We discussed the meaning of *baud*.

The short version: In RS-232 contexts, *baud rate* is the same as *bits-per-second* counting the start and stop symbols.

Example: An RS-232 link is set up at 9600 baud using 8N2 signaling.
What is the maximum rate of sending data in bytes/second?

The Tx clock runs at 9600 Hz. Each 8-bit byte sent also requires one start bit and two stop bits ("8N2" implies this).
Thus each transmitted byte requires 1 + 8 + 2 = 11 bit-cells of time.
At 9600 Hz a bit cell takes 1/9600 s. A transmitted byte of useful data plus start and stop will take 11/9600 s.
The overall useful byte rate will be 9600/11 = 872.727 bytes/sec.
(If ASCII text is being sent with one bit of parity, then this channel can send 872.727 characters/second.)

But the "RS-232 definition of baud" is not the true definition of baud.
Baud is the reduced fundamental binary bit rate of communication after all redundancy is removed.
In practice, the baud rate of a channel is always less than (or equal to) the binary bit rate.

Example: Terrestrial broadcast digital TV in North America uses eight-level vestigial sideband modulation (8VSB).
The transmitter's symbol clock runs at 10.8 MHz. Each symbol bears (at most) the equivalent useful information of three binary bits. Thus the baud rate of a TV channel is 10.8 X 3 = 32.4 M-baud. Any information source that can be losslessly compressed to 32.4 Mbits/sec. or less can be sent via this channel. This is a 32.4 Megabaud channel.

Using the true definition of baud, the RS-232 channel in the example above is running at only 8727.27 baud. (Only 8727.27 bits per second of data can be passed through that channel.) We should say that the Tx clock is 9600 Hz, not that the baud rate is 9600.

1

What do we mean by, "removing redundancy?"

In the English language certain rules of spelling and grammar force redundancy into the text.
If I write, "Hello Worl"
Then you can fully understand that the message is "Hello World."

In another example, the letter "u" is supposed to follow "q" in normal spelling.
So, why don't we make a rule to just drop every "u" that follows "u" when transmititng normal text.
If I write that, "We will qit sending the letter 'u' in this situation." it is easy for you to re-insert the "u" and understand the message: "We will quit sending the letter 'u' in this situation."
But these are only minor examples based on some pretty complicated rules of spelling and grammar.

Huffman code finds and eliminates redundancy much more efficiently.
In Huffman code an information source is scanned in some way and a code table is produced and sent in the clear using a standard method (ASCII would work) and then following that the encoded message is sent.
Huffman coding is optimal in a certain narrow sense, but is not globally optimal for all information sources.

https://en.wikipedia.org/wiki/Huffman_coding

There is lots more to this topic beyond Huffman coding.

2

USB RS-232 As a quintessential example

From Asynchronous to Synchronous

use Ethernet as an example

Definition of a synchronous serial interface:

Data is sent in packets composed of many bits of data. The data may be streamed continuously or sent in bursts called packets. The on-going nature of data transmission allows timing information for symbol (bit or byte) decoding to be embedded in the stream of data.

The receiver recovers a clock signal from this embedded timing information. The recovered clock at the receiver is therefore phase coherent (has only timing delay) with the transmitted clock.

There is a very easy way to send the transmitter's clock information: Use a parallel wire (or wires!). Technically, this is a synchronous serial protocol. We will discuss a few examples later. (I^2C, SPI) However an extra wire for a clock signal only makes sense over short distances (across a circuit board). For USB, Ethernet, Firewire, etc. we cannot afford extra wiring just for the clock.

Rather than using an extra wire (copper is expensive) the data stream may be modified to include clock information.

3

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.

