IEEE 802.11
Wireless Access Methods and Physical Layer Specifications

TITLE: A Look at Some Scrambling Techniques Used in Various Data Transport Protocols

DATE: Nov. 8-12th,1993

AUTHOR: Ed Geiger
Apple Computer
One Infinite Loop
Cupertino, CA 95014
edg@goofy.apple.com

Introduction

This paper looks at some of the various forms of scrambling used in telecommunication data transport protocols. Hopefully this discussion will demonstrate some of the advantages and disadvantages associated with scrambling and provide insight into which scrambling method if any, best fits our application.

1.0 Scrambling

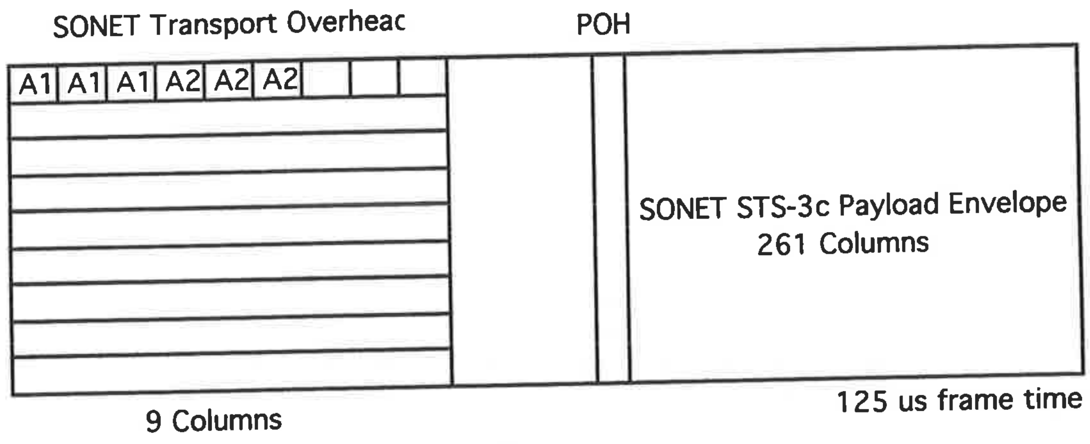
Generally there are two areas of interest relative to most data transport protocol scramblers:

- Synchronization
- Error Multiplication

Synchronization refers to how the receiving node synchronizes its scrambler to the one located in the transmitting node. Scrambler synchronization requires two conditions to be met at both the transmitting and receiving nodes. First, that each node start scrambling in the same place in the data stream and secondly, that each node use the same seed at at the start. Error multiplication refers to how some scramblers might take a single bit error in the received data stream and exploded that error into several bits errors when unscrambling.

2.0 SONET's Frame Synchronous Scrambler

Frame synchronous scramblers use some framing indicator to start and seed the scrambler's polynomial. An example of this type of scrambler is one use in SONET (Synchronous Optical NETwork). Figure 1 shows the SONET frame for STS-3 (155M bits).



SONET STS-3c Frame Structure
Figure 1

SONET is a byte oriented data transport protocol which uses a 125 usec frame system and a statical framing mechanism to acquire frame sync. The frame indicator is a string of six bytes refer to as the A1,A2 bytes. The SONET frame synchronous scrambler is 7 bits and is seeded with 7Fh during the FRAME pattern. It is turned on just after the last A2 byte in the frame sequence and turned off before the first A1 byte of the next frame sequence. The scrambler polynomial is:

$$1+x^6+x^7$$

The incoming data stream is added bit by bit, modulo 2, to the output from the x^7 position of the scrambler. (1)

2.1 Advantages and Disadvantages

This scrambler in it implementation serves two purposes. The first is to attempt to prevent the framing pattern from appearing in the user data and confuse the statical framing mechanism during startup. The second purpose is to serve as a whitener for breaking up long runs of ones and zeros.

The primary advantages of this type of scrambler is that it is simple to implement and has a bit sequence which doesn't repeat for 120 plus bits. It also doesn't do any error multiplication when a erred bit is de-scrambled. But the disadvantage is that is doesn't handle the pathological case well. For example, there will always be some pattern of data which matches the scrambler's bit inversion sequence, resulting in a long string of ones or zeros. SONET is designed to handle run lengths of 200 plus bits so the odds of this knocking out the clock PPL is pretty low but this occasionally happens. The biggest question regarding this failure is will it happen again on the re-transmission of the same data? In other words, can the offending data sequence ever be sent over the data transport?

