Data Link Layer

The data link layer (DLL) and the physical layer are closely related. Many text books show the TCP/IP suite as a four-layer model: application, transport, Internet and Network Interface. This approach, however, ignores the fact that layers 1 and 2 are predominantly ISO standards. While the application and Transport layers are predominantly TCP/IP layers, the bottom layers are OSI and have worldwide dominance. Today, for example, when a new LAN technology is developed by the IEEE, it is not officially a standard until passed by the ISO. This is because the bottom 2 layers are largely OSI layers. This leads to the 5-layer hybrid which we are using in this class. The DLL and the physical layer work together to implement data link layer technologies which often specify hardware and encoding requirements. Ethernet is a perfect example; the IEEE 802.3 standard specifies not just how Ethernet works at the data link layer, but also its various physical layers. Network interface cards which work at the DLL are often called "Ethernet cards" There are also network interconnection devices, such as bridges and switches which are often called "layer 2" devices because they make decisions about the data they receive, by looking at data link layer frames. The second layer of the TCP/IP protocol stack is responsible for wired and wireless LAN technologies.

Data Link Layer Functions

1. Data Framing: The data link layer is responsible for the final encapsulation of higher-level messages into frames, so it can be sent across a single switched network by the physical layer. And the synchronization of machines at each end of the link

2. Media Access Control (MAC): On wireless networks or older non-switched Ethernet networks, the DLL is responsible for setting rules to manage devices which share a common medium.

3. Addressing: Each device on a network has a unique number, usually called a hardware address or MAC address. The data link layer is responsible for adding the source and destination MAC addresses to the frame header to ensure proper delivery.

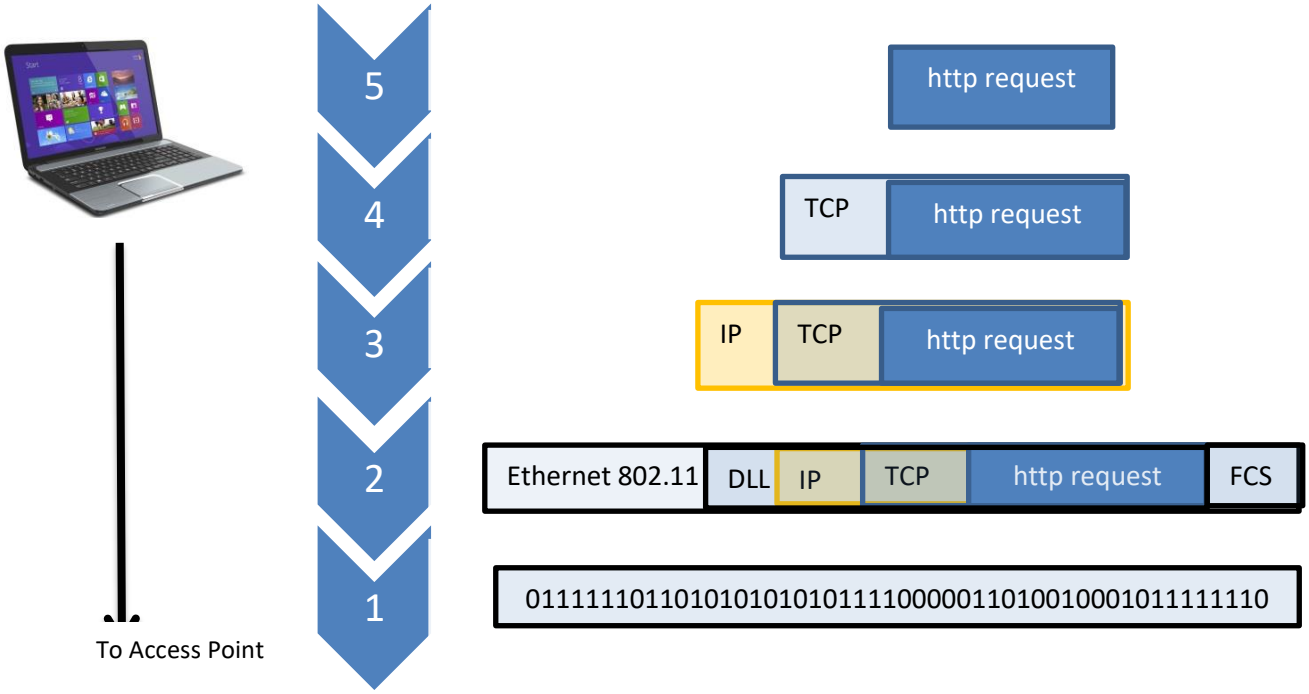
4. Flow Control: Some devices are faster processing in others, so the data link layer can control the speed of sending frames to avoid filling the receiving devices buffer capacity and lose frames.

5. Error Correction: The data link layer handles errors that occur in transmission using a Frame Check Sequence which allows the receiving host to detect if the data was received correctly.

