block definition in the H.245 [7] codec
   capability negotiations.


6.1  NonStandardMessage format

   For Speex use in H.245 [7] based systems, the fields in the
   NonStandardMessage should be:

   t35CountryCode   = Hex: B5
   t35Extension     = Hex: 00
   manufacturerCode = Hex: 0026
   [Length of the Binary Sequence (8 bit number)]
   [Binary Sequence consisting of an ASCII string, no NULL terminator]

   The binary sequence is an ascii string merely for ease of use.
   The string is not null terminated.  The format of this string is

       speex [optional variables]

   The optional variables are identical to those used for the SDP
   a=fmtp strings discussed in section 5 above.  The string is built
   to be all on one line, each key-value pair seperated by a
   semi-colon.  The optional variables MAY be ommited, which causes
   the default values to be assumed.  They are:

       ebw=narrow;mode=3;vbr=off;cng=off;ptime=20;sr=8000;penh=no;

   The fifth byte of the block is the length of the binary sequence.

   NOTE:  this method can result in the advertising of a large number
```

of Speex 'codecs' based on the number of variables possible. For most VoIP applications, use of the defailt binary sequence of 'speex' is RECOMMENDED to be used in addition to all other options. This maximizes the chances that two H.323 based applications that support Speex can find a mutual codec.

6.2 RTP Payload Types

Dynamic payload type codes MUST be negotiated 'out-of-band' for the assignment of a dynamic payload type from the range of 96-127. H.323 applications MUST use the H.245 H2250LogicalChannelParameters encoding to accomplish this.

7. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [2], and any appropriate RTP profile. This implies that confidentiality of the media streams is achieved by encryption. Because the data compression used with this payload format is applied end-to-end, encryption may be performed after compression so there is no conflict between the two operations.

A potential denial-of-service threat exists for data encodings using compression techniques that have non-uniform receiver-end computational load. The attacker can inject pathological datagrams into the stream which are complex to decode and cause the receiver to be overloaded. However, this encoding does not exhibit any significant non-uniformity.

As with any IP-based protocol, in some circumstances a receiver may be overloaded simply by the receipt of too many packets, either desired or undesired. Network-layer authentication may be used to discard packets from undesired sources, but the processing cost of the authentication itself may be too high.

8. References

1. Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

2. Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for real-time applications", RFC 1889, January 1996. (Updated by a Work in Progress.)

3. Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

4. Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

5. Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6. ITU-T Recommendation H.323. "Packet-based Multimedia Communications Systems," 1998.

7. ITU-T Recommendation H.245 (1998), "Control of communications between Visual Telephone Systems and Terminal Equipment".

9. Acknowledgments

The authors would like to thank Equivalence Pty Ltd of Australia for their assistance in attempting to standardize the use of Speex in H.323 applications, and for implementing Speex in their open source OpenH323 stack. The authors would also like to thank Brian C. Wiles <brian@streamcomm.com> of StreamComm for his assistance in developing the proposed standard for Speex use in H.323 applications.

10. Author’s Address

Greg Herlein <gherlein@herlein.com>
2034 Filbert Street
San Francisco, CA
United States 94123

Jean-Marc Valin <jean-marc.valin@hermes.usherb.ca>
Department of electrical and computer engineering
University of Sherbrooke
2500 blvd Université
Sherbrooke, Quebec, Canada, J1K 2R1

Simon MORLAT <simon.morlat@linphone.org>
35, av de Vizille App 42
38000 GRENOBLE

FRANCE

Roger Hardiman <roger@freebsd.org>
49 Nettleton Road
Cheltenham
Gloucestershire
GL51 6NR
England

^L

10. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the

Internet Society.

D GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- H. Include an unaltered copy of this License.

- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another;

but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Index