Network Up Demo Design High Level Notes

These notes accompany NetUpBlock1.jpg

Files M_1, M_2, …, M_N contain pre-encoded speech in N different modes. N=3 is a good value for prototype. It might grow but N will never exceed 10. The same speech is in each file. Mode 1 has the highest encoding rate and thus will be the largest file. The rate of the encodings, and thus the file sizes decreases for each subsequent higher mode. No mode file will exceed 10 MB in size.

Packetizer places a fixed number (B) of bytes into each UDP packet. B will not exceed 1600 and will probably be 320 in our prototype. The packetizer is responsible for adding the redundancy that is described in the "packet contents" section of these notes.

The file that feeds the packetizer is determined by the N position mode selection switch. A pointer must keep track of the "playout position" so that when the switch changes position the proper bytes are taken from the new file so that the speech is continuous, not restarted.

The packet deletion mechanism removes UDP packets according to the rate and distribution set by the user. The goal is that these can be changed on the fly and we can hear the results through the speaker in real time. Rate is simply percent of packets deleted. Distribution is a number that tells how these are distributed in time. It allows deletions to be independent in time (like coin flips) or to occur in bursts, runs, or "clumped together."

The packet loss rate module calculates just that. It will generate a new local average value every W seconds. W will be in the range 0.5 to 5.

The Mode LUT converts each of the loss rate values to one of N modes. Note that the LUT could equally well reside in the TX laptop or the RX laptop. The Mode LUT output drives the mode selection switch in the TX laptop. The trend is that higher packet loss causes higher modes to be selected. The thresholds and W are to be determined. Hysteresis will be required to prevent excessive mode switching. So 2*(N-1) thresholds will be required

The speech decoder decodes received packets and takes special actions to account for the packets not received. It passes buffers of sound samples to the sound player.

The sound player receives buffers of samples from the speech decoder. The sample rate at the input to the sound player may be 8000 or 16,000 smp/sec. The bit depth at the input to the sound player is 16 bits/smp.

Network Up Demo Packet Contents Notes

The differences between the modes are the speech coding rate and the amount of redundancy per UDP packet. These are selected so that the UDP packet size (and hence the transmitted data rate) remain constant independent of mode. Mode 1 will always contain the highest speech coding rate and the least redundancy. Higher numbered modes will provide lower speech coding rate but higher redundancy.

PacketContents.jpg shows an example and we will use this example for our prototype. Mode 1 uses speech coding with 8 bits/smp. Each encoded frame of speech is transmitted only once and that gives no redundancy. Mode 3 uses only 2 bits/smp for speech coding. And Mode 3 also has quadruple redundancy (each encoded frame of speech is sent in four different UPD packets). Mode 2 in the example is intermediate in terms of speech coding and redundancy.

Higher speech coding rate produces higher baseline speech quality in the absence of lost UDP packets. But higher redundancy prevents outages when packets are lost. This means that the best mode to use will depend on the current channel conditions (packet loss rate).

The files Mode1.wav, Mode2.wav, and Mode3.wav provide examples of how the 3 different modes sound when no packets are lost. Each contains 4,142,984 samples. Sample rate is 16,000 smp/sec. So duration is 258.9 sec or 4.3 min.

Network Up Demo Speech Decoding and Playback Notes

We should include a mode indicator in each UPD payload to aid in decoding. In addition we should include two bytes that give the number of the first frame in the payload (or that could probably be calculated from the UDP packet sequence numbers if preferred).

Using the example in PacketContents.jpg, here is one example of how decoding and playback might work.

> Place zeros in the input buffer of sound player (SPIB). The sound player will have to play zeros until a packet is received.
>
> When a packet is received, decode any frames that have not already been placed in SPIB and send them to SPIB.
>
> If SPIB does not contain a frame when it is time to play it, place a frame of zeros in SPIB.