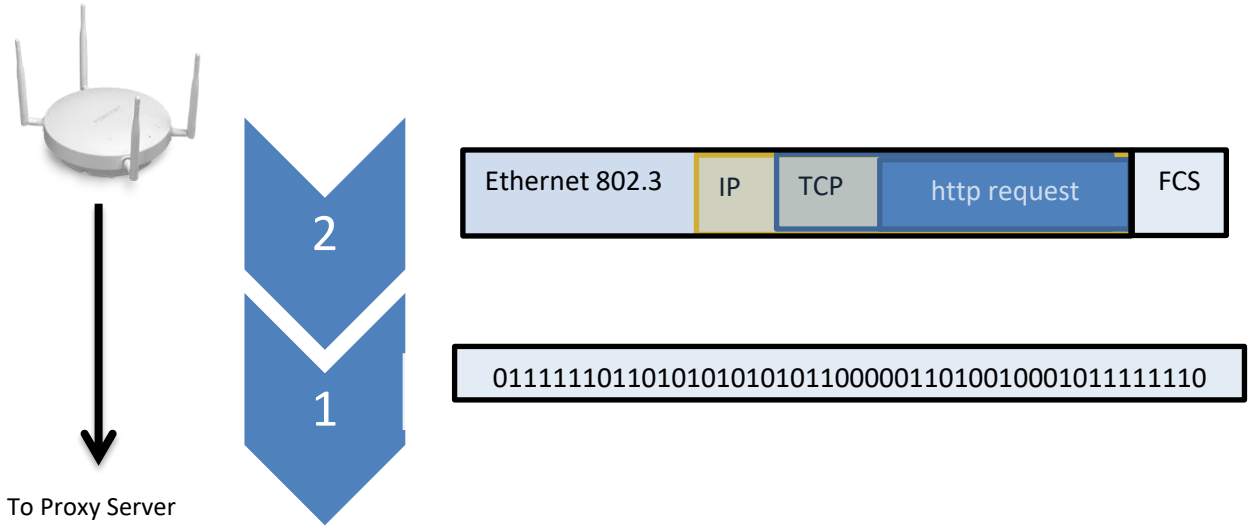
Data Link Layer: A Seneca Case Study

Suppose you are in an open lab and want to access a web page on Microsoft's network. You open your browser and type in the URL, **www.microsoft.com** and press ENTER key. Your request is formatted into an http request at the application layer by your browser. The request is then handed down to the Transport layer which encapsulates the request inside a TCP packet. The TCP header contains protocol information, sequence number and destination and source port information. The packet is passed to the Internet layer which adds the IP address of the host and the destination Microsoft web server. The

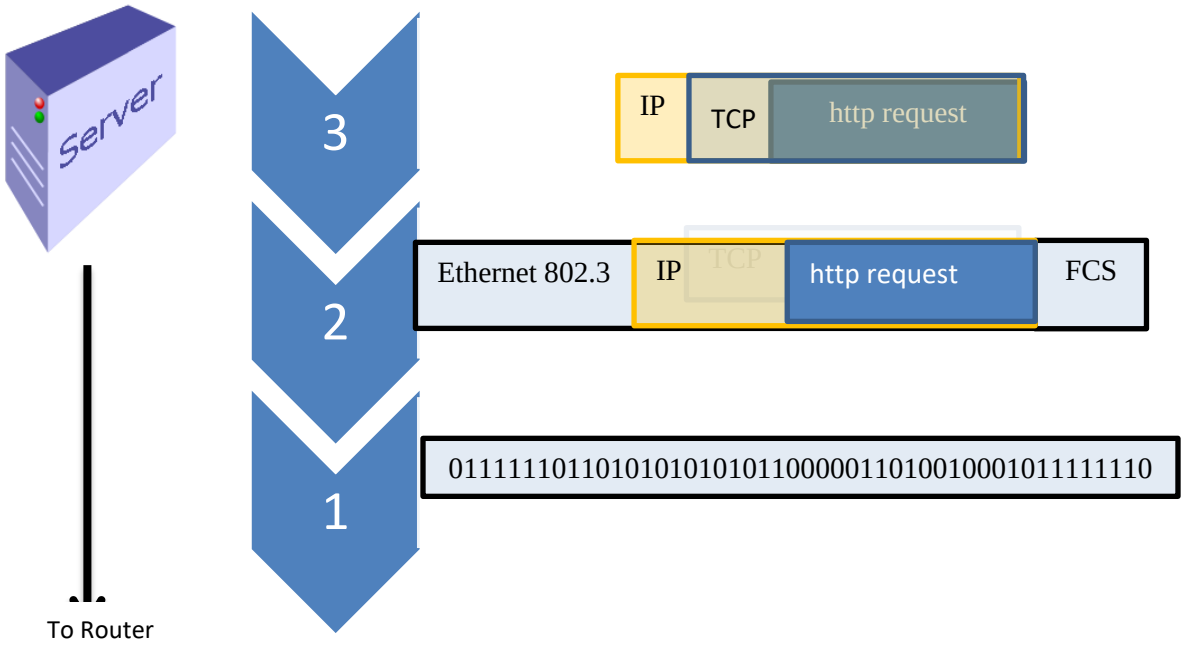
IP header is encapsulated inside an IP datagram, which is then passed to the data link layer. The latter adds a header and trailer to the datagram, creating a DLL frame. The header is an Ethernet 802.11n wireless frame format specifying the Media Access Control (MAC) address of the access point. The trailer is a Frame Check Sequence (FCS) to check for any errors in transmission by the Access point. The physical layer then sends the frame to the Access point



