

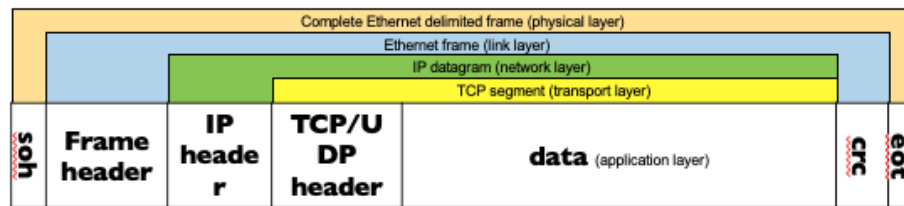
Exploration: Ethernet Frame and Multiple Access

Introduction



In this exploration we continue the discussion of ethernet standards that are common for all ethernet implementations. That is, the ethernet standards that are common for the Link Layer. To do this we will also look a bit into the physical layer. Ordinarily, we leave the physical layer to the electrical engineers. But we will look at some of what happens when an ethernet frame gets transmitted to another node. We won't look at the electronics, but will take a more abstract view.

Ethernet Frame



We've already looked at the ethernet frame format. The frame is a packet that has a specific format for a specific technology. Each of those transmission technologies must have some way to tell when a frame begins and when a frame ends. This is typically done by putting a byte at the beginning of the frame and at the end of the frame so the frame can be recognized by the receiving hardware.

These special characters can also be used to detect sending or receiving problems. If there is a missing byte at the end, it may mean that the sending computer has crashed, while a missing byte at the beginning may mean that the receiving computer has somehow missed part of the transmission.

The characters that are chosen can be any of the unused values that would not appear in the frame itself. If the frame is limited to the printable ascii characters, you can use some of the non-printable codes such as 0 – 31.

- Ethernet uses the “start-of-header” (soh) character for the beginning of the frame, which is ascii character #1.
- Ethernet uses the “end-of-transmission” (eot) character for the end of the frame, which is ascii char #4.

The frame as its transmitted from the link layer, gets encapsulated between the soh and eot characters (see above). The destination link will look to receive all the data between any soh and eot characters. If there are no other errors, it can pass the received data up to the higher layers.

Of course there's a problem if you want to send anything other than printable characters, because there will be no unused characters available. If the soh or eot appear anywhere in the binary bytes you are trying to send, it would confuse the receiver, and it would be impossible to reliably send and receive a complete frame.

The solution is to encode soh, eot, and other special characters for unambiguous frame transmission. The sender will notify the receiver of the specific character encoding in the frame preamble.

Byte Stuffing

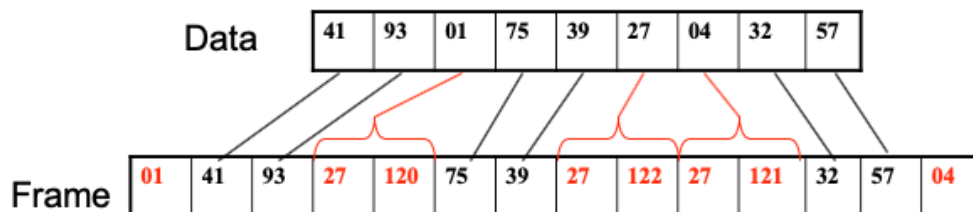
The process of encoding special frame characters is handled by a protocol called byte-stuffing. With this protocol, you use the same soh and eot characters. But whenever these appear in the binary data they will be replaced by two characters: The third special character, such as esc or ampersand, and a different character. This prevents confusion at the receiving end, because there will only be one real soh and eot delimiting the frame.

Character in data	Characters sent
soh (01)	esc x(27 120)
eot (04)	esc y (27 121)
esc (27)	esc z (27 122)

Here's an example... This table establishes the protocol for byte stuffing. The special character that we will use is the esc character or ascii 27.

Using the table we see that soh becomes esc x, eot becomes esc y, and esc itself becomes esc z.

You might be wondering what would happen if your binary data already contained any of the "byte-stuffed" characters (in the right column of the table). How would those be encoded? The answer is, no differently. Let's see how this works with data:



In this example, note that the soh and eot are still the same. All byte stuffing does, is remove any of the false soh and eot from the data before the frame is sent. After it is received, the receiver will go through the data, and revert any byte-stuffed characters to their original form, and so the original data is recovered.

This is a clever way to handle any kind of data, with the same soh and eot.

Ethernet Multiple Access (MA) Protocol

Now that our data is framed and byte-stuffed, we are ready to send it over a shared media. What MA protocol does Ethernet use to accomplish this?

Ethernet will use Carrier Sense Multiple Access with Collision Detection (CSMA/CD). Here is how it works:

1. A NIC receives datagram from network layer, and creates the frame
2. If the NIC senses channel is idle:
 - a. It starts frame transmission
3. Otherwise, If the NIC senses channel is busy:
 - a. It will wait until channel is idle, and then transmit
4. If the NIC transmits the entire frame without detecting another transmission:
 - a. Then the NIC is done with frame
5. Otherwise, If the NIC detects another transmission while transmitting:
 - a. Then a collision has occurred: The NIC aborts and sends a jam signal
 - b. After aborting, the NIC enters exponential backoff and then will retry

Exponential Backoff

Exponential backoff is an adaptive retransmission algorithm that attempts to take into account the current load on the network. If there are many collisions, the sender can infer a heavier network load, and so the random wait time will be

longer.

