Ray Alfano
3/10/13
Network Computing

Data Requirements for Project and Highlights from Notes

Please be aware that my notes for this project are 13 pages long in addition to bookmarks in the textbooks I reference in my primary report. The most relevant data requirement notes are provided here as they are reflected in my program.

Data and throughput requirements, assuming FM radio is the worst quality we are willing to accept.

The worst-case file size given here is for uncompressed audio (WAV). More testing is required to find OPUS's requirements, which have been uniformly much better in limited tests.

Here are rough connection speeds that I will be using for testing.

Connection speed (roughly)	Audio bandwidth (KHz) and bit depth	Bit rate	Worst-case file size for 3 minute track
Very high-speed, unimpeded	96000kHz/24-bit (ultra-high quality)	4.39Mbit/s	99mb
High-speed, such as normal FiOS	48000kHz/16-bit (DVD-Audio)	1.46Mbit/s	33mb
High-speed, such as normal FiOS	44100kHz/16-bit (CD-audio)	1.35Mbit/s	30.3mb

More granularly researched requirements: All fields refer to stereo audio. It is possible to accept lower quality than these, but since a critical interest of mine is audio fidelity, FM radio is the bare minimum I am willing to accept at this point.

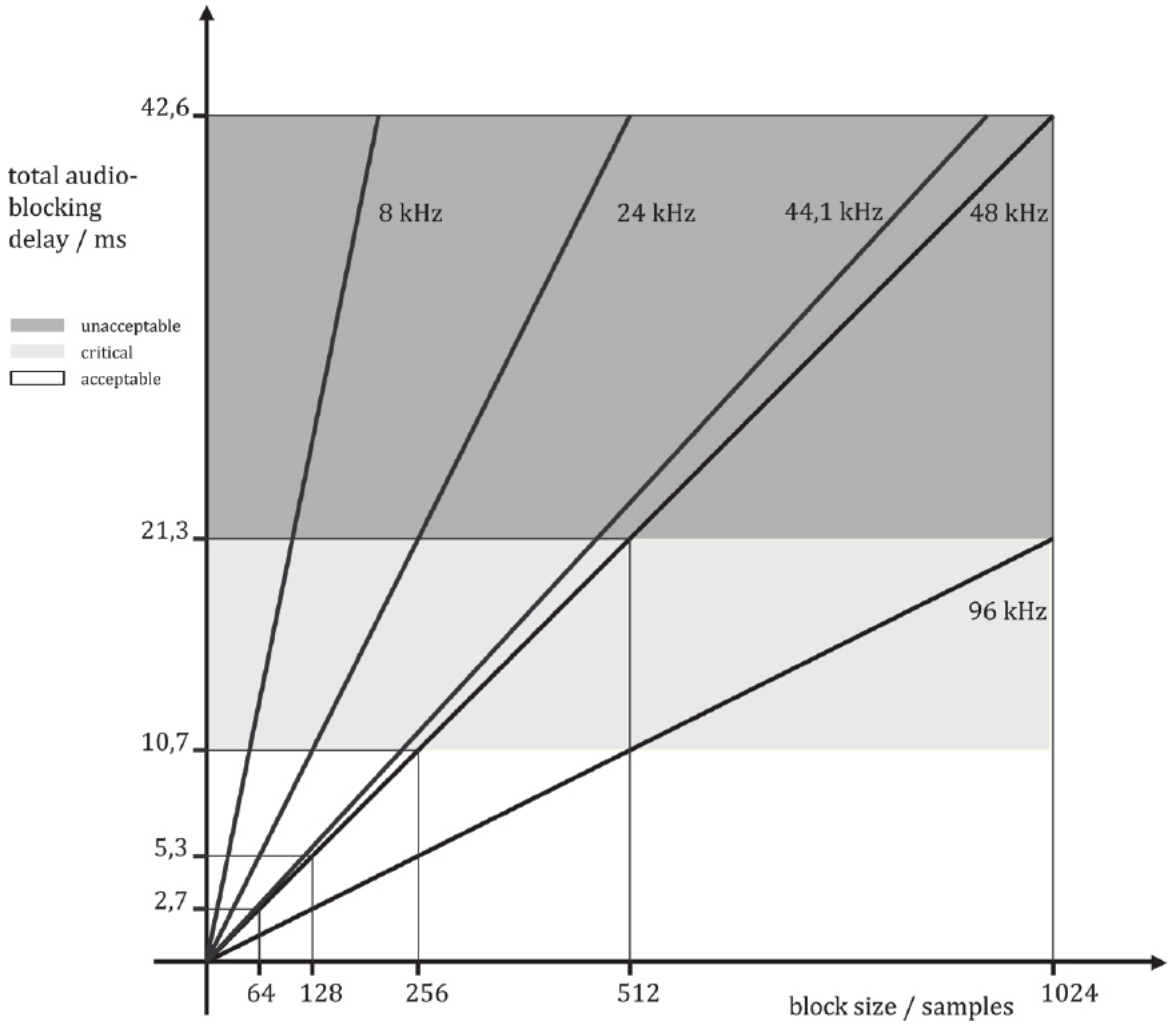
Connection speed (KBPS)	Approximate audio quality	Real bandwidth (kHz)
128-156 (observed)	CD	20-22kHz (depending on encoding)
96	Worst-case lossy CD audio	16-20kHz
64	FM radio (roughly)	12kHz (observed)
56 (minimum for my implementation)	Semi-lossy FM radio (observed)	12kHz

Equations necessary to my program:

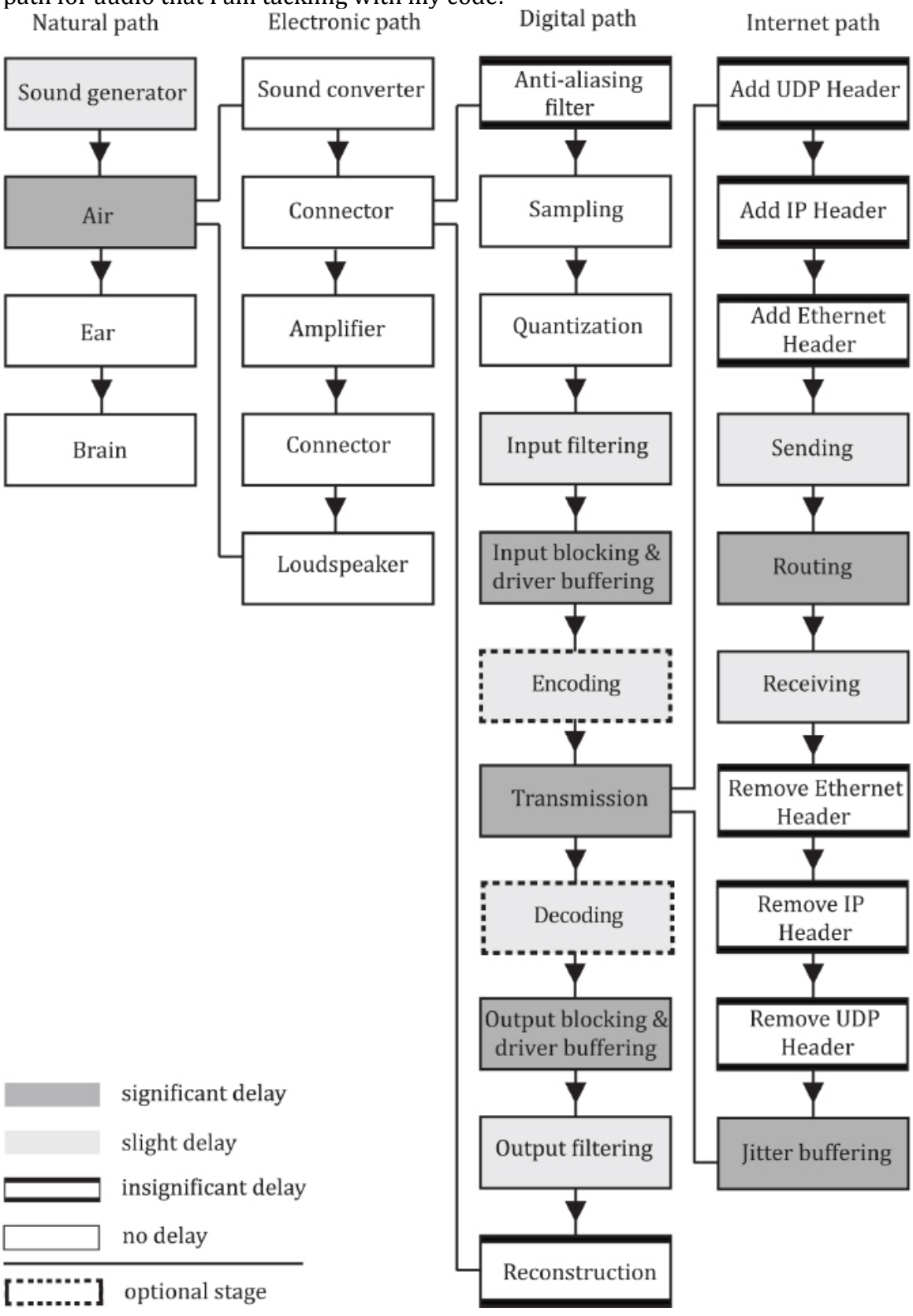
- Practically speaking, the time to process N samples must be constant to provide a consistent stream of audio. My program will implement throttling of bitrates over time, but at a given time: T(N)[processing] = t(N)[functioncall] + t[overhead]
- The baseline audio sample size in OPUS for a 16-bit audio stream is 2 bytes per sample, leading to audio blocks that are 64 bytes. In my tests so far, OPUS will reliably handle far greater block sizes, which corresponds to higher quality. The equation for the blocking delay is crucial: blocking delay = block size / sample rate
- The remainder of the equations is specific to certain situations and can be provided at length if requested. The complete sound path from end-user across a network to the next end-user is accounted for at this point.

I found the following diagrams to be helpful in modeling my research, they will appear in my final report.

Modeling latency by audio resolution, with 30ms being my worst-case latency at the transmission end:



Sound path as given by Zhu et al. This is an excellent explanation of the signal/sound path for audio that I am tackling with my code:



Additional constraints: From subjective experience, an audio latency of 30ms or less is the goal for my project, and throttling of bit rates will be implemented to keep close to that. At latencies greater than 30ms the “feel” of playing in real time is often lost. I am leaving out an explanation of the various theories for introducing delays as an artistic compromise, but several of them outlined by Zhu et al justify larger audio latencies. My goal is to provide low latency strictly real time performance, so 30ms will be my practical limit if possible. This will consequently implement the Latency-Accepting Approach to audio transmission, which is simply to say that it permits some small amount of asynchronicity that should not disrupt the “real-time” nature of the performance.

~~Portion that will be expounded upon further in report #2. MIDI signal representations as extremely low-latency audio transmissions.~~