DCF255

Lecture 4 | Data Link Layer

Agenda

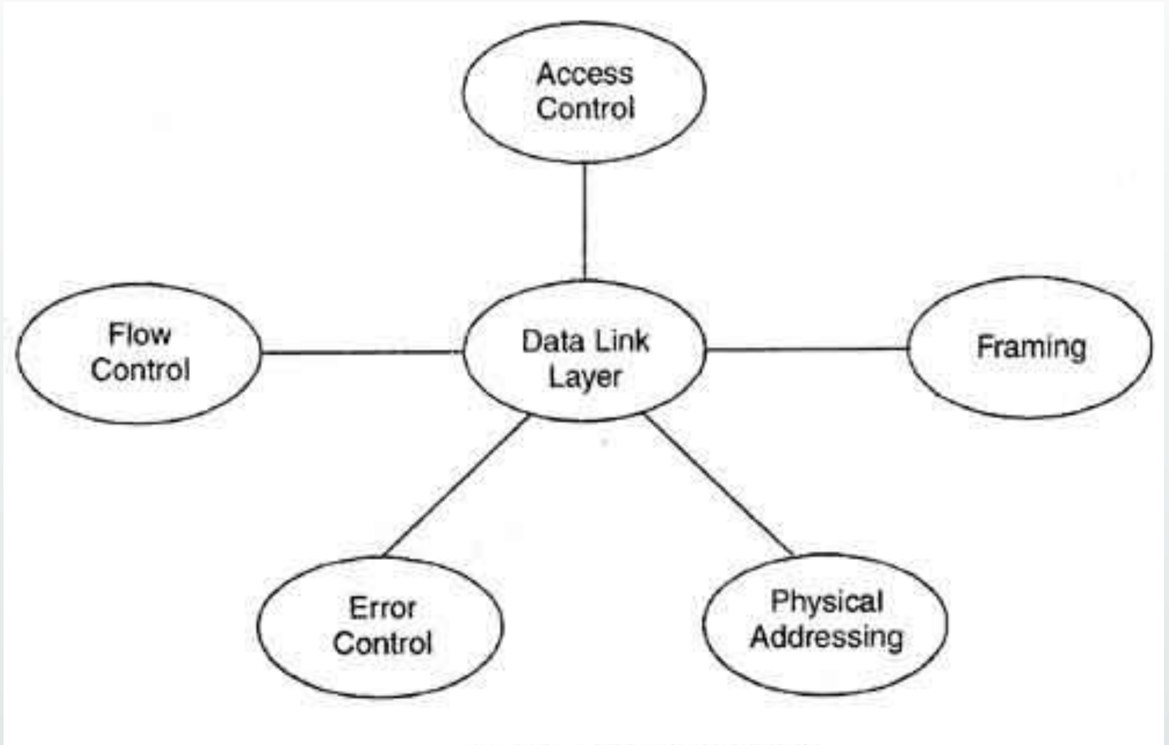
- Data Link Layer Functions
- Data Link Layer: Seneca Case Study
- Synchronization Problem
- Asynchronous and Synchronous Communications
- Data Link Layer Programming
- Transmission Errors
  - Error Detection
  - Error Correction

Data Link Layer

Functions

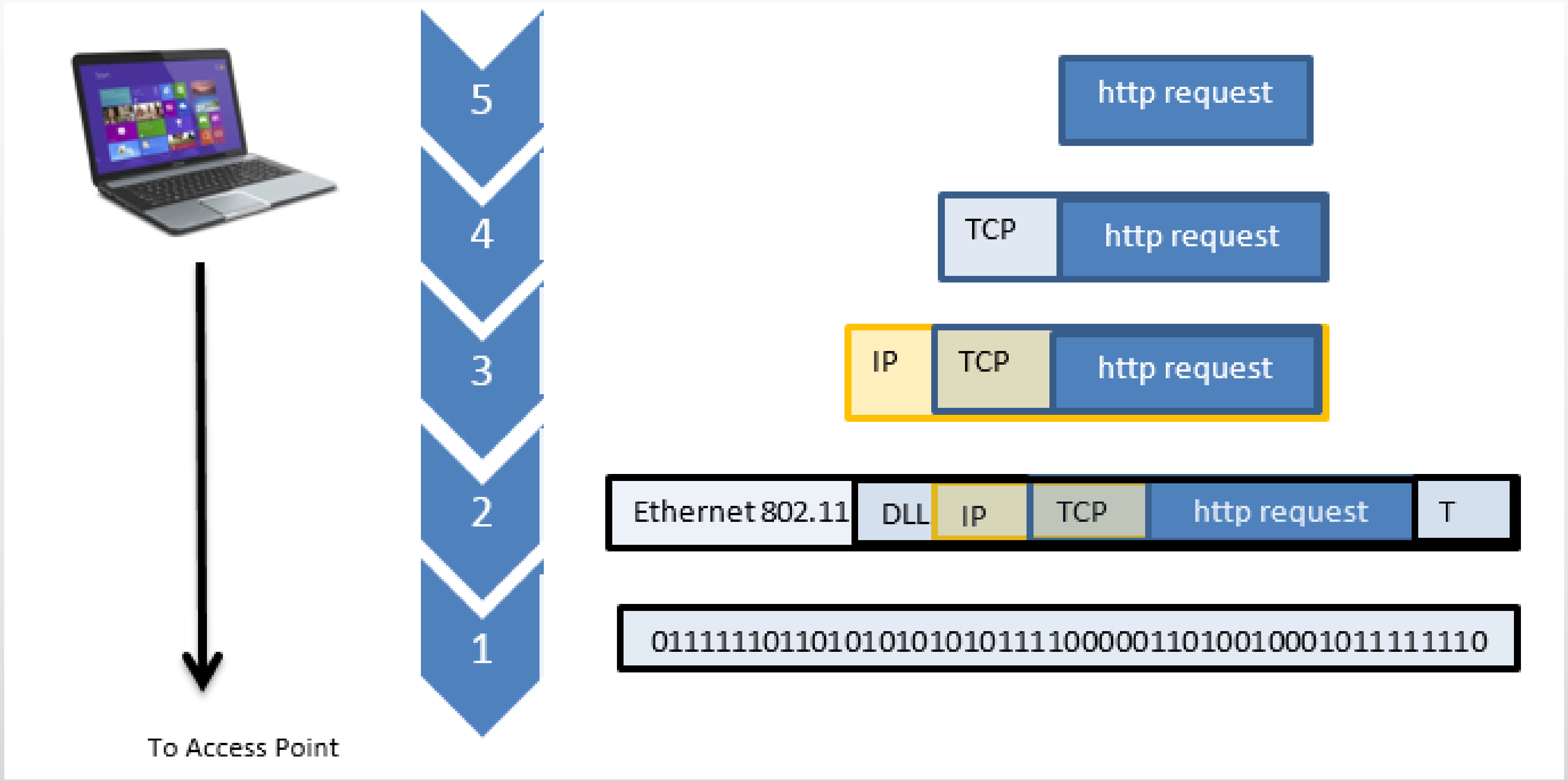
Data Link Layer Functions

1. Data Framing
2. Media Access Control (MAC) –not used with switched Ethernet
3. Physical Addressing.
4. Flow Control
5. Error Control

