In the name of God

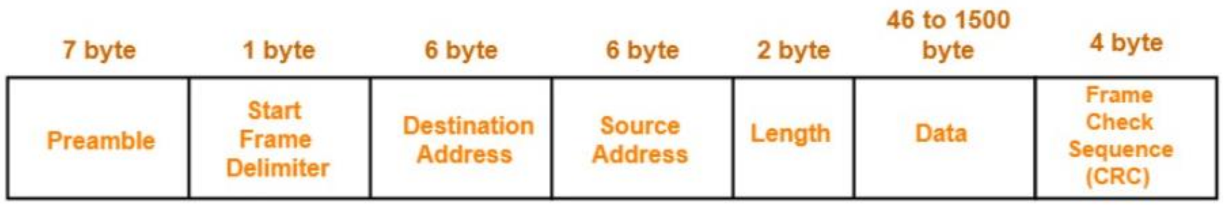
Digital Logic Circuits Final Project

Professor: Dr.Shaahmansouri

Ahmadreza Tavana: 98104852

1) Preamble: Ethernet frame starts with 7-Bytes Preamble. This is a pattern of alternative 0's and 1's which indicates starting of the frame and allow sender and receiver to establish bit synchronization. Initially, preamble was introduced to allow for the loss of a few bits due to signal delays. But today's high-speed Ethernet don't need Preamble to protect the frame bits.

Preamble indicates the receiver that frame is coming and allow the receiver to lock onto the data stream before the actual frame begins.

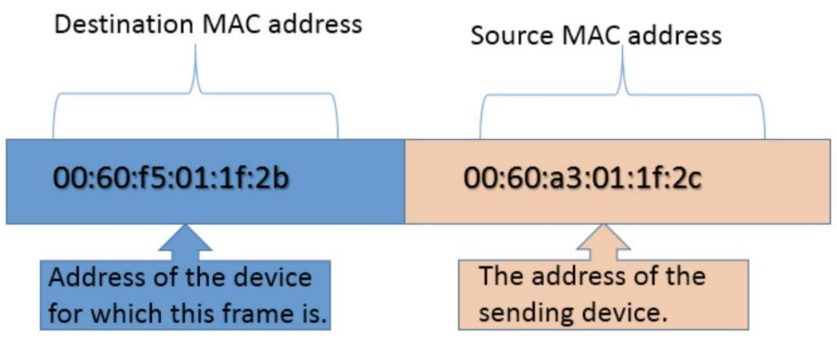


2) Start of frame delimiter(SFD): This is a 1-Byte field which is always set to 10101011. SFD indicates that upcoming bits are starting of the frame, which is the destination address. Sometimes SFD is considered the part of Preamble, this is the reason Preamble is described as 8 Bytes in many places. The SFD warns station or stations that this is the last chance of synchronization.

As it mentioned before SFD can't be changed.

3) Destination Address: This is 6-Byte field which contains MAC address of source machine for which data is destined. The address in a frame is compared to the MAC address in a device. If there is a match, the device accepts the frame. It can be a unicast, multicast, or broadcast address.

Source Address: This is a 6-Byte field which contains the MAC address of source machine. This field identifies the originating NIC or interface of the frame.


Destination MAC address
Source MAC address
00:60:f5:01:1f:2b
00:60:a3:01:1f:2c
Address of the device for which this frame is.
The address of the sending device.

4) Ether Type (Length): Length is a 2-Byte field, which indicates the length of Payload.

The maximum value that can be accommodated in this field is $2^{16} - 1 = 65535$ but it doesn't mean maximum data that can be sent in one frame is 65535 bytes. The maximum amount of data that can be sent in an Ethernet frame is 1500 bytes.

5) As mentioned in previous part the size of payload is stored in Ether Type so we can calculate the size of entire frame by Ether Type.

6) Frame Check Sequence(FCS): FCS field is a 4-Byte field, which is used to detect errors in a frame. It uses a cyclic redundancy check(CRC). The sending device includes the results of a CRC in the FCS field of the frame. The receiving device receives the frame and generates a CRC to look for errors. If the calculations match, no error occurred. Calculations that do not match indicate that the data has changed; in such a case, the frame is dropped. A change in the data could be the result of a disruption of the electrical signals that represent the bits.

7) CRC32 is a popular checksum algorithm used to detect data corruption. Multiple variants of the algorithm exist which have similar mathematical properties. The most common variant of the CRC32 checksum, sometimes called CRC-32b, is based on the following generator polynomial:

$$p(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1.$$

How CRC32 works:

1) Consider the message as a sequence of n bits in chronological order (if the message is structured in words or bytes: with low-order bit first unless otherwise specified).
2) Add 32 zero bits to the end of sequence to form a sequence of n+32 bits.
3) Complement the first 3232 bits of that sequence.
4) Form the binary polynomial of degree (at most) n+31, with term x^{n+32-j} present (resp. zero) when the jth bit in the result of the previous step is *one* (resp. *zero*), for j with $0<j\leq n+32$.
5) Compute the remainder of the polynomial division of that polynomial by the binary polynomial $p(x)$, forming a binary polynomial of degree (at most) 31.
6) Form the 32-bit sequence with the jth bit *one* (resp. *zero*) when the term x^{32-j} is present (resp. absent) in that polynomial, for j with $0<j\leq 32$.
7) Append that 32-bit sequence to the **ORIGINAL** message

The example is in the folder of project: CRC32_Example.xlsx

How my code works: first of all, we should check the preamble and SFD. After that the MAC addresses will come which we don't need them right now (but we ought to store them and check with the source and destination MAC addresses). Then Ether Type will come which we need it because represents the size of Payload. After Ether Type the most important part of frame which is Payload will come and we should store it by FCS after it.

Now we have all the details and it's time to check possible errors. First we calculate the reminder of Payload+FCS by p(x). if it was zero so there isn't any error in Payload otherwise there is an error. The error may have occurred in both Payload and FCS but we consider it is in Payload and change every bit of Payload and check the reminder again. The Payload which reminder is zero is the correct Payload and put it into the output.

Test Benches:

1) The first test bench tests the code when there isn't any error in the input.

Payload: 00001010

FCS: 00101111100010101101011011010110

Error: There isn't any error

2) The second test bench tests the code when there is only an error in input.

Payload: 00001110 (Wrong, the correct form is 00001010)

FCS: 00101111100010101101011011010110

Error: There is an error in 6th bit.

References:

1) wikipedia.org
2) https://crypto.stackexchange.com/
3) www.geeksforgeeks.org/
4) commandlinefanatic.com