The Process:

Audio and video is recorded live and converted into .avi files

\/

Add our “High-Frequency-Timecode” to the audio file

\/

Files are compressed and put onto streaming server

\/

Streaming server pushes files to client server

\/

Client media player decompresses files

\/

Client media player (?) analyzes audio file; matches the timecode of the video and audio track

\/

Client media player plays the aligned tracks

Big Picture Questions:

Why do we need a new timecode system? Aren’t there timecode systems in place (SMPTE & LTC)?

Wouldn’t it be easier/faster to add some non-audio metadata to the audio file server side, that could then be interpreted client side (prevents altering the raw audio)?

Detail Questions:

How exactly would a program like this work for streaming (sequences through chunks of audio)?

Would we need to “hack” a media player to do our audio timecode analysis before it plays any media?

Try:

Generate a wav file of 18kHz pulses every .5 seconds,

Overlay this file onto another wav file (music, pinknoise, etc.)

Filter this file using a very narrow passband filter (very steep rolloff, regardless of passband attenuation)

Take the fft of small chunks of this file (chunks: not containing any of the pulse, containing part of the pulse, containing the entire pulse) and see whether we can differentiate pulse vs. no pulse

Results:

can differentiate with great consistency. Taking an FFT of 3 ms is still large enough to distinguish a peak at 18kHz. Tested on segments of Beethoven’s 5th

Try:

How short can the pulses be (and therefore the fft) and still get distinguishable results?

Results

8 pulses per frame. Using Differential Manchester encoding, this allows 4 bits per frame

future step:

establish a timecode protocol.

IDEA FOR GENERATING TIMECODE:

read in an audio file of steady 18kHz tone at proper amplitude: -45dB

start at frame 00:00:00-00.

Encode current time & frame into binary

Turn the binary code into an array of 1’s and 0’s corresponding to changes in amplitude per sample