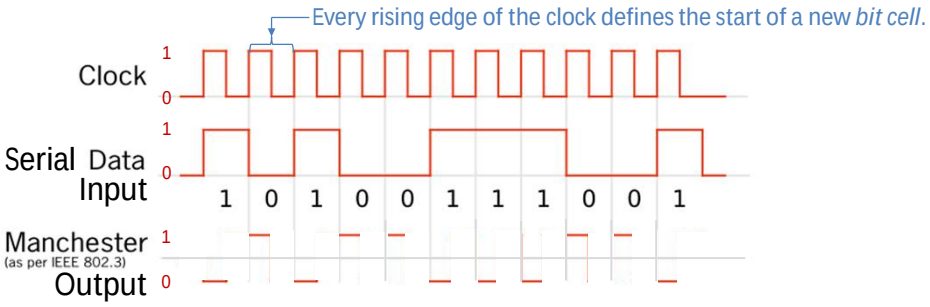
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

4

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.
The transmitter sends false data during the first half of the clock cycle



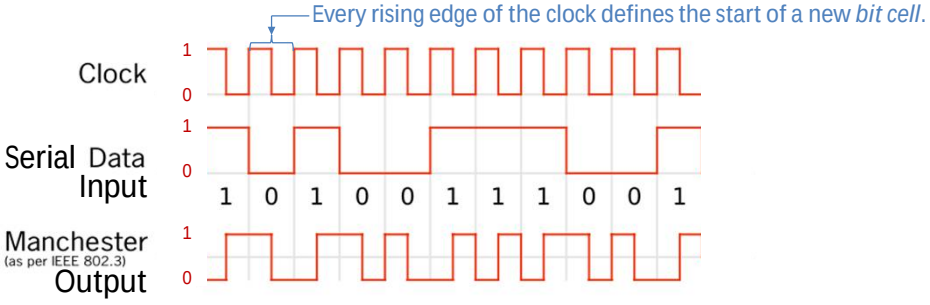
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

5

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.
The transmitter sends false data during the first half of the clock cycle and
true data during the second half of the clock cycle.



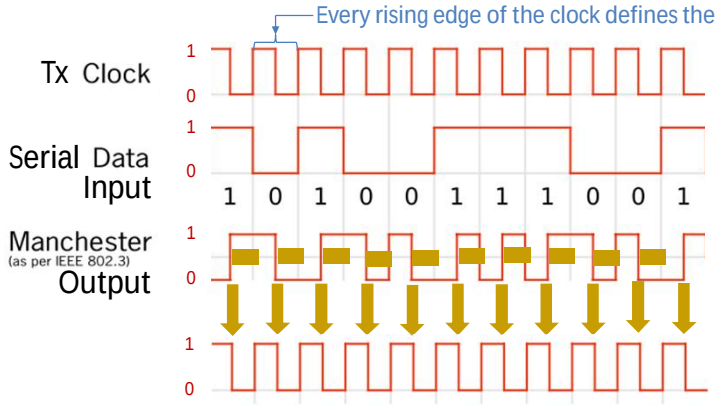
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

6

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.
The transmitter sends false data during the first half of the clock cycle and
true data during the second half of the clock cycle.
The result is a transition in the middle of every clock cycle—Use this to reconstruct the clock at the receiver.
(After each Manchester code transition, lock the clock detector out for a bit more than a half clock cycle.)



Rx Clock—identical to Tx clock except for propagation delay

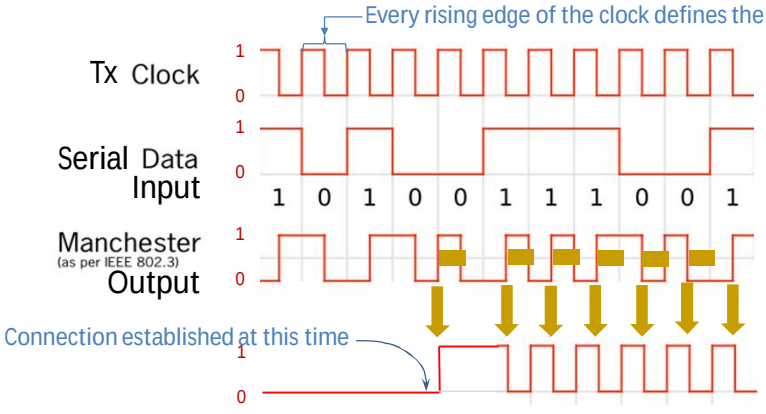
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

7

Manchester coding

A simple way to embed clock timing in the data stream

What if a connection drops in on an existing Manchester encoded stream and grabs a "wrong" edge for clock detection?