This is a simple approach and thus good for getting the basic framework in place getting speech flowing out the speaker. If time and resources permit, we can add packet loss concealment. This is an algorithm that lets us put something better than zeros into SPIB when no decoded frame is available. This algorithm uses a history of previously decoded frames to generate a plausible extrapolation for a missing frame or frames. This can be effective for perhaps 50 to 70 ms, but then it will have to revert to zeros because there is no way to successfully extrapolate speech beyond that point.

In terms of speech coding and decoding we will start very simple to get proof of concept. The goal is to get the entire system up and running, and to be able to demonstrate the concept. If time and resources permit, and if the code is modular, we can replace the speech decoder with a more sophisticated one. That could be a hunk of code from an external source, or it could be written specifically for this project. If we get to that point we can consider the options.

So for a starting point we will use G.711 and a sample rate of 16,000 smp/sec. It is not state-of-the-art but it is simple and it will demonstrate what we need to demonstrate. We will also use sample rate conversion. Mode 1 will be wideband and modes 2 and 3 will be narrowband. The very simple approach specified below is not a best practice in signal processing, but it is simple and will demonstrate the point.

How to decode mode1.chn: Each byte produces one audio sample. Thus the 4,142,984 bytes in this file will produce 4,142,984 audio samples.

In Matlab, the bytes are read thus:

```
y = fread(fid,'uint8'); %range is 0 to 255
```

And they are decoded to speech samples thus:

```
gm = 128;
u = 255;

y=y-gm; %range is now -128 to +127

y(0<=y) = y(0<=y) + 1; %shift 0 and positive values up one
%range is now -128 to -1 and +1 to +128

s = sign(y); %extract sign

y = abs(y);      %extract magnitude

x = (exp( (y/gm) * log(1+u) )-1)/u;
%calculate the magnitudes of the speech samples
%(in Matlab "log" is the natural log)

x = x.*s; %apply the proper sign
```

The values in x are audio sample values for SPIB.

How to decode mode2.chn: Each byte produces two samples. Thus the 2,071,492 bytes in this file will produce 4,142,984 audio samples.

Each byte is decoded to produce a sample in the same manner as for mode 1. For example byte 1 produces sample 1 and byte 2 produces sample 3. Sample 2 is found by simple linear interpolation: x_2=(x_1+x_3)/2.

Repeat to end of frame. Byte 160 produces sample 319. Sample 320 can be set to zero since there is no sample 321 to enable interpolation.

How to decode mode3.chn: Each byte produces four samples. Thus the 1,035,746 bytes in this file will produce 4,142,984 audio samples. For example byte 1 produces samples 1 and 3. Then linear interpolation is used above to get samples 2 and 4. Here is how it goes in Matlab:

The bytes are read thus:

```
y = fread(fid,'uint8'); %range is 0 to 255
```

The bytes are separated into two integers based thus:

```
y1 = floor(y/16); %Samples 1, 5, 9, … will be calculated from the 4
msbs, range for y1 is 0 to 15

y2 = rem(y,16); %Samples 3, 7, 11, … will be calculated from the 4
lsbs, range for y2 is 0 to 15
```

Then integers are decoded to speech samples thus (we use y in place of y1 or y2 below)

```
gm = 8;
u = 255;

y = y-gm; %range is now -8 to +7

y(0<=y) = y(0<=y) + 1; %shift 0 and positive values up one
%range is now -8 to -1 and +1 to +8

s = sign(y); %extract sign

y=abs(y);  %extract magnitude

x= (exp( (y/gm) * log(1+u) )-1)/u;
%calculate the magnitudes of the speech samples
%(in Matlab "log" is the natural log)

x=x.*s; %apply the proper sign
```

This process will give the odd numbered samples. The even numbered samples are found by linear interpolation as given for decoding mode 2 above.

Global notes for all 3 modes:

The above will produce values in the nominal range +/- 1.0 . Depending on the properties of the SPIB, it may be necessary to scale these by 2^15 in order to get to the range of 16 bit signals (+/- 2^15).