The answer is yes it can but the solution is not with the scrambler but with how user data is placed into the SONET User Payload. Data packets are always insert into the payload asynchronously to the framing bytes. This has the effect of running a different scrambling sequence over the offending data packet the next time it is re-transmitted

3.0 ATM's Self Synchronous Scrambler

The BISDN world was forced to use a scrambler to solve the problem of long runs lengths of zeros confusing the HEC cell delineation scheme in the case of nulls. They also wanted this scramble to not be tied to a framing mechanism to avoid the pathological case which exists with frame synchronous scramblers. The scrambler chosen was a self synchronous scrambler using the following polynomial:

$$1 + x^{43}$$

This self synchronous scrambler does not require seeding at any specific time for proper operation. At the transmitter, the scrambler is seeded with a random pattern. At the receiver, the de-scrambler seeding is accomplished when it receives the first 43 bits from the transmitter. Like the frame synchronous scrambler, it must be started and stopped at the same point in the data stream at both nodes. The advantage of this scrambler is that scramble pattern is random and that not repeatable from a given reference point. This handles nicely the pathological problem the frame synchronous scrambler has. Unfortunately the self synchronous scrambler has several disadvantages. First, it must be seeded with 43 bits before it can actually de-scramble data properly. This poses a problem whenever the first packet is sent and anytime the scrambler lose sync because of a lost packet or errors at the end of the last packet. A second disadvantage is that this scrambler will do error multiplication. Any erred bit entering the de-scrambler at the receiver will produce another erred bit 43 bits away.

4.0 Discussing the Alternatives

Of the two scramblers discussed, the pathological failure of the frame synchronous scrambler is probably the biggest challenge to overcome. Any and all data patterns must be able to be carried by the WLAN. There are several possible ways to correct this failure. One is to come up with a system to dynamically change the way data is positioned in the packet relative to the frame indicator. Another is to have several different seed patterns for the scrambler and have some method for cycling through the various seeds. Both these approaches probably require some information to be transmitted as part of the packet to indicate to the receiver how to unscramble or unbundle the data. Both these approaches would require that some form of error protection must accompany the information describing these procedures to avoid error multiplication via the scrambler.

Error multiplication is also a serious subject. In the case of the data transport protocol proposed by Apple, error multiplication can impact the ability to apply forward error correction in some cases. In addition, errors must be detectable for the WLAN to function reliably. CRC polynomials are designed to detect errors within certain limits based upon the expected errors in the system. Error multiplication tends to reduce the strength of the CRC's ability to detect all the errors for which it was originally designed. This may place constraints on packet length.

The self synchronous scrambler used in ISDN is fairly long. It is that length to insure that when error multiplication occurs, its distance is such that in doesn't impact the CRC32's ability to detect errors within a 65k byte packet. Since a 65k byte packet is way beyond anything packet size we probably will send over the air at one time, this length might be reduced for our implementation. This would require further study. The more significant problem with a self-synchronous scramblers is that they require a certain number of scrambled bits to enter the scrambler before the scrambler outputs valid de-scrambled data. In a wireless environment where packet loss is common, it is a given that somewhere in the front of the user payload a field for scrambler synchronization would be required.

What might be an alternative to scrambling? One might be to do bit stuffing every 16 bits. This bit stuffing would expand every 16 bits of user data to 17 bits. The 17th bit would be the compliment of the 16th bit. This bit would only be used by the radio to breakup any string of 16 ones or 16 zeros. This doesn't impact any coding or error detection and correction scheme. It also doesn't require start and end delimiters nor does it expand the data field into variable lengths such as the case with SDLC. This does require building radios which handle run lengths of 16 bits but our research says this is achievable.

References

1. American National Standards for Telecommunications, "Digital Hierarchy, Optical Interface Rates and Formats Specifications", ANSI T1.105-1988.

2. Draft American National Standards for Telecommunications, "Broadband ISDN Customer Installation Interfaces: Physical Layer Specification", T1E1.2/93-020R2.