Literature Survey

on

HDSL2 Modem Modeling and Simulation

Patrick Jackson

Reza Koohrangpour

March 23, 1999

EE 382C: Embedded Software Systems

Spring 1999

1 Introduction

Our project models and simulates HDSL2, a high-bit-rate digital subscriber line modem. HDSL2 provides a full-duplex payload data rate of 1.544 Mbps over a single copper pair. The modem promises to overcome some strict limitations of the T1 protocol by using state-of-the-art digital communications techniques made practical through recent advances in digital signal processor technology. The older T1 protocol provides an equal data rate, but has some significant drawbacks. T1 requires two copper pairs that are free of bridge-taps, must use repeaters to travel long distances, and is not spectrally compatible with other services that may be running on pairs in close proximity [1]. We plan to develop our HDSL2 model with the Ptolemy development environment, following the specifications described in the HDSL2 draft standard [2] and an implementation proposal written in support of the draft [3].

The primary objective of this survey is to describe the end-to-end operation of an HDSL2 modem. The second objective is to give a summary of our plans for modeling and simulation.

2 Modem Operation

2.1 Startup Mode

As the draft describes, the modem operates in two modes, startup mode and data mode. While in startup mode, the transmitter sends a predefined sequence of data to allow the receiver's adaptive equalizer to train on the channel. Figure 1 provides a diagram of the components that comprise a startup mode session. We will begin with a description of the transmitter, followed by the receiver.

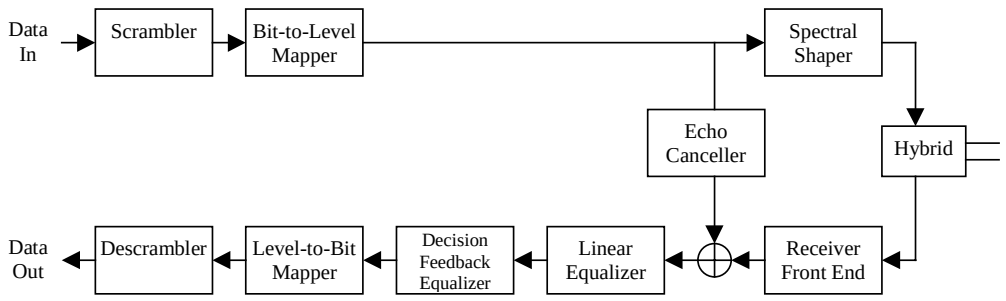

Figure 1 Startup Mode of an HDSL2 Modem

The first component to process the training sequence is the scrambler. The scrambler acts to "whiten" the data to be sent. For modem components such as the echo canceller and the equalizer to function properly, an assumption has to be made that the data is random and IID (independent and identically distributed) [4]. This assumption can be easily violated since long sequences of zeros or ones may be sent. The scrambler tries to ensure this pretext by making the bit sequence look random.

In the next module, the bit-to-level mapper converts the bit sequence to the appropriate output level. During startup mode, a simple two-level pulse amplitude modulation (PAM) scheme is used. Each bit input to the mapper will result in an output level of –9/16 or 9/16 for zero and one, respectively. This results in a symbol rate equal to the bit rate. The mapper passes these symbol pulses to the shaper. The shaper performs the filtering on the symbol sequence needed to produce a continuous-time signal for transmission over the channel. This component must ensure that the modem meets the strict spectral compatibility requirements previously mentioned [5].

In order to perform full-duplex communication over a single wire pair, the signals from the transmitter and receiver must be combined with a hybrid. This hybrid is the transmitter's primary source of echo. Due to impedance mismatches between the hybrid

and the wire pair, an attenuated and distorted version of the transmitted signal is reflected back to the receiver. The echo canceller is an adaptive transversal filter that learns the response of the hybrid and generates a replica of the reflected signal to be subtracted from the received waveform. The receiver front end performs timing recovery and sampling [4].

The linear equalizer and the decision feedback equalizer (DFE) compose the system that processes the incoming signal to reverse the linear amplitude and phase distortion caused by the channel. This distortion, is known as inter-symbol interference (ISI), causes adjacent symbols to overlap and interfere with one another. The structure of the system is shown in Figure 2.

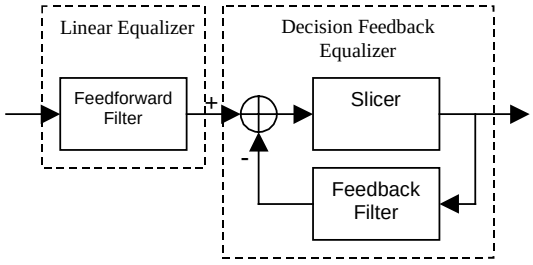

Figure 2 Startup Mode Equalization System

The linear equalizer is an adaptive feedforward filter that provides two functions. It suppresses channel precursors (ISI generated by samples to be received in the future), and it helps to whiten the noise added by the channel. The DFE is made up of a slicer and an adaptive feedback filter. The slicer quantizes the filtered signal, providing an estimate of the symbol received. With this, and past estimates, the feedback filter cancels the channel postcursors (ISI generated by previous samples) [6][7]. The filters are jointly adapted during startup mode while the training sequence is being processed. Briefly, the adaptation algorithm uses the error signal given by the difference between the resulting

sequence and the known training sequence to adjust the filter coefficients. The adjustment is made in the direction that will more accurately produce future symbols.

