Framing

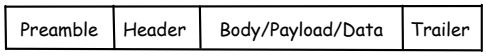
After this chapter you should be able to: determine if a data communication system requires framing or not, and choose between bit- and byte-oriented framing. You should be able to insert and remove escape sequences and bit stuffing from byte- and bit sequences respectively. For each of the framing techniques described in this chapter (line coding methods, 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. You should be able to add and remove padding bits.

Introduction

Data communication systems often need to group bits or bytes into logically-related groups called "frames" or "packets." Each frame contains information that is treated together. All of the data within a frame usually comes the same source and is going for the same destination. The data within a frame is typically treated as a unit for error checking and for routing through a network.

The word "frame" typically refers to grouping of bits or bytes at the physical and data link layers (e.g. an "Ethernet frame") while the word "packet" is often used for higher-level protocols (e.g. an IP packet).

A frame is typically composed of header, data and trailer portions, transmitted in that order:



- the header contains information about the frame. The format is defined by the protocol. It 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 appears at the end of the frame and often includes a checksum that allows for detection of errors.

Bit- and Byte-Oriented Framing

Some protocols allow frames that contain with an arbitrary number of bits while others protocols require frames to contain multiples of 8 bits (bytes).

Bit-oriented framing, not often used today, is used when the payload has a small number of bits and the overhead of rounding off to a multiple of 8 bits is not desired.

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

Physical-Layer Framing Information

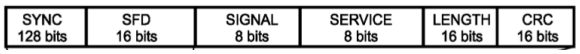
One way to mark the start or end of a frame is to use a unique physical-layer waveform. One example is to violate a line coding rule. For example, it could involve transmitting the same polarity mark twice when using AMI. Another example is the use of block code output words that are reserved to indicate the start or end of a frame such as JK and TR for 4B5B.

The impact on physical layer characteristics such as the DC content of the signal is minimized because these special violations or block codes words only appear at the start – and possibly the end – of the frame.

Exercise 1: Draw the waveform for an AMI-RZ encoded sequence of bits '011011' assuming the previous mark was transmitted as a positive pulse. Draw the waveform assuming the second '1' indicates the start of a frame.

Another common framing technique is to use a *preamble*. A preamble is a special type of physical-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 may contain additional information required to receive the frame such as the modulation 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 describes 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 2: Preambles such as this allow multiple transmission formats to be used in a backwards-compatible way. What might be some disadvantages of using such a preamble? *Hint: to be decoded by old ("legacy") devices the preamble must be transmitted at the lowest possible data rate. This can be 100 times slower than the fastest devices.*

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.

Strings in software often use a backslash character (\) as an escape character. For example, the two characters, `\n` are interpreted as a single line feed (`0x0a`) character.

The ASCII ESC (`0x1b`) character was originally intended to be used as an escape character for printers and displays. In this case ESC followed by another character is interpreted as a special command. For example, ESC followed by the character 'A' (ESC-A) might be interpreted by a display to mean that the cursor should move 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 3: In this case, by how much does the use of escape characters slow down a link transmitting a continuous stream of escape characters?

Byte-Oriented Framing

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 mechanism must be defined to "escape" them.

The use of specific characters for framing is the origin for the names of many of the ASCII control characters such as STX (start of text), ETX (end of text), and ESC/DLE (escape or "data link escape"). These were used in protocols, such as "Bisync", that are no longer in use.

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.

An example is RFC 1662 ("PPP in HDLC-like Framing") which is often used when IP frames are to be sent over a data link that does not support a means of separating frames. An typical example is sending IP frames over a serial ("RS-232") interface.

PPP uses the PPP 'flag' character (`0x7e`) to indicate the start and end of a frame and the PPP escape character (`0x7d`, not to be confused with the ASCII ESC character, `0x1b`) to escape both flag and escape characters within the frame. In addition, the byte being escaped is XOR'ed with `0x20` (this inverts bit 5 and makes control characters – which are often escaped as well – printable).

Exercise 4: What sequence of bytes would be sent to transmit a PPP-encapsulated frame containing the bytes `0xff 0x03 0x7d 0x1b 0x7e`?

PPP framing also requires adding a header, error-checking and includes optional features such as compression and escaping of certain control characters. These will be covered in a separate course.

Bit Stuffing

Bit stuffing is a technique used both for framing and to avoid long runs of all 1's or all 0's. The transmitter and receiver both count the number of consecutive 1's (and/or 0's) seen and if this count reaches some upper limit the transmitter inserts a bit of the opposite value and the receiver removes this extra bit.

The reason this works is that the transmitter and receiver both keep track of the number of consecutive 0's or 1's seen so they both know when a bit should be "stuffed." Note that the stuffing must be done regardless of the value of the next bit because the receiver does not know what that value will be.

One disadvantage of bit stuffing is that the data rate is no longer predictable because it depends on how often bit stuffing happens which in turn depends on the data values being transmitted.

Exercise 5: You receive the sequence of bits 10001101 and are told that bit stuffing was used to limit runs of 0 to three or fewer. What was the original data sequence?

Bit-Oriented Framing

Bit-oriented framing uses a special bit sequence to indicate the start and end of each frame. The most common bit-oriented framing protocol is that used by the HDLC (High-Level Data Link Control) and many similar protocols.

This framing technique uses a flag sequence of consisting of six '1' bits preceded and followed by one zero bit (01111110). This "flag" sequence is added before and after each frame.

To ensure this sequence doesn't occur within the frame, bit stuffing – stuffing a 0 after five consecutive 1's – is used.

Exercise 6: 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. Assume bits are transmitted lsb-first.

Exercise 7: An HDLC receiver sees the sequence 1000 0111 1110 1111 1001 0111 1110 0110. What data bits were contained within the frame?

Note that the frame defined by bit-oriented framing can contain fields with different number of bits and the bit order within each field could vary (e.g. a field may be transmitted lsb-first).

Framing Fixed-Length Frames

For fixed-length frames the start of the frame can be marked by a bit or byte that always has a specific value (e.g. always 0 or always 1). If received does not see this bit or byte value in the expected location it recognizes that it has lost frame synchronization. To resynchronize it assumes the next bit or byte with the correct value is the framing bit or byte. If the framed data is random (e.g. because it has been scrambled) the receiver will eventually synchronize. This technique adds an overhead of one bit or byte to each frame.

Padding

The physical layer transmits symbols that may contain more than one bit. For example a system using four symbols transmits two bits per symbol. When the number of bits in a frame is not a multiple of the number of bits per symbol, additional "padding" bits must be added to the end of the frame to bring the number of bits up to the next-largest multiple of the number of bits per symbol. These extra padding bits can be ignored by the receiver if it knows the number of bits in the frame.

Exercise 8: 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?