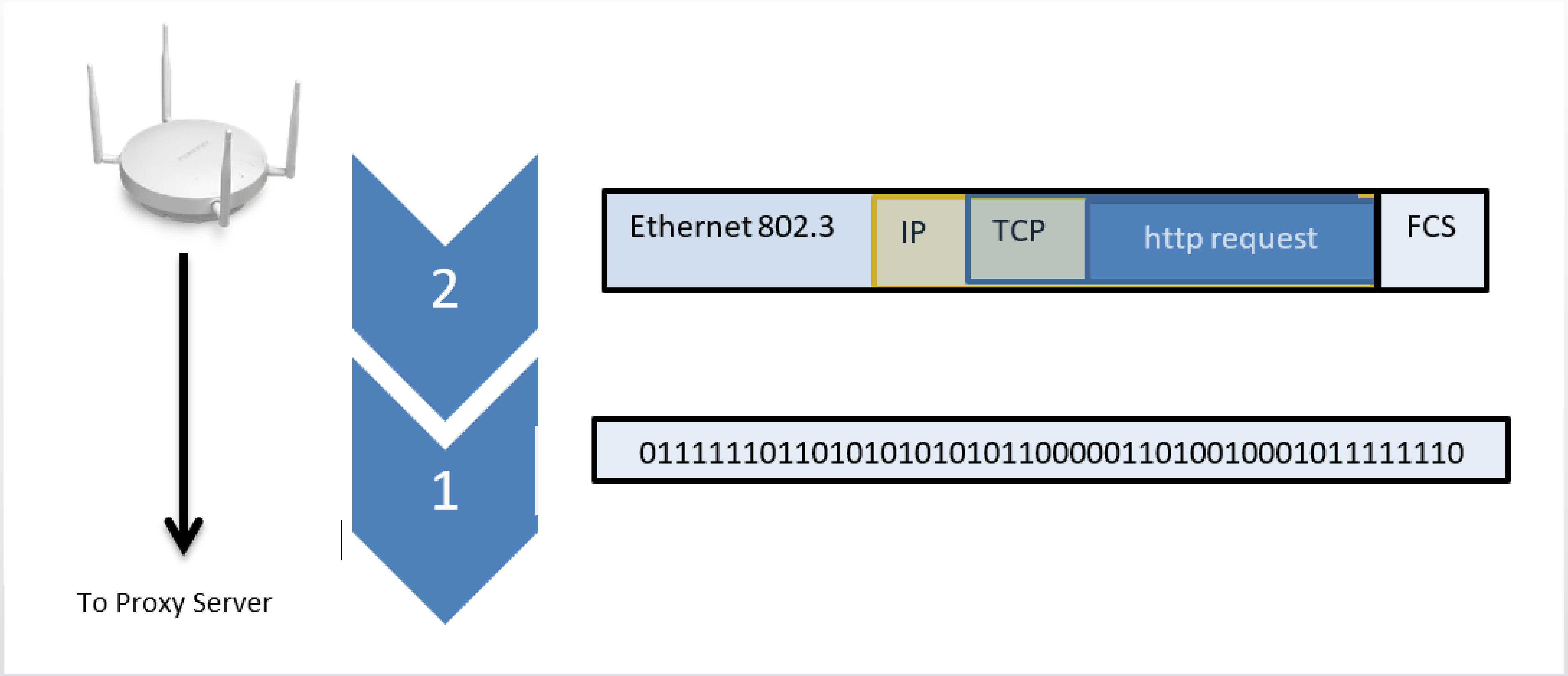


Functions of data link layer

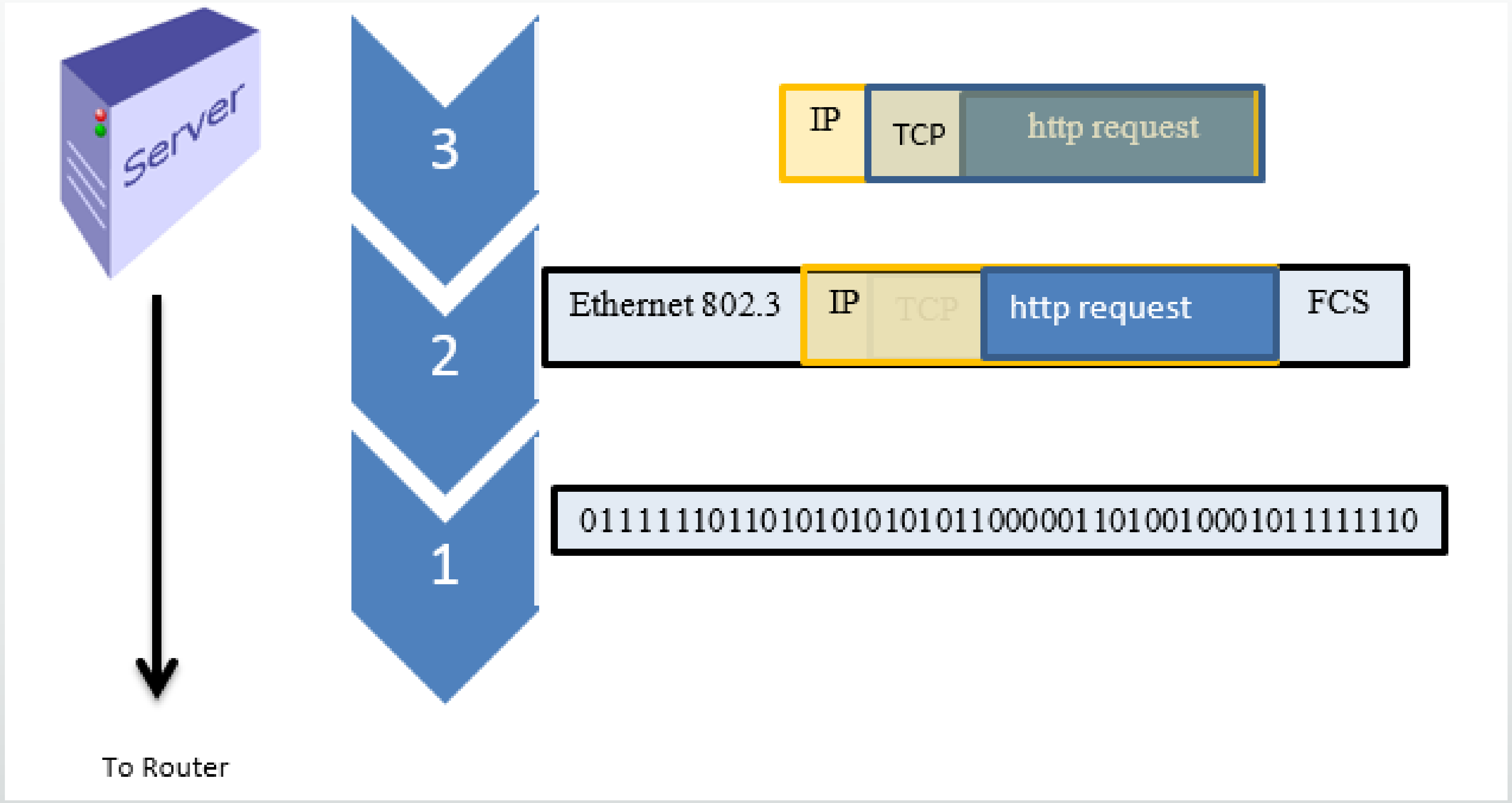
Data Link Layer: A Seneca Case Study



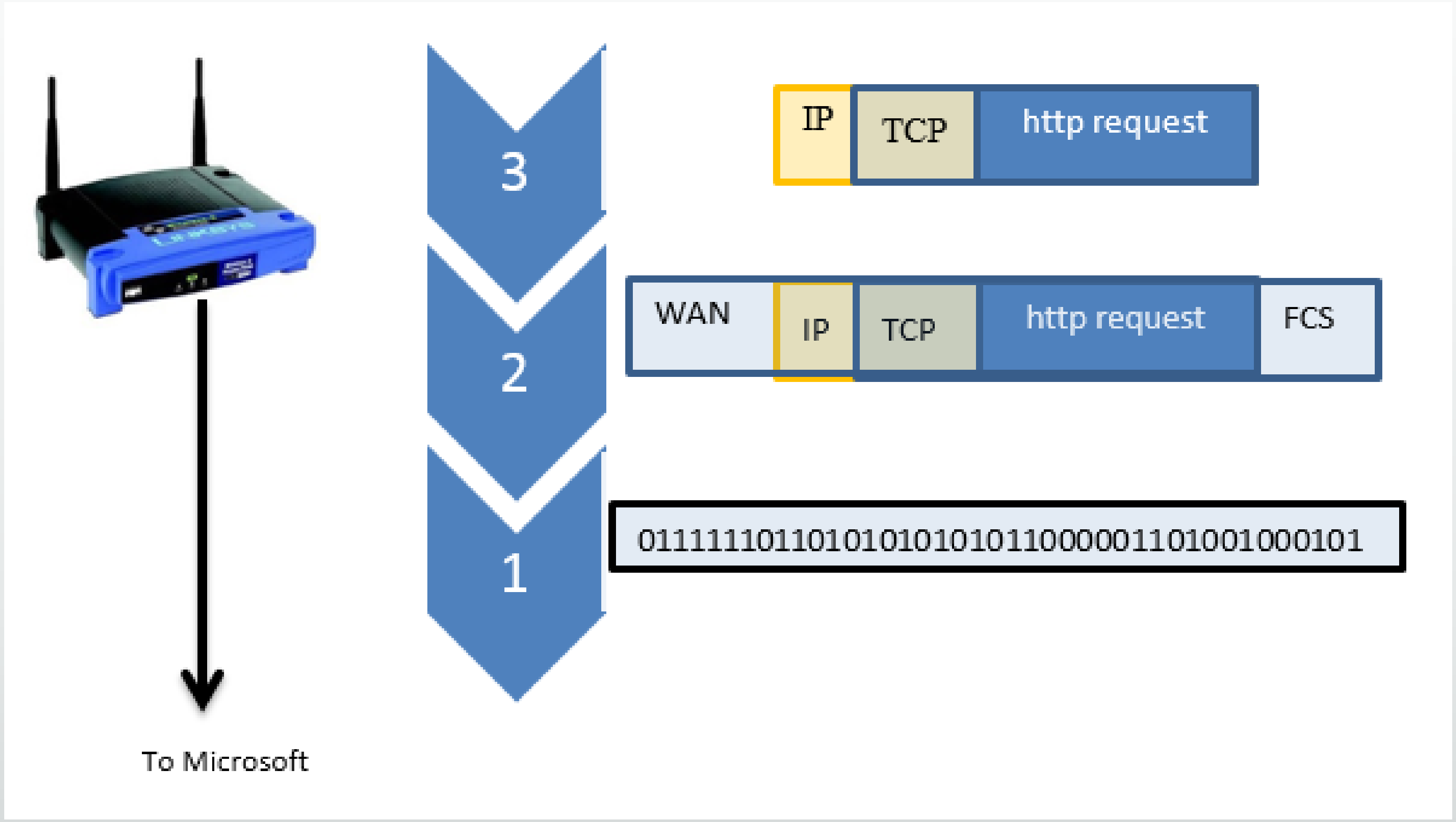
Data Link Layer: A Seneca Case Study



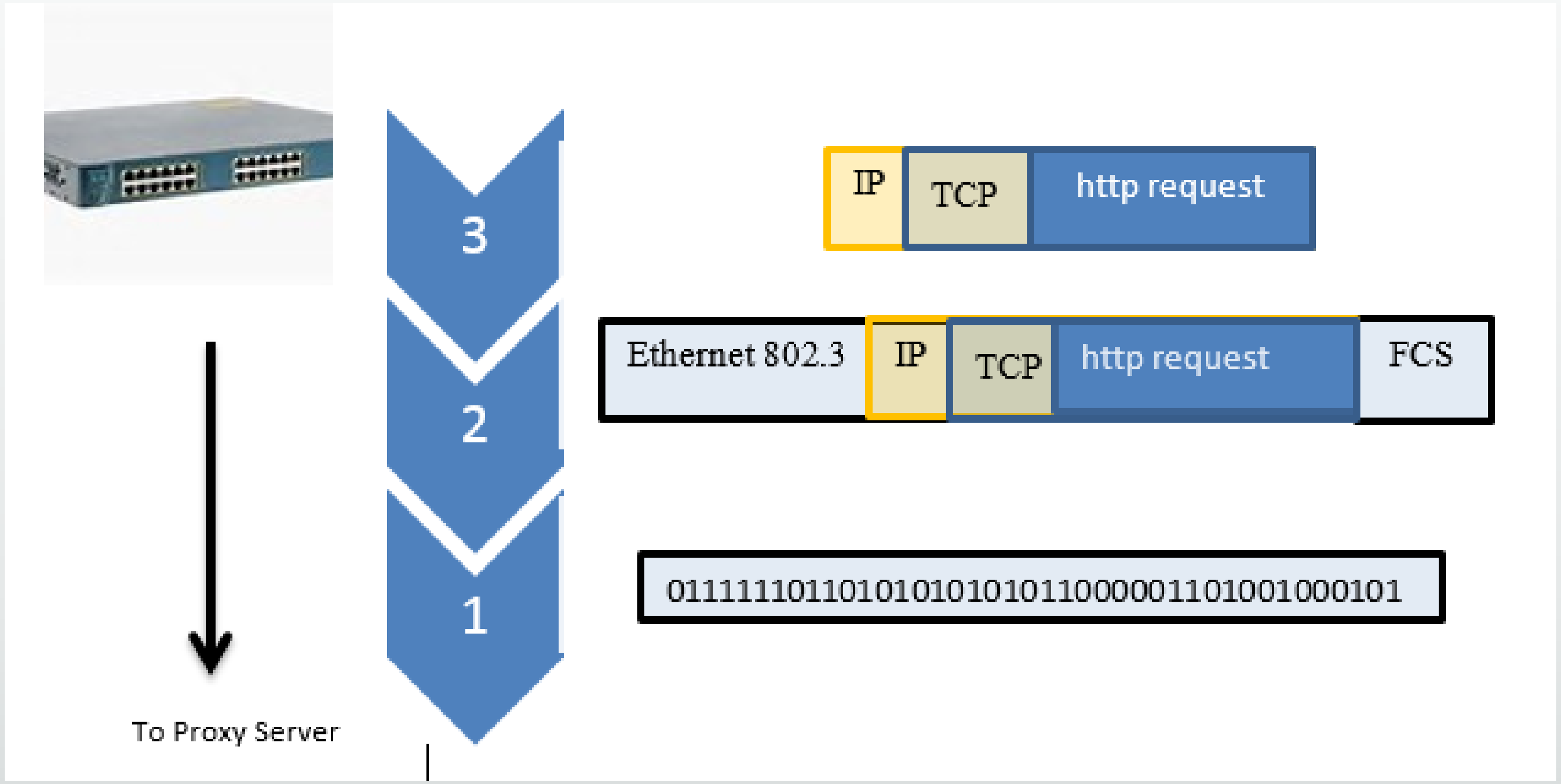
Data Link Layer: A Seneca Case Study



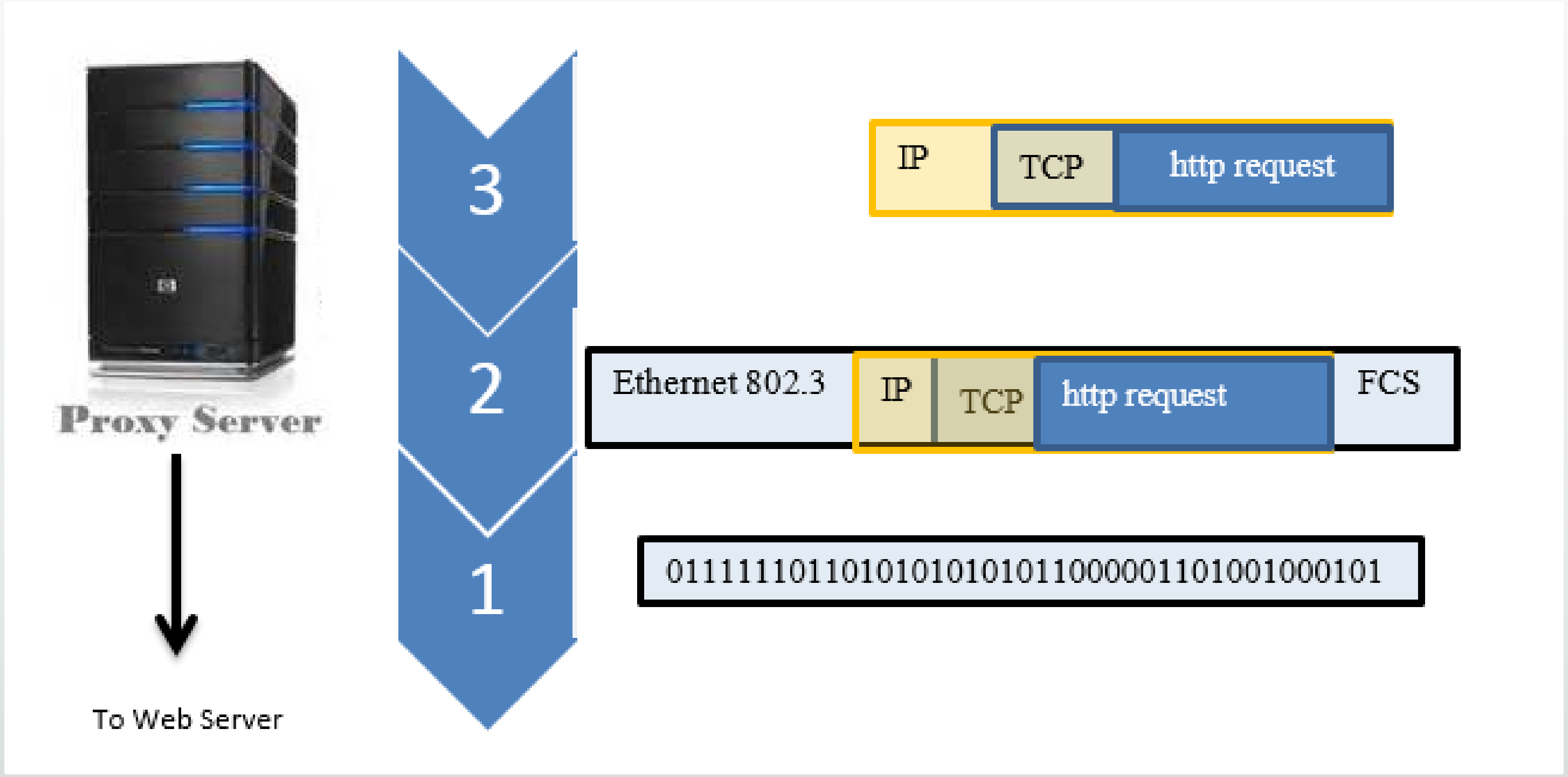
Data Link Layer: A Seneca Case Study



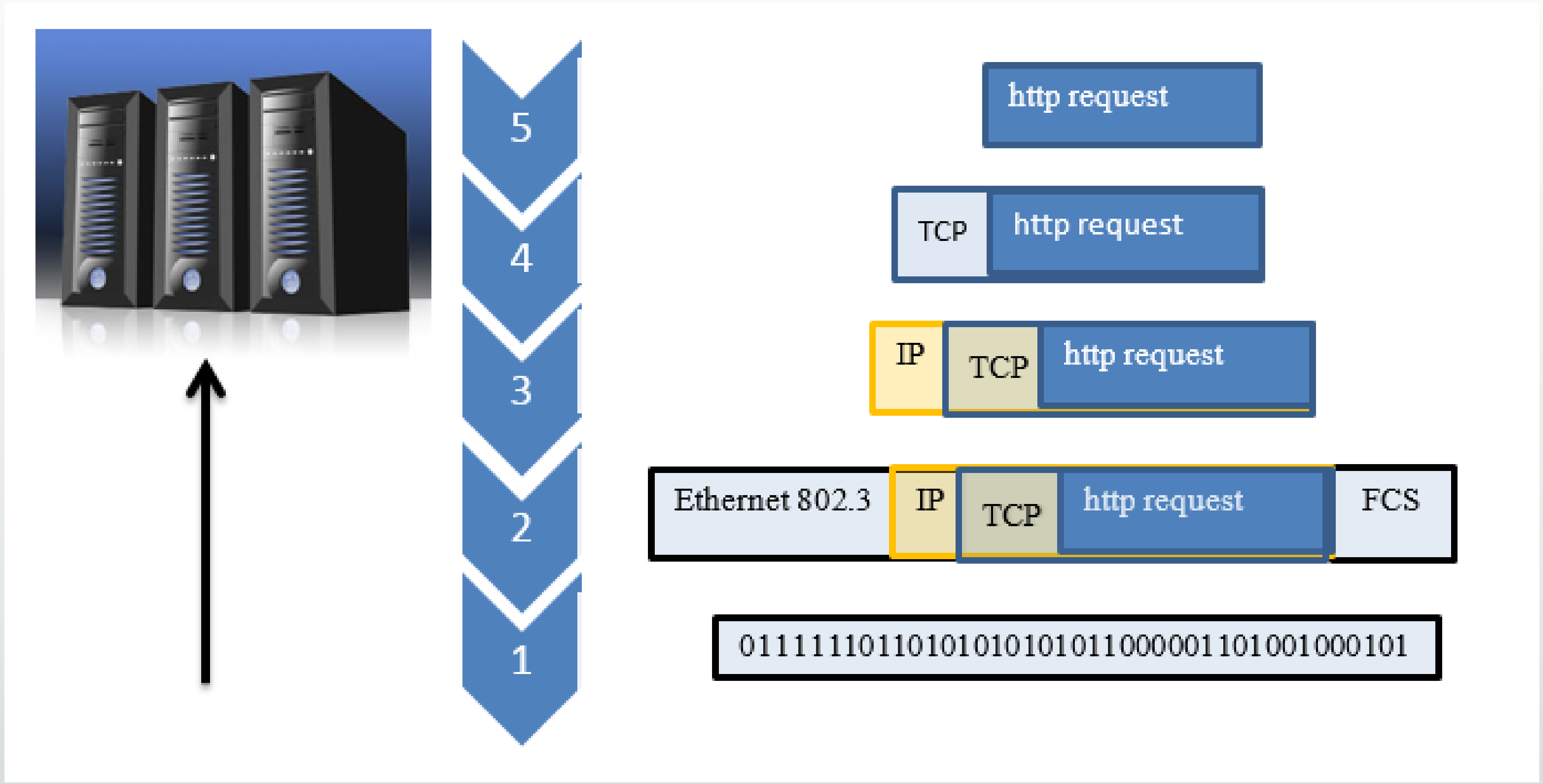