The equalized pulse samples are fed to the level-to-bit mapper. This component reconstructs the bit stream by quantizing the samples from the PAM levels received. Finally, the desrambler simply acts to return the bit stream to its prescrambled form, producing the received data to output.

After training, the receiver sends configuration data back to the transmitter. This configuration data consists of precoder coefficients found by the decision feedback equalizer and the convolutional encoder parameters that the receiver expects the transmitter to use. We give a description of these components in the following section. The modem then switches to data mode.

2.2 Data Mode

Figure 3 shows the components that make up the data mode session.

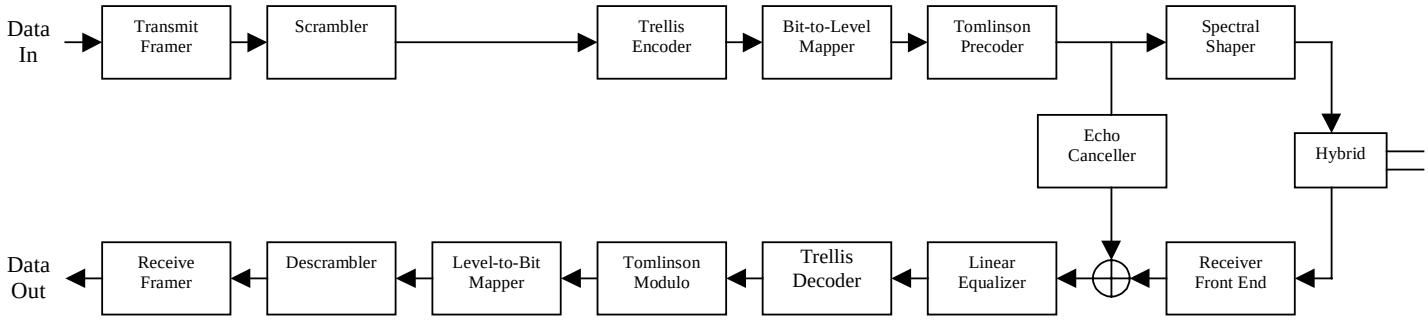

Figure 3 Data Mode of an HDSL2 Modem

The transmit framer encapsulates DS-1 formatted payloads to send as HDSL2 frames [2]. The scrambler processes these frames as described for startup mode. The scrambled frames are passed to the trellis encoder, whose structure is shown in Figure 4. The incoming bit stream is first converted to a sequence of 3-bit parallel words. The least

significant bit (earliest in time) of each word is given to the convolutional encoder, while the remaining two are passed through unaltered. The convolutional encoder produces two bits for every one it is given, resulting in a 4-bit word. The 4-bit encoded symbols are sent to the mapper to be converted to one of 16 PAM levels, equally spaced between –15/16 and 15/16.

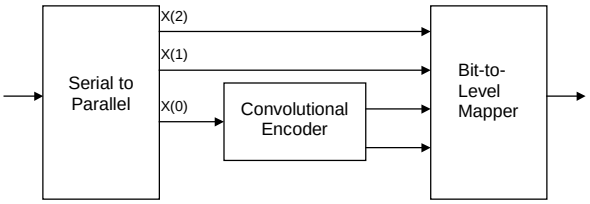

Figure 4 Block Diagram of the Trellis Encoder

The convolutional encoder is a programmable implementation as shown in Figure 5. It is a feedforward encoder, where T_s is a delay of one symbol time, '⊕' is binary exclusive-OR, and '⊗' is binary AND. The **a** and **b** coefficients are programmable and are sent to the transmitter from the receiver during the startup phase. The encoder adds redundancy to the data stream to aid in the correction and prevention of errors [4]. The receiver will decode the sequence received according to these coefficient parameters.

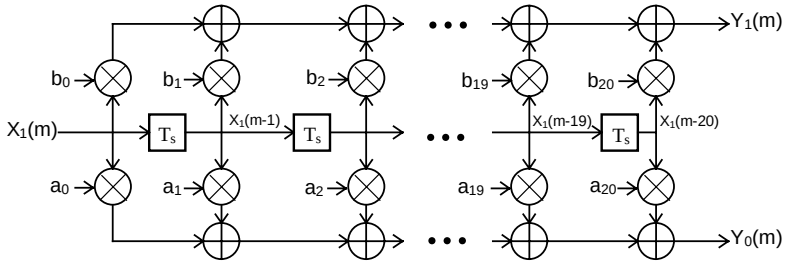

Figure 5 Block Diagram of Programmable Convolutional Encoder

The trellis coded PAM pulses are passed to the Tomlinson precoder. A diagram of the precoder is shown in Figure 6.