We have seen a similar algorithm before. The transport layer's AIMD. The ethernet's version of this is to count the number of collisions that occur for a particular frame, and after the m th collision, the NIC chooses K at random from the set $\{0, 1, 2, \dots, 2^m - 1\}$. The NIC then waits $K * 512$ bit times, and then attempts to resend the data.

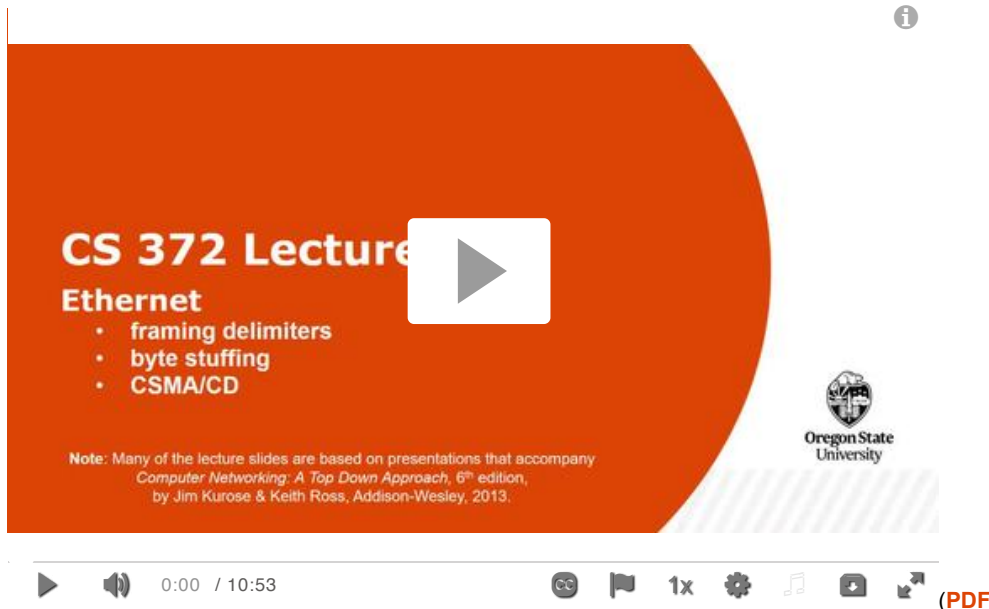
A bit-time is the time to transmit one bit... So if you have a 10 Mbps Ethernet, that takes about one tenth of a microsecond to transmit one bit. For the largest number in the set, after 10 collisions $K=1023$ and the wait time is about 52 ms.

This is all happening pretty fast. The whole idea is to back off randomly so that you don't end up with another collision with the same frame being sent from a different node. In other words, if two senders collided last time, the hope is that both will choose different random backoff times, and when they retransmit, they won't collide with one another again.

This concludes our discussion of the Link layer. For more on this topic be sure to view the video lecture, then test your knowledge with the Self-Check exercises below.

Video Lecture

Ethernet - Byte Stuffing



CS 372 Lecture
Ethernet

- framing delimiters
- byte stuffing
- CSMA/CD

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach*, 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Oregon State University

0:00 / 10:53

(PDF)

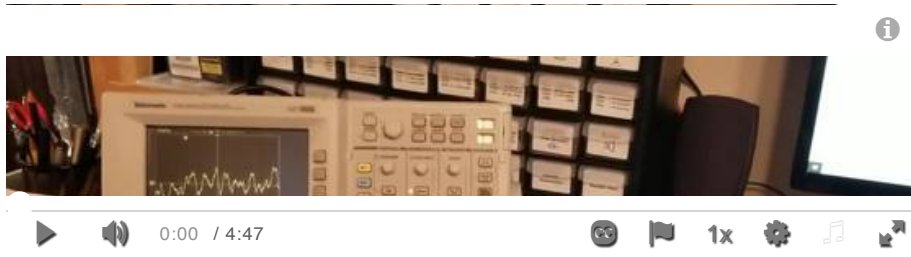
<https://oregonstate.instructure.com/courses/1798856/files/83165083/download?wrap=1>

<https://oregonstate.instructure.com/courses/1798856/files/83165083/download?wrap=1> | PPT

<https://oregonstate.instructure.com/courses/1798856/files/83165299/download?wrap=1>

<https://oregonstate.instructure.com/courses/1798856/files/83165299/download?wrap=1>

OSU student demonstration video



Self-Check Exercises

Draw the complete Ethernet frame (with Ethernet services, abstracting the lower layers to each layer's header/payload – assume TCP/IP).

Turn

Card 1 of 3



Given the following "byte stuffing" scheme (soh and eot are framing characters):

Character in data	Characters sent
soh	esc x
eot	esc y
esc	esc z

Character	Hex code
soh	01h
eot	04h
esc	1Bh
'x'	78h

'y'	79h
'z'	7Ah

Data:

78h	04h	1Bh	7Ah	01h	1Bh
-----	-----	-----	-----	-----	-----

If Ethernet-style byte stuffing is used to transmit Data, what is the byte sequence of the frame, including the framing characters? (Assume that all headers are included in Data.)

 Turn

Card 1 of 1