Wireless Channel Simulator for Linux

Josef N. Peterson

***Abstract*—Computer simulations of wireless communications channels allow engineers to evaluate design trade-offs before the first bit of circuit design. While simulations can easily be created in MATLAB, open-source high level programming languages, like Python, allow engineers another path to simulate system characteristics. Python offers clear readable syntax, often similar to MATLAB, but at much lower cost and often at higher speed. The language contains complex data objects, like lists and arrays, numerical analysis packages, plotting capabilities, and freely available on-line documentation. It offers a path to graphical user interface (GUI) implementation, using the Gnome Tool Kit (GTK+) and Glade. Coupled with Linux open sound system (OSS) audio, an engineer can rapidly implement a complete wireless channel simulator, from microphone to speaker.**

I. Introduction

THIS proposal documents the initial design of the open-source wireless channel simulator. Some design decisions are implied in the title itself. The software will use open-source resources. Specifically, it will use the Python programming language and Linux open sound system. This proposal presents the details of the programming and computing resources, major tasks, and published references needed to implement the software.

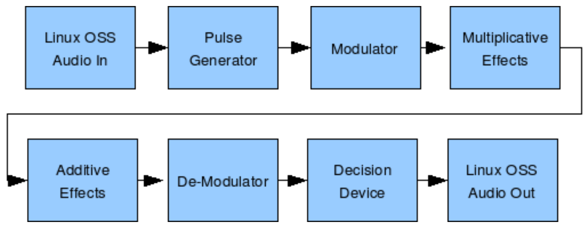

Figure 1: Wireless Channel Simulator Block Diagram

The block diagram for the wireless simulator is shown in Figure 1. It will be a simplex channel. The audio input device produces a µ-Law pulse code modulation (PCM) stream, and is discussed further in section II.A. The pulse generator and modulator will produce a stream of voltage or current values representing the modulated PCM. Multiplicative effects and additive effects will include small-scale fading and AWGN respectively. The de-modulator will be the optimal demodulator for the chosen method of modulation. The decision device will recover the PCM stream, which will then be delivered to the Linux audio system. The intent is for the system to operate in near real-time. The user will speak into a microphone and immediately hear his voice from the system speakers.

J. N. Peterson is a student at the University of Florida Research and Engineering Education Facility, Fort Walton Beach, FL 32542 (phone: 229-834-5704; e-mail: joecool2002@ gmail.com).

II. Major Development Tasks

A. Convert PCM Audio Signal to Baseband Signal

8-bit µ-law PCM symbols arrive from the OSS audio device at a sampling rate f_s. The sampling rate can be set using the OSS python module ossaudiodev. The python module returns a string with an 8-bit value. The strings must then be converted to a series of baseband pulses before they can be modulated to a pass-band signal. The number of samples over T_b of the baseband pulses depends on the f_{max} of the anticipated carrier signal, to satisfy the Nyquist sampling rate. In this project a standard 10% oversampling will be used. Estimated completion date: March 20.

B. Modulate Baseband Signal to Carrier Frequencies

The baseband pulses must be converted to pass-band before they can be passed through the channel. For this project, three modulations will be used: BPSK, QPSK, and MSK. The signal waveforms are described mathematically in Proakis' *Digital Communications* text. Estimated completion date: March 25.

C. Develop Linear Time Variant (LTV) Channel

The channel will include three distortions: small-scale fading, large-scale path loss and additive white Gaussian noise (AWGN). The effects are described mathematically in Proakis' *Digital Communications* text and Rappaport's *Wireless Communications* text. Estimated completion date: April 5.

D. Develop Signal Demodulator

The demodulator will be paired with the corresponding modulated signal. The demodulators will be based on the ideal Correlation Demodulator described in the Proakis text, page 233. The output of the demodulator will be a stream of decision statistics. Estimated completion date: April 10.

E. Develop Decision Device Simulator

The decision device will be a simple maximum likelihood estimator. The output of the decision device will be a recovery of the bit stream developed from the OSS audio device. Estimated completion date: April 12.

F. Develop Output Code to Linux OSS Audio Device

The recovered bit stream must be converted into 8-bit symbols. The symbols must then be sent to the OSS Audio device at a rate greater than or equal to the sampling rate to prevent buffer under-runs and choppy playback. Estimated completion date: April 16.

G. Create GUI to Adjust Modulation and Channel Variables

Each component of the simulator will have variables that the user may wish to adjust to change the performance of the system. While this can be accomplished in the code itself, it would be most beneficial if the user could change the variables in real-time. The PyGTK module allows a programmer to create windows, add widgets, and connect those widgets to pieces of code. The planned adjustment points for the GUI are: type of modulation, carrier/center frequency, fading (on/off), gain(s), time delay(s) (based on simulator, p. 223 in Rappaport text), AWGN (on/off) and propagation distance. Estimated completion date: April 21.

III. Final Report

The final report will include a detailed description of each component of the software system, and pertinent mathematics. The report section will be completed along with the implementation of each component. Estimated completion date: 28 April.

References

[1] B. Porat. *A Course in Digital Signal Processing*. New York, NY: John Wiley & Sons, Inc., 1997.

[2] T. S. Rappaport. *Wireless Communications: Principles and Practices, 2nd Edition.* Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[3] J. G. Proakis. *Digital Communications, 4th Edition*. Boston, MA: McGraw-Hill, 2001.

[4] F. G. Stremler. *Introduction to Communications Systems, 3rd Edition*. Reading, MA: Addison-Wesley, 1990.

[5] H. Stark and J. W. Woods. *Probability and Random Processes with Applications to Signal Processing, 3rd Edition.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[6] Pilgrim, Mike. *Dive into Python*. http://diveintopython.org, 2004.

[7] Anonymous. *μ-law Algorithm*. http://www.wikipedia.org, 2008.

[8] Anonymous. *Open Sound System – Audio Programming.* http://www.4front-tech.com/pguide/audio.html, 2008.

[9] G. van Rossum. Python Library Reference. http://python.org/doc/2.5/lib/lib.html, 2006.

[10] PyGTK. http://www.pygtk.org, 2008.

[11] NumPy, http://numpy.scipy.org, 2008.

[12] SciPy, http://www.scipy.org, 2008.

[13] Matplotlib, http://matplotlib.sourceforge.net, 2008.