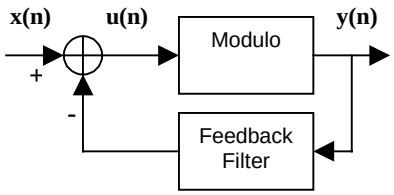

Figure 6 Block Diagram of Tomlinson Precoder

The precoder performs, in the transmitter, the same function as the DFE does in startup mode. This method provides many advantages over the startup mode structure. One advantage stems from the fact that equalization performed in the receiver not only counters the effects of the channel response, but also amplifies the noise induced by the channel. Because the precoder filters the signal before transmission, no noise is present to be amplified. Other advantages are gained by removing the slicer. Because the precoder is operating directly on the undistorted data, no decisions need to be made (hence no slicer). This eliminates the chance that a decision error will be propagated through the feedback filter. Also, the DFE requires a decision to be made with zero delay. The trellis decoding algorithm, discussed latter, introduces decision delay and so can not be implemented within the DFE. Placing this decision after equalization allows us to use both encoding and feedback equalization and recognize the combined gains they provide [6][7].

The result of the feedback filtering, shown as u(n), is no longer guaranteed to be bounded to the range (-1,1]. The modulo operator brings the output back into this range in a manner that is reversible at the receiver. By definition, the modulo block determines and integer, d(n), such that $-1 \leq u(n)+2 \bullet d(n)/16 < 1$. The output is then calculated with: $y(n) = u(n)+2 \bullet d(n)/16$. This will result in an expanded symbol set at the receiver.

The spectral shaper, receiver front end, echo canceller, and linear equalizer all provide functionality equivalent to startup mode. However, because the training sequence is no longer available a technique called blind equalization must be employed for adapting the linear equalizer during data mode [8]. This adaptation will counter slowly time-varying properties of the channel caused by environmental changes surrounding the transmission media.

The equalized symbols are passed to the trellis decoder. The decoder uses the Viterbi algorithm to decode the symbols generated by the convolutional encoder. A full discussion of the Viterbi decoding algorithm is given in [9]. The Tomlinson modulo operator recovers the original symbol from the expanded symbol set produced by the precoder. The symbols are then converted back to a bit stream by the level-to-bit mapper. The receive framer processes the resultant bit stream. The framer first scans for the sync word that marks the start of a new frame. Once this is detected, the DS-1 payloads are extracted and output.

3 Modeling and Simulation

Our implementation plan is to perform the simulation as two separate parts, a startup mode simulation and a data mode simulation. We will first run the startup simulation with a training sequence that allows the adaptive equalizer to converge on the channel. On termination, the startup equalizer coefficients will be saved to a file. The data mode simulation will then be run. The precoder will be programmed with the coefficients saved by the DFE in startup mode. We hope to do all modeling in the synchronous data-flow (SDF) domain of Ptolemy.

We plan to simulate these components as specified in the draft standard: scrambler/descramber, mappers, linear equalizer, DFE, Tomlinson precoder, and trellis encoder/decoder. We will leverage the wealth of pre-existing signal processing and communications components that Ptolemy provides to build most of the simulation blocks, adding code only when needed. At this point, the most significant code development we anticipate doing is the trellis decoder.

The primary impact of our project will be to provide a framework for evaluating HDSL2 designs. This is important because the draft standard gives a significant degree of designer flexibility by including programmable components and specifying implementation details only when necessary. Another impact will be the contributions our design will make to Ptolemy.

4 References

[1] J. Lechleider, "High Bit Rate Digital Subscriber Lines: A Review of HDSL Progress," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 6, Aug. 1991.

[2] M. Rude, "Draft for HDSL2 Standard," ADC Telecommunications contribution, T1E1.4/99-006, Jan. 1999.

[3] R. Goodson, K. Schneider, and J. Moore, "Proposal for Single-Loop HDSL Using Simple Coded PAM," ADTRAN contribution, T1E1.4/96-037, Apr. 1996.

[4] E. Lee and D. Messerschimitt, *Digital Communication*, 2nd Edition, Kluwer Academic Publishers, ISBN 0-7923-9391-0, 1994.

[5] R. Gaikwad and R. Baraniuk, "Optimal Transmit Spectra for HDSL2," Rice University contribution, T1E1.4/98-162R1, Jun. 1998.

[6] G. Pottie and M. Eyuboglu, "Combined Coding and Precoding for PAM and QAM HDSL Systems," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 6, Aug. 1991.

[7] A. Aman, R. Cupo, and N. Zervos, "Combined Trellis Coding and DFE through Tomlinson Precoding," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 6, Aug. 1991.

[8] C. Johnson, Jr., P. Schniter, T. Endres, J.Behm, D. Brown, and R Casas, "Blind Equalization Using the Constant Modulus Criterion: A Review," *Proceedings of the IEEE, Oct. 1998.*

[9] A. J. Viterbi, "Convolutional Codes and Their Performance in Communications Systems," *IEEE Transactions on Communications*, Oct. 1971.