Data Link Layer: A Seneca Case Study



Data Link Layer: A Seneca Case Study



Data Link Layer: A Seneca Case Study



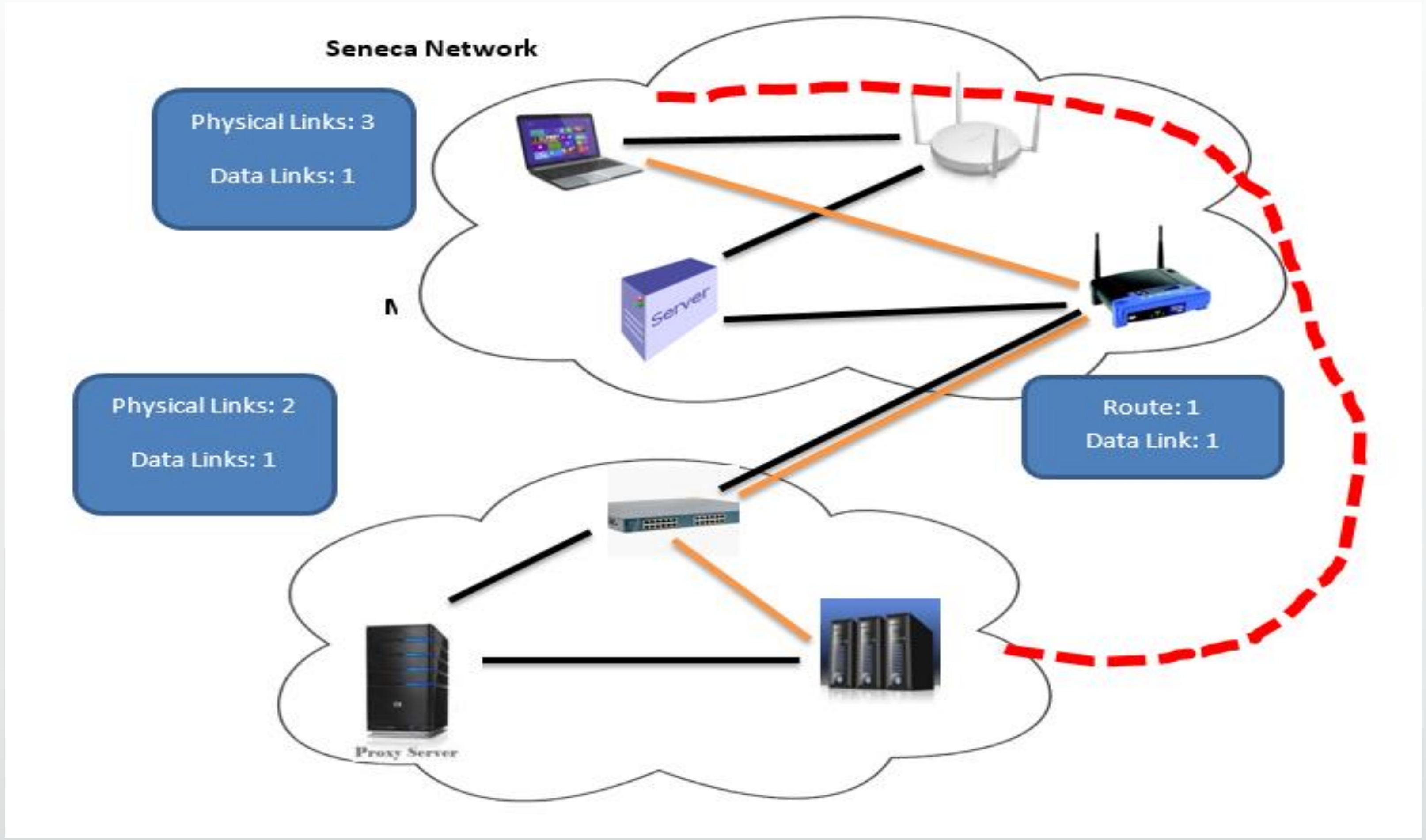
Data Link Layer: A Seneca Case Study

1. The Data Link layer is responsible for framing and creating the LAN/WAN headers and trailers so that the frame can travel across single switched networks..

2. We think there is a direct connection between sending and receiving host suing the IP address, but this connection is virtual or logical, the message actually moves down 5 layers of the protocol stack, across each physical link, link by link. and up 5 layers of the stack

3. Two addresses are required because all LAN/WAN technologies use the MAC\hardware or similar address to move across a data link. IP routing, the Internet layer, is built on top of switching to provide a globally unique address.

