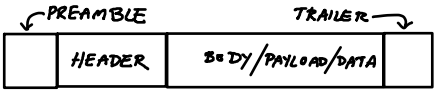
Framing

After this lecture you should be able to: determine if a data communication system requires framing or not, and choose between bit- and byte-oriented framing. You should be able to insert and remove escape sequences and bit stuffing from bit- and byte- sequences respectively. For each of the framing techniques described in this lecture (line coding violations, byte escape sequences, and HDLC flags) you should be able to: write out a properly-framed bit- or byte-sequence, and extract the data sequence from a bit- or byte-sequence that contains framing information.

Introduction

Data communication systems often need to group bits or bytes into logically related groups called "frames." Each frame contains information that is grouped and treated together. All of the data within a frame typically comes from the same data source and is destined for the same sink. The data within a frame typically is treated as a whole for the purposes of error checking and for routing through a network.

A frame typically consists of a header, data and a trailer:



- the header contains information about the frame. This can include destination and source addresses, frame length, priority, and many other fields. The header sometimes includes a "preamble" for physical-layer synchronization.
- the data is typically a variable-length field and contains the "payload" data that is being communicated.
- the trailer often includes a checksum that allows for detection of errors and may include other fields.

This lecture describes various ways that have been developed to indicate the start and end of a frame.

Bit- and Byte-Oriented Framing

Some protocols transmit arbitrary-length sequences of bits while others transmit data as complete bytes (multiples of 8 bits). The type of framing must be consistent with the underlying structure of the data.

Bit-oriented protocols are often used when maximum efficiency is desired. The length of each portion of data within a frame can be set to the minimum number of bits required. A typical example is status or control bits used to manage hardware such as the values read or written to control registers.

Byte-oriented protocols are used when the underlying data source consists of bytes. Most modern data networks use byte-oriented framing.

Physical-Layer Framing Information

One way to mark the start or end of frame is to use a unique physical-layer waveform. One example is to violate some rule of the line code. For example, it could involve transmitting the same polarity mark twice when using an AMI line code.

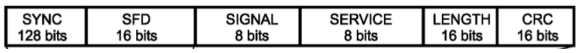
Exercise 1: Draw the waveform for an AMI-NRZ encoded sequence of bits '01101' assuming the previous mark was transmitted as a high. Draw the waveform assuming the second '1' indicates the start of a frame.

The impact on physical layer characteristics such as the DC content of the signal is minimized because this violation only happens at the start, and possibly the end, of the frame.

Exercise 2: What is the average DC value for the two cases in the previous exercise?

Another common PHY-layer framing techniques is to use of a *preamble*. A preamble is a special type of PHY-layer "frame before the frame." The preamble is encoded in a way that can be easily decoded and carries information about the rest of the frame. Typically the preamble will help the receiver synchronize to the received signal and contains all the additional information required to receive the frame including the data rate to be used in the rest of frame and the frame duration.

The following figure, from the IEEE 802.11 specification, shows the preamble used by an 802.11 wireless LAN frame:



SYNC is a waveform used by the receiver for synchronization. SFD (start of frame delimiter) marks the end of the SYNC field. The SIGNAL field decribes the modulation format and data rate and the LENGTH field is the duration of the frame in microseconds. The CRC field is used for error detection.

Exercise 3: What might be some advantages of using a preamble? What might be some disadvantages?

Escape Sequences

An escape sequence consists of a special character that is used as a prefix to indicate that the following character should not be treated as data but as a command to perform some action. For example, if the ASCII ESC (0x1B) character is used as the escape character, then ESC followed by another character is interpreted as a special command. For example, ESC followed by the character 'A' (ESC-A) might mean to move the cursor up one line.

To allow the escape character itself to be transmitted, an escape sequence can be defined that means "insert one escape character." Often this sequence is the escape character repeated twice.

Exercise 4: By how much does the use of escape sequences slow down a link if random 8-bit characters are being transmitted? What is the overhead if a continuous stream of escape characters need to be sent over the link?

Byte-Oriented Framing

Escape sequences can also be used to mark the start and/or end of a frame. For example, we can insert one escape sequence to mark the start of a frame and another to mark the end of a frame.

Another alternative is to use specific characters to indicate the start or end of a frame. Of course, if these characters are to be allowed within the frame some sort of escape sequence must be defined to "escape" them.

The use of specific characters for framing is the origin for the names of many of the ASCII characters such as STX (start of text), ETX (end of text), and ESC/DLE (escape or "data link escape").

Bit-Oriented Framing

Bit-oriented framing often uses the special bit sequence (01111110) consisting of six '1' bits preceded and followed by one zero bit. This "flag" sequence is added to the start and end of each frame.

When this sequence occurs in the data within the frame, bit stuffing (stuff a 0 after five 1's) is used to make sure this sequence does not occur within data.

Bit stuffing can thus be used not only to prevent long runs of 0's or 1's (as covered earlier) but also to escape flag sequences in bit-oriented framing.

Exercise 5: Write out the complete sequence of 1's and 0's required to transmit the 12 bits 0110 1111 1100. Include the start and end flag sequences and any necessary bit stuffing.

Exercise 6: An HDLC receiver sees the sequence 1000 0111 1110 1111 1001 0111 1110 0110. What data bits were contained within the frame?

Padding

The physical layer transmits symbols that may contain more than one bit. Thus the number of symbols required to transmit a frame may not be an integer. In this case additional "padding" bits must be added to increase the number of bits to the next-largest multiple of the number of bits per symbol. These extra padding bits are ignored by the receiver.

Exercise 7: A physical layer transmits 3-bits per symbol. A frame of 128 bytes is being transmitted. How many padding bits will have to be added to the frame?

Implementation

Detection, insertion and removal of framing and escape sequences is implemented with sequential digital logic circuits and is conveniently defined in terms of state machines.

Exercise 8: How many states are required to implement a circuit that detects stuffed bits in an HDLC frame? How many bits of state are required?