Rx Clock—falls into phase-lock after a few transitions.

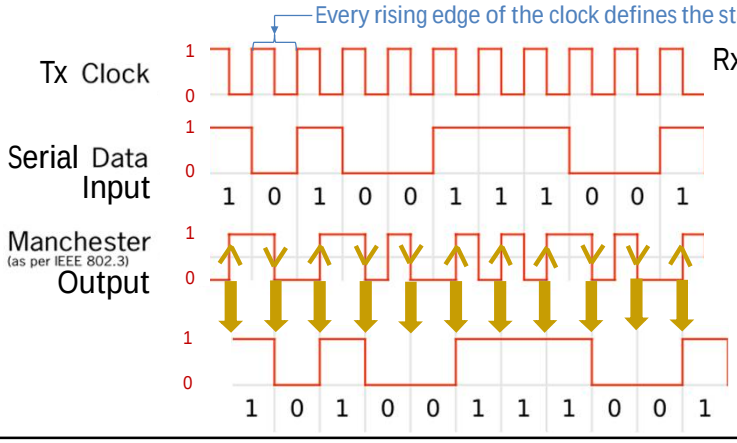
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

8

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.
The transmitter sends false data during the first half of the clock cycle and
true data during the second half of the clock cycle.
The result is a transition in the middle of every clock cycle—Use this to reconstruct the clock at the receiver.
The direction of the transition tells the receiver what the data was. Rising edge = logic-1, Falling edge = logic-0



Rx Clock—identical to Tx clock except for propagation delay

Rx Data—identical to Serial Data Input except for delay.

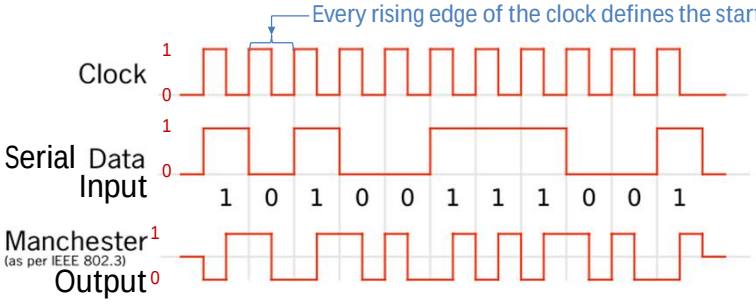
https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

9

Manchester coding

A simple way to embed clock timing in the data stream

The transmitter clock defines bit cells, say with each rising edge of the clock.
The transmitter sends false data during the first half of the clock cycle and
true data during the second half of the clock cycle.
The result is a transition in the middle of every clock cycle and
the direction of the transition tells the receiver what the data was. (Rising edge = logic-1, Falling edge = logic-0)



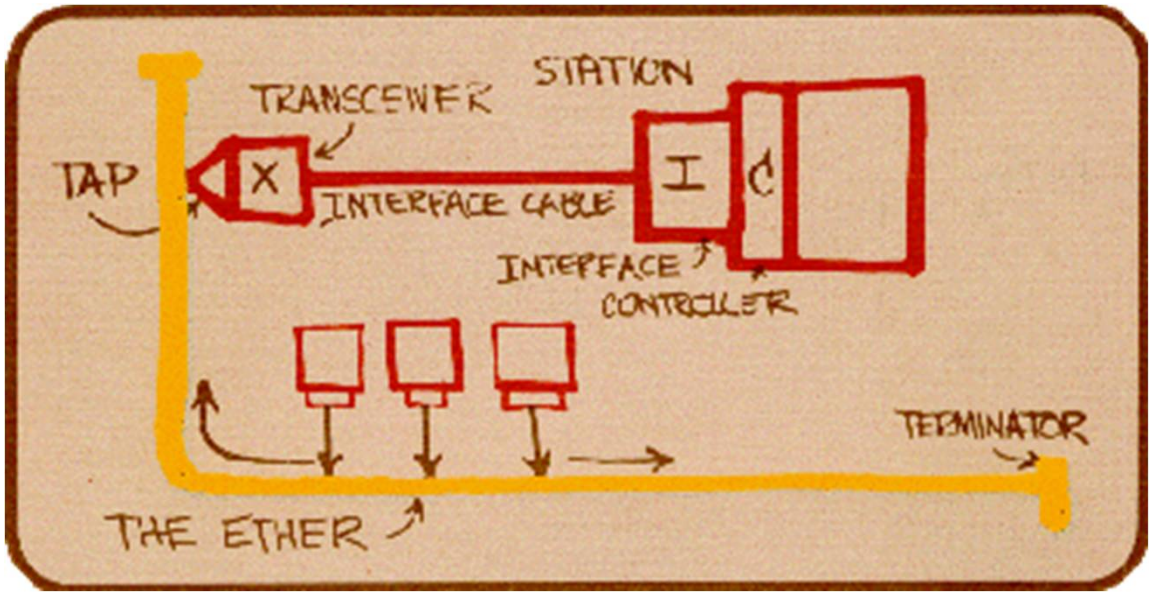
You might notice that if the communications channel can send pulses as fast as needed for Manchester code, then the clock info. is consuming ½ the bandwidth since a simple base-band channel can support 2 bits/clock cycle. (There is no free lunch—the clock info is not sent for free.)

RLL codes are even more efficient than Manchester code.
http://tomkleen.com/CSCI280/Null%20and%20Lobur%20Instructor%20Files/PowerPointPres_Ch2A_ECOA2e.ppt

https://commons.wikimedia.org/wiki/File:Manchester_encoding_both_conventions.svg

10

Ethernet—Originally it was done this way



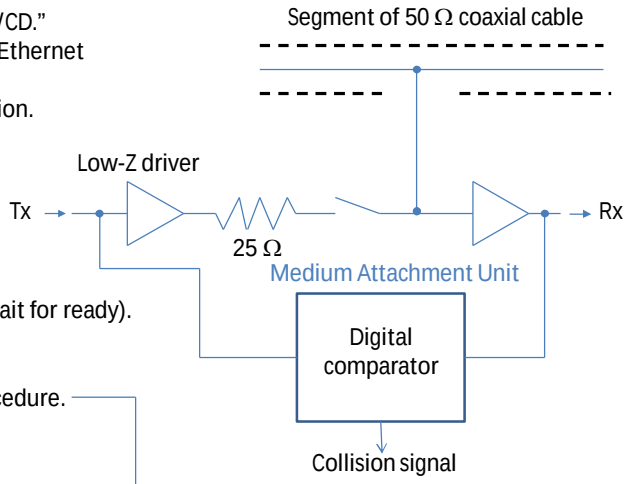
http://www.digibarn.com/collections/diagrams/ethernet/

11

Ethernet—an example of a synchronous serial network

The original Ethernet. Used Manchester code and "CSMA/CD."
officially named "10 Base 5" aka Thickernet, Garden Hose Ethernet

CSMA/CD = Carrier sense, multiple access/collision detection.
It was adapted from Aloha Net (University of Hawaii)



Main procedure:

- Is my frame ready for transmission? If not, ask again (wait for ready).
- Is medium idle? If not, wait until it becomes idle
- Start transmitting.
- Did a collision occur? If so, go to collision detected procedure.
- Reset collision counter and end frame transmission.

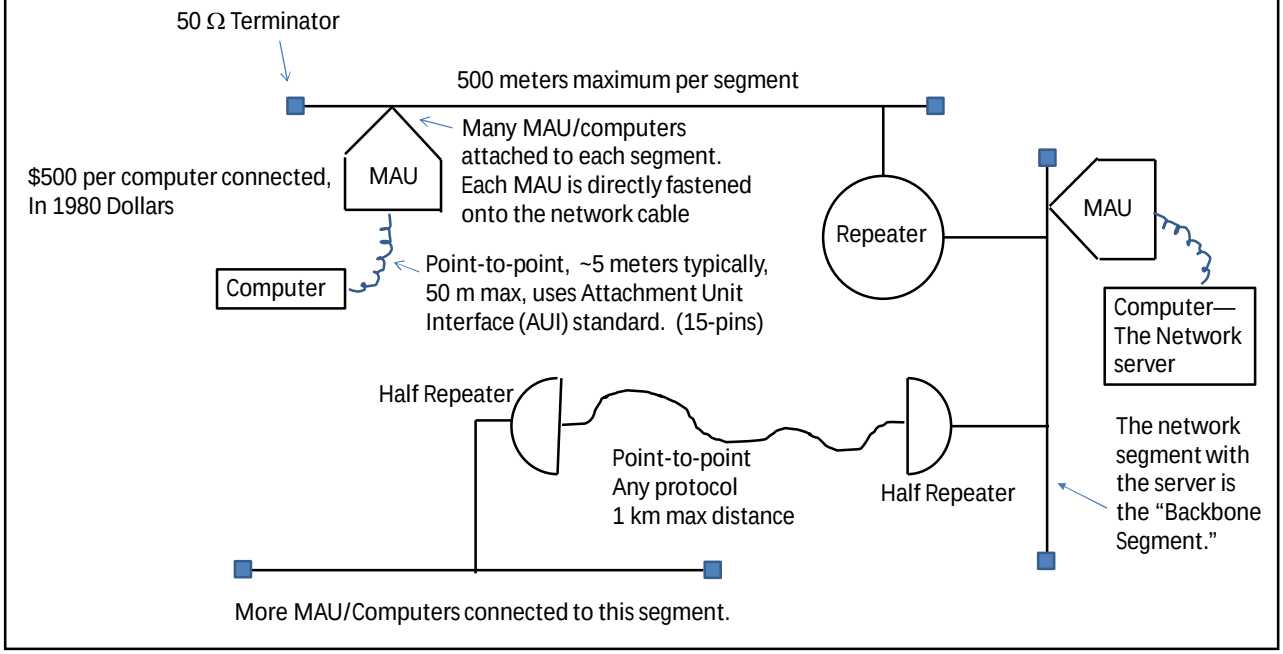
Collision detected procedure:

- Continue transmission with a jam signal instead of a normal frame
- Increment collision counter.
- If too many attempts, abort transmission—report network failure.
- Calculate and wait random back off period based on number of collisions.
- Re-enter main procedure at idle detection stage.

Medium attachment unit transmits and receives simultaneously.
If a mismatch, a collision is signaled.

12

10Base-5 Ethernet, "Thick Ethernet" "Garden Hose Ethernet" "Original Ethernet"



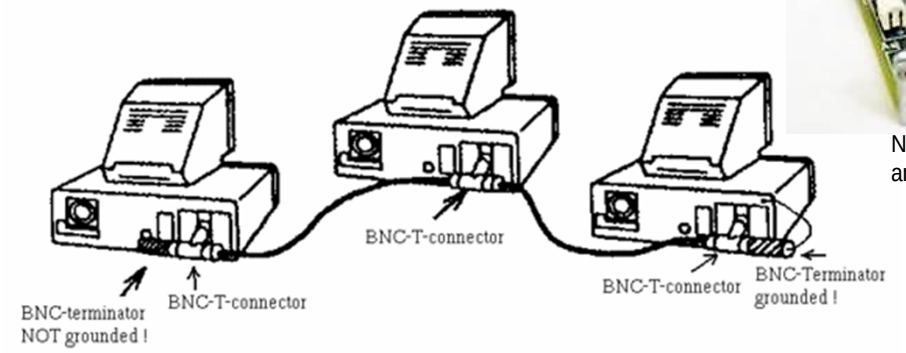
13

Ethernet—the original way

14

Ethernet—an example of a synchronous serial network

The first simplification of Ethernet,
officially named "10 Base 2" aka Thinernet, Cheapernet
10 Mbits/sec, same as 10Base5, very fast for the era.
Used Manchester code and CSMA/CD



NIC with AUI (connects to a MAU) and thin Ethernet connectors.

BNC tee—required on the back of the NIC

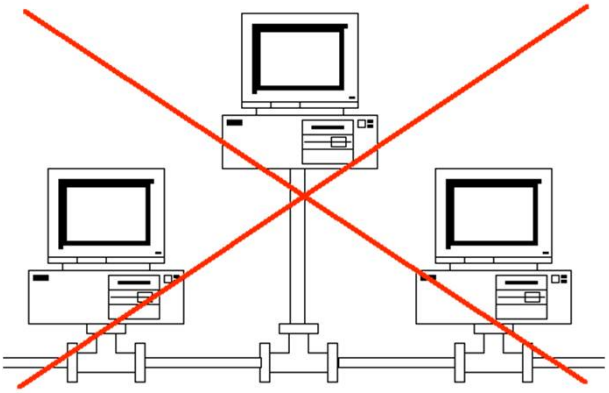
Total length of a segment limited to "200 meters" (actually 185 m)
185 m/30 computers = ~6.2 m between each computer.
The limited segment length limits the number of computers on the network.

http://www.windowsnetworking.com/articles-tutorials/netgeneral/thin.html
https://www.amazon.com/VideoSecu-Adapter-Connector-Splitter-BNC28/dp/B0002ZPFHG

15

Ethernet—an example of a synchronous serial network

The first simplification of Ethernet,
officially named "10 Base 2" aka Thinernet, Cheapernet



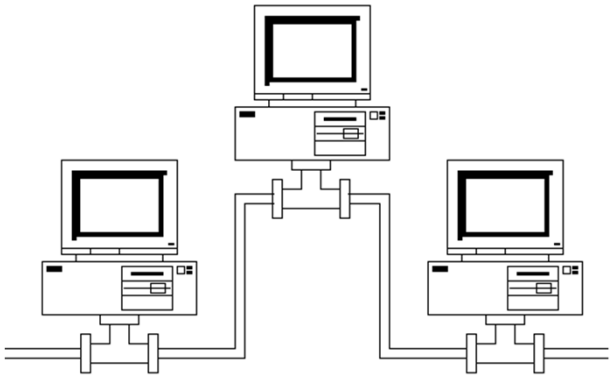
Maintaining correct network topology was difficult since users think a wire is a wire is a wire. The stub shown above causes echoes which create collisions.

http://www.windowsnetworking.com/articles-tutorials/netgeneral/thin.html

16

Ethernet—an example of a synchronous serial network

The first simplification of Ethernet,
officially named "10 Base 2" aka Thinernet, Cheapernet



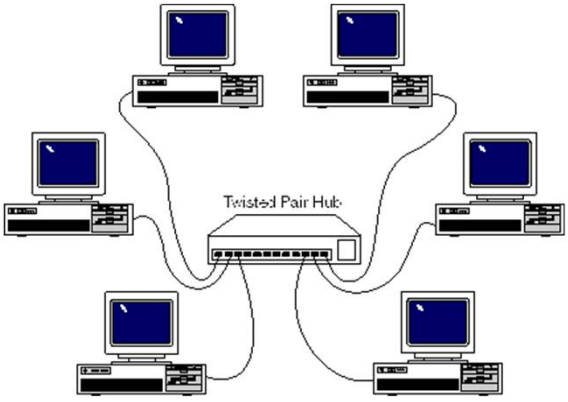
This is the correct way to connect the computers, but the two cables running side-by-side seems wasteful to most people.

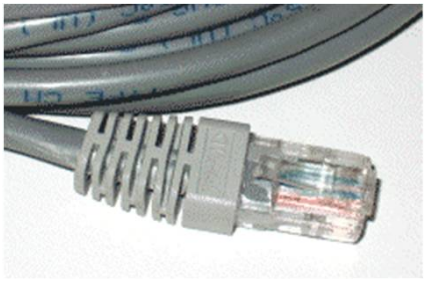
http://www.windowsnetworking.com/articles-tutorials/netgeneral/thin.html

17

Ethernet—an example of a synchronous serial network

The second simplification of Ethernet,
Officially named "10 Base T" aka Twisted Pair Ethernet, Cat-3 Ethernet.





Took advantage of analog "Plain old telephone service" wiring.
Many commercial buildings were loaded with extra "cat-3" wiring when built.
This was the technology that eclipsed all other local area networks.
Still 10 Mbps, Manchester, CSMA/CD except added buffering could avoid collisions

Hub can be replaced with a switch
New: Build an address table on the fly by monitoring source and destination addresses. Gives administrative control over connections.

https://pattito0.wordpress.com/

18