Data Link Layer: A Seneca Case Study



Total 6 physical links (black)

Total 3 Data Links (orange)

Total 1 Route (red)

Synchronization Problem

Synchronization Problem

- Remote computers have
  - "propagation delay"
    - the time for one bit to travel from sending to receiving computer.
  - Latency
    - The time to process a bit across a link
    - Delay caused by congestion

- How can we keep remote computers in synch?

Three Solutions to Solve Synchronization Problem

1. Run a separate clocking wire to join all the computers and synchronize them.
   - Expensive and impractical when there are great distances
2. Use a “self-clocking” encoding system like Manchester forcing the sending and receiving computer to reset its clock 2 times for every bit.
   - Impractical with speeds above 10 Mbps
   - NRZI more efficient when 1 followed by a 0 2 bits are sent per clock cycle
3. Send data asynchronously so that timing is less of an issue. Used internally in PC

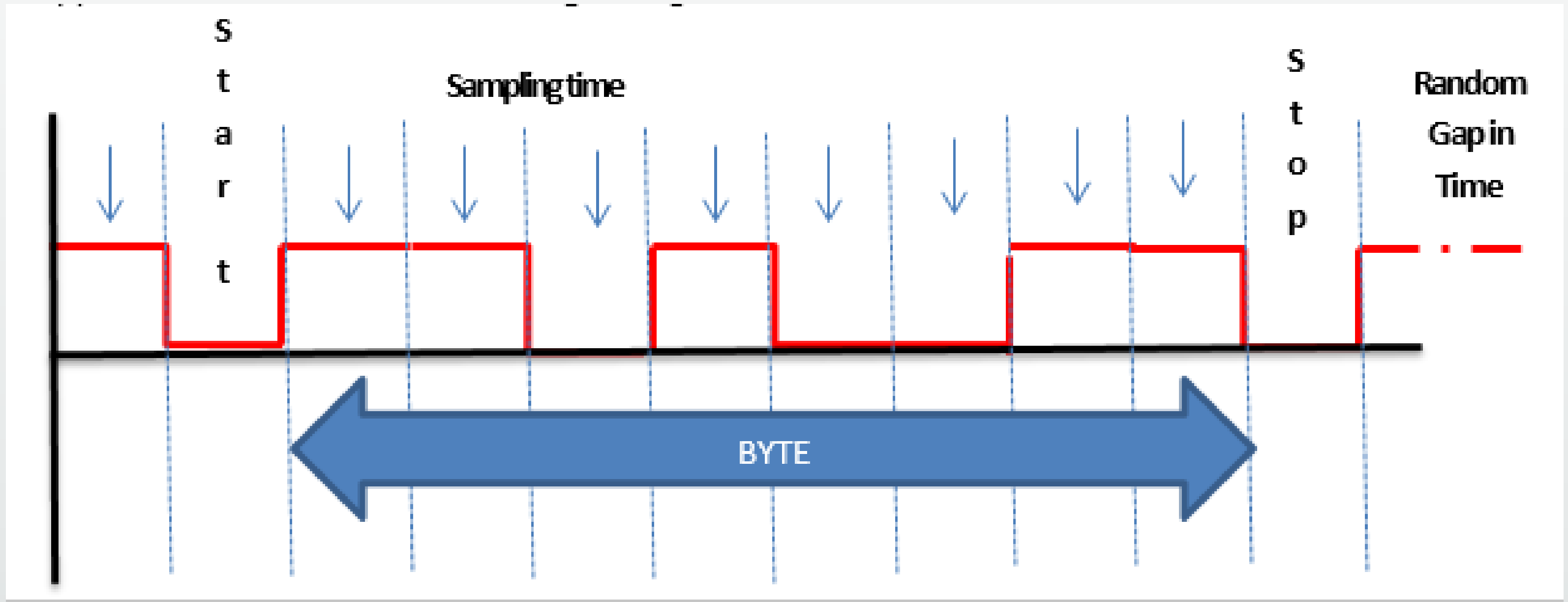
Asynchronous Transmission

1. Drop in voltage indicates the start bit of a transmission.

2. Followed by 8 bits of data. With only 8 bits being sent. Jitter is minimized

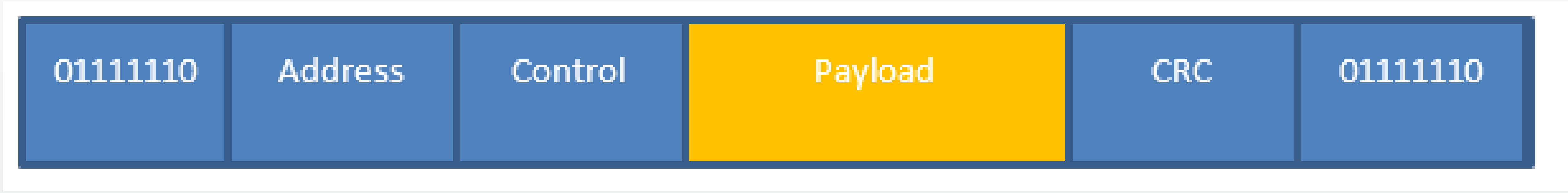
3. A stop bit signals the end of the transmission

4. Sending computer waits a random gap in time and then repeats process



Problem: Sending 2 bits for every 8 bits is 25% overhead which does not provide enough throughput for large applications, like databases

Synchronous Transmission



1. Large block of data – Ethernet 1500 bytes. -- No shown is 7 byes of zeros and ones which is used to synchronize the devices before the start flag
2. Followed by a start of transmission flag of 01111110 (126 or 0x 7E) . The same flag is used for end of transmission
3. The address field is the MAC address of the sending and receiving hosts
4. The control field is one or more bytes and contains information about the frame
5. The Payload is the data carried and is much bigger than shown here
6. The CRC is a 2 byte field that shows if any errors occurred in transmission

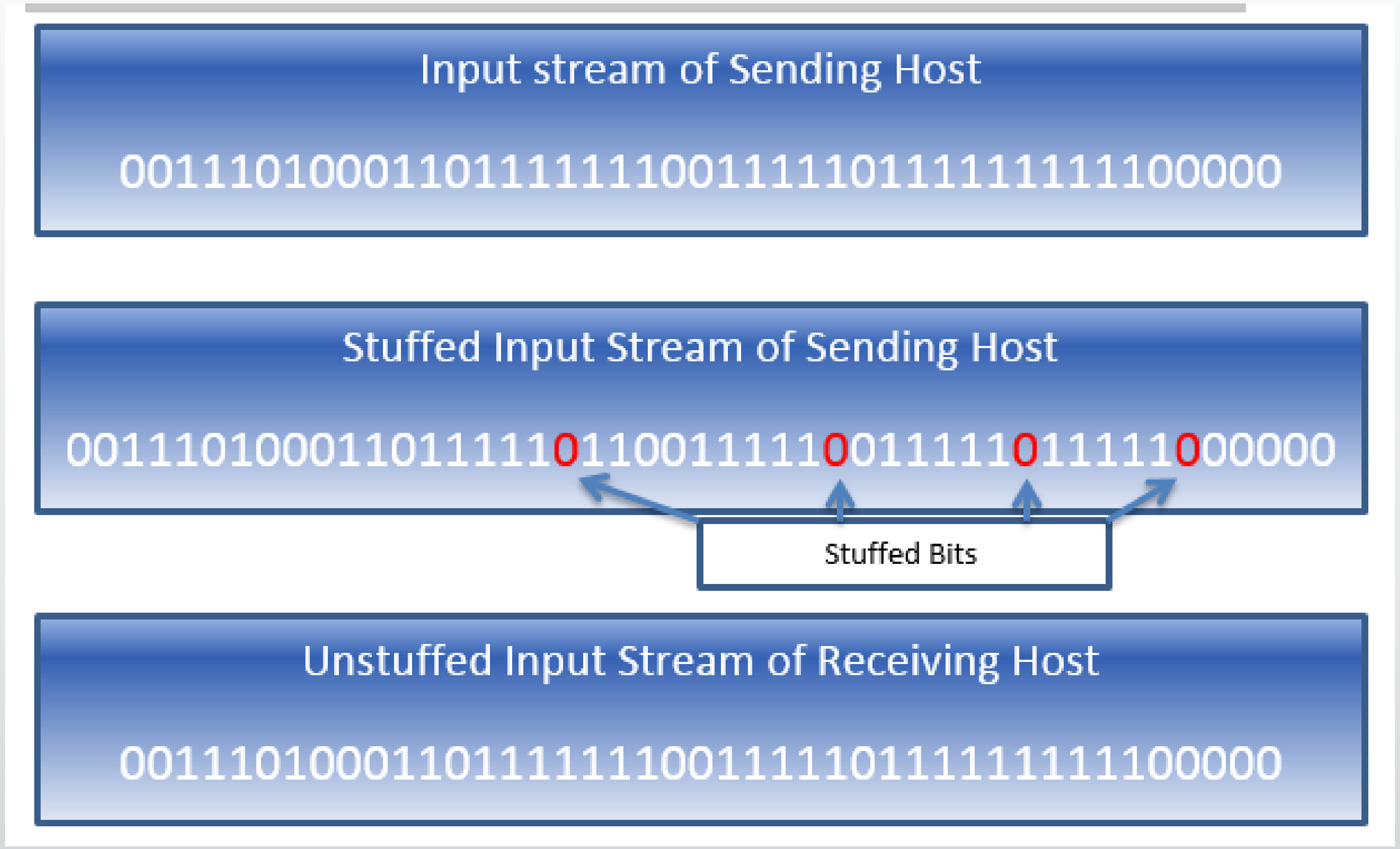
Summary Table

Type of Transmission	Advantages	Disadvantages
Asynchronous	Simple, uses less hardware and programming. Equipment less expensive.	Low throughput because of high overhead and slower speed
Synchronous	High throughput because of less overhead and larger frame size	Requires more hardware and programming. Equipment more expensive

Data Link Layer Programming

- Synchronous transmissions are the norm for the data link layer
- Start and Ending flag of decimal value 126 or 7E in hexadecimal “01111110”
- The flags are essential to synchronous transmissions telling the receving computer when a transmission starts and ends.
- What happens if this value of 7E appears in the payload or address portion of the transmission? The receiving computer could mistakenly interpret the value as the ending flag.
- To avoid this probability (57%) DLL has a built in subroutine called “bit stuffing”

Bit Stuffing


Input stream of Sending Host

0011101000110111111100111110111111111100000

Stuffed Input Stream of Sending Host

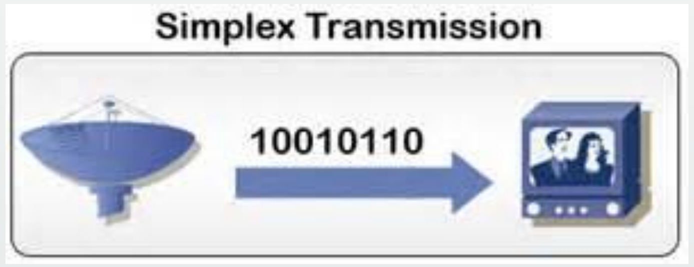
00111010001101111101100111110001111101111110000000

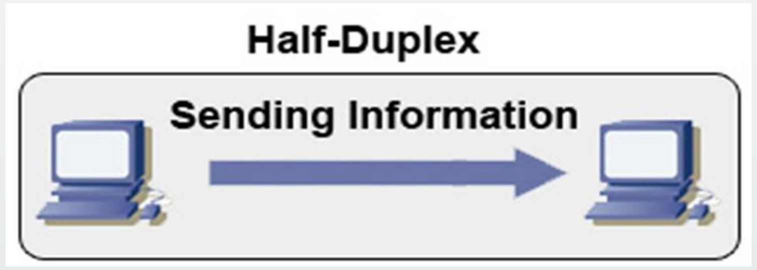
Stuffed Bits

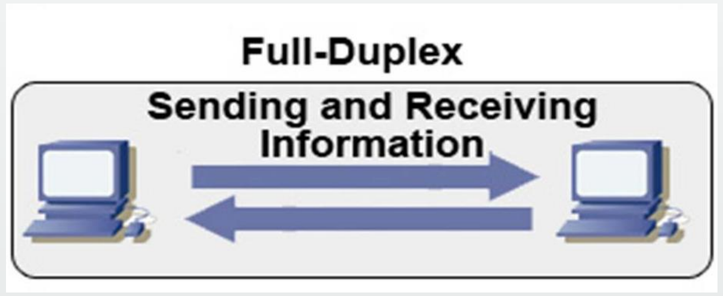
Unstuffed Input Stream of Receiving Host

0011101000110111111100111110111111111100000

Simplex Half Duplex Full Duplex







One Way Communication only

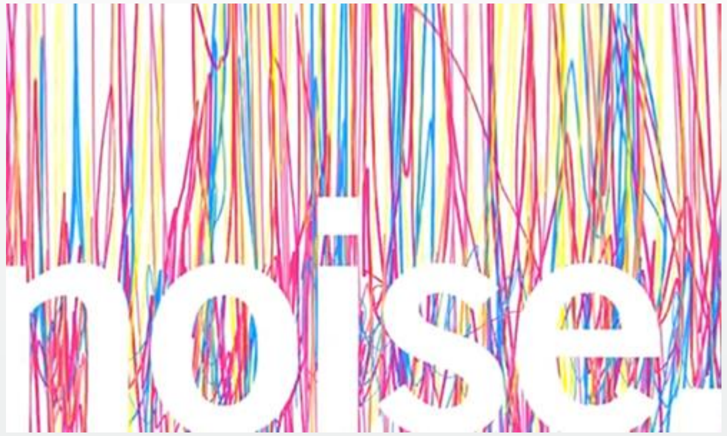
Two Way Communication, but not at the same time

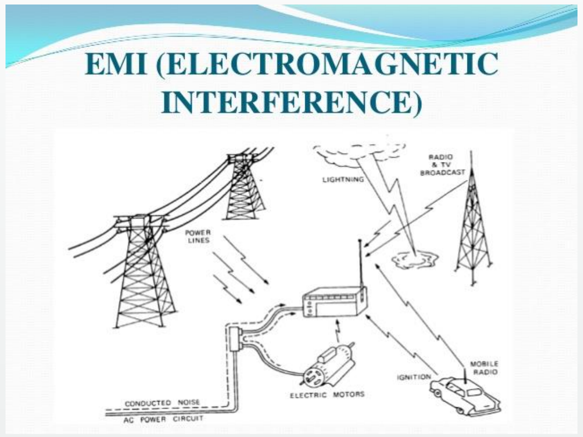
Two Way Communication, at the same time

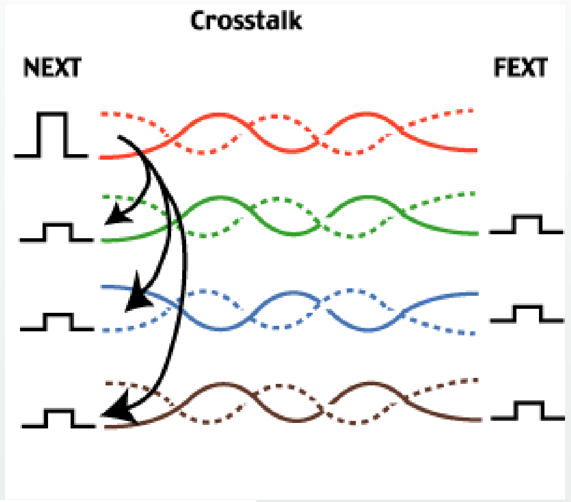
Transmission Errors

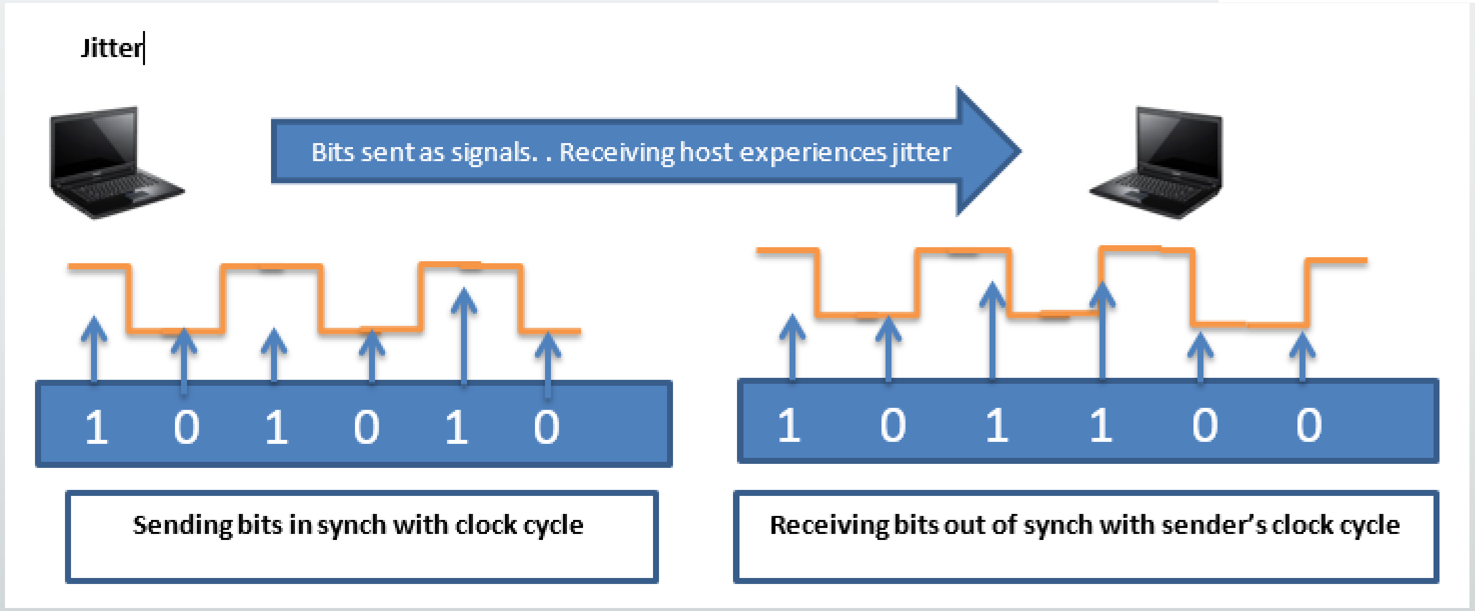
Noise EMI CrossTalk Jitter

Common Transmission Errors





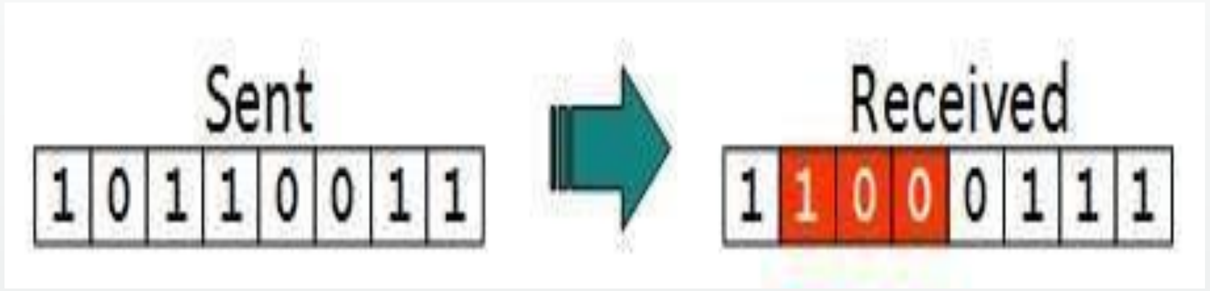



Jitter
Bits sent as signals. . Receiving host experiences jitter
1 0 1 0 1 0
1 0 1 1 0 0
Sending bits in synch with clock cycle
Receiving bits out of synch with sender's clock cycle

Error Control

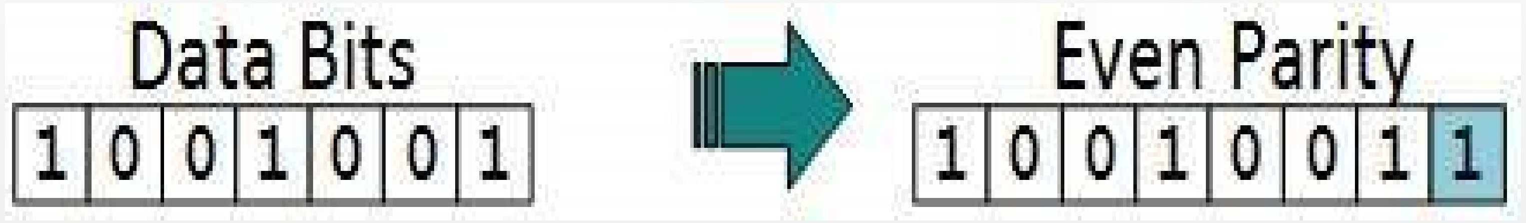
Detection and Correction

Error Detection



- Parity
  - Even or Odd (Even most common)
- CRC – Cyclical Redundancy Check
  - Based on Polynomial arithmetic

Parity

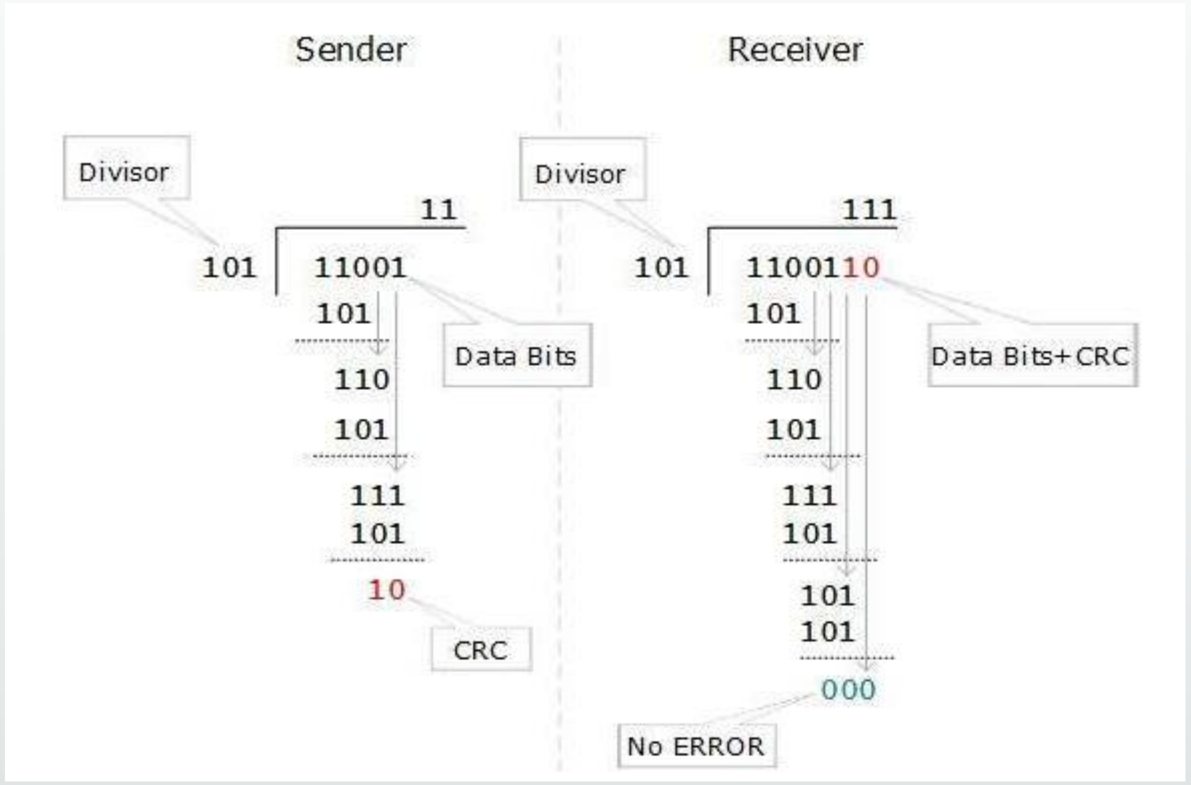


- Even parity
  - Count the number of "1"
  - If count is ODD add a one to make the count even

If 2 bits are flipped the parity bit will not detect the error . This is why there are additional check points at the transport and application layers

CRC – Cyclical Redundancy Check

- Uses Binary Division combined with Polynomials
- Sender performs a division on the bits being sent and calculates the remainder
- The CRC check bits are added to the end of the data bits
- The receiver performs a division operation using the same CRC divisor
- The remainder should be "o"



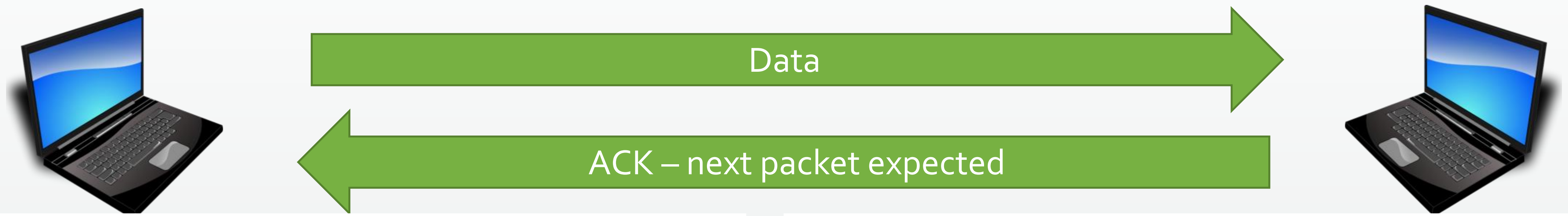
CRC – very fast built into the hardware, overhead is very low and detects 99.99% of all errors

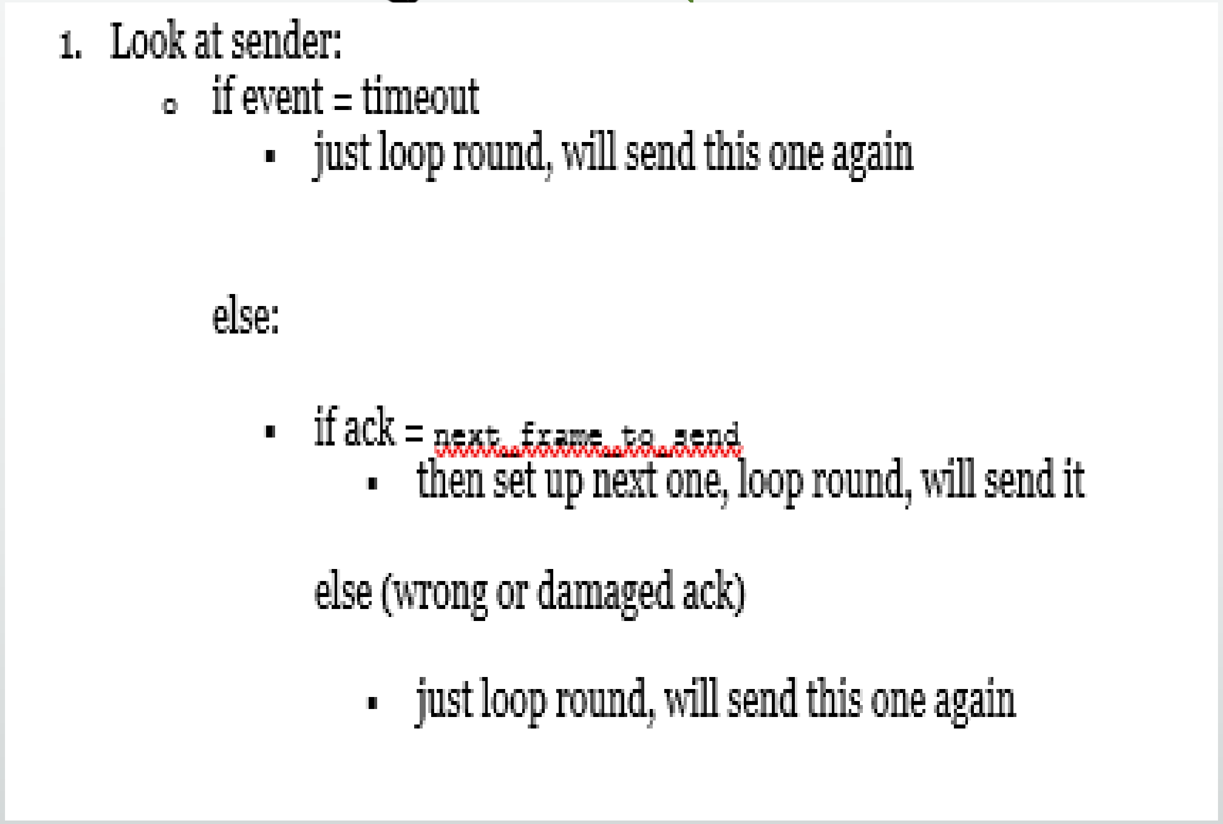
Error Correction

- Backward Correction
  - Send a message to the sender to retransmit the corrupted data

- Forward Correction
  - Includes parity bits and redundant bits so the receiver can not only detect the error but fix the error “on the fly”

Backward Correction





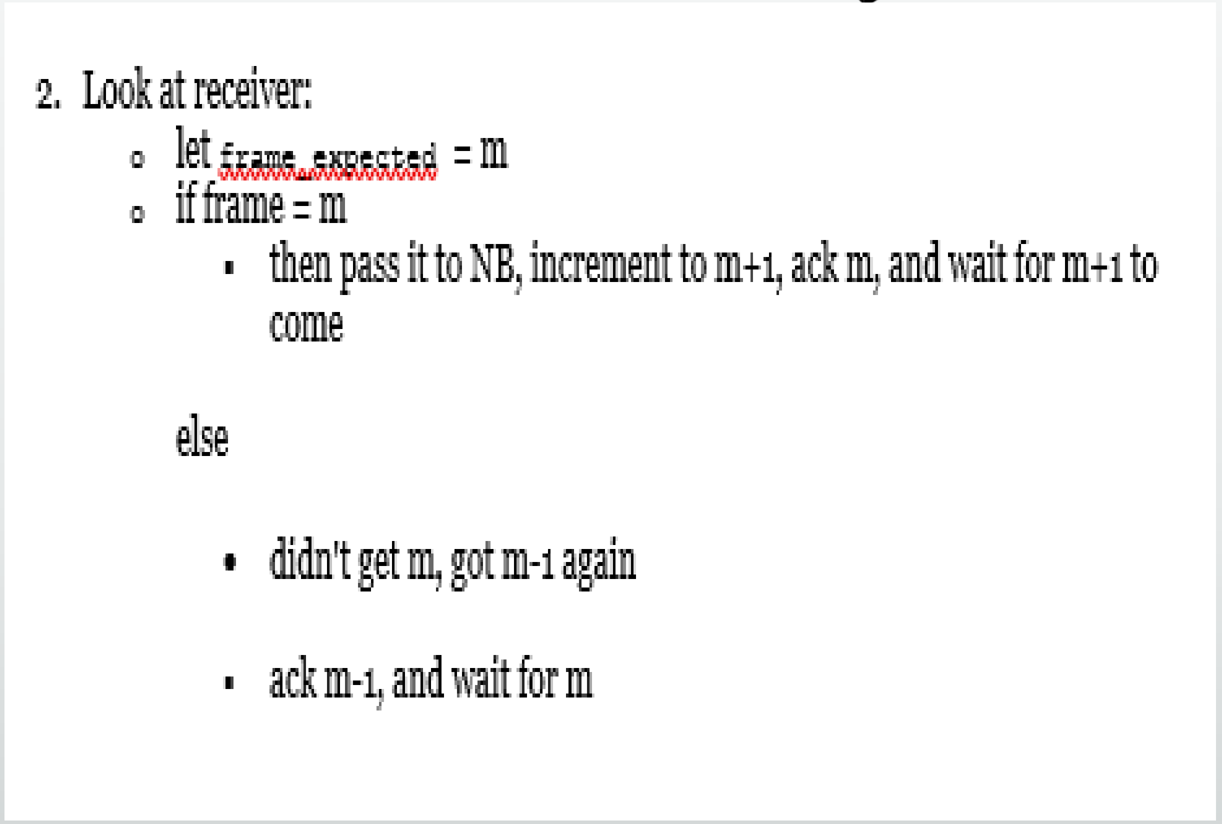
1. Look at sender:
   - if event = timeout
     - just loop round, will send this one again

   else:

     - if ack = next_frame_to_send
       - then set up next one, loop round, will send it

     else (wrong or damaged ack)

       - just loop round, will send this one again



2. Look at receiver:
   - let frame_expected = m
   - if frame = m
     - then pass it to NB, increment to m+1, ack m, and wait for m+1 to come

   else

     - didn't get m, got m-1 again

     - ack m-1, and wait for m

Forward Correction

- Used in real time content VoIP, IPTv where it is not practical to send an ACK packet and wait for a retransmission
- Hamming codes are redundant bits added to the data stream combined with parity so that the receive can not only detect errors, but can correct the bad bits on the fly
- To learn more about how hamming codes work – click on the button to the right.

YouTube Hamming Codes

Summary

1. The Data Link layer is responsible for framing data by adding and removing LAN/WAN headers required for the next link. Data is sent link by link across single switched networks. Routing is built on top of switching so that there is a logical connection between the IP address of the sender and receiver hosts

2. Synchronization is a problem in data transmission. Each computer's clock cycle must be closely aligned to avoid errors. The best method is to connect all devices with a clocking wire, but this is expensive and impractical if devices are separated by great distance.

3. Data communications uses asynchronous and synchronous types of transmissions to correct the synchronization problem. Asynchronous communications use a start and stop bit and only transmit 1 byte at a time. While effective, it is too slow for processor intensive applications. Synchronous transmissions are best but require bit stuffing sub routines as well as low noise and reliable networks to be effective

4. Error detection is done using parity and CRC. Error correction is based on backward correction, such as sending a request to retransmit bad packets and forward correction which uses parity and hamming codes to correct the bad bits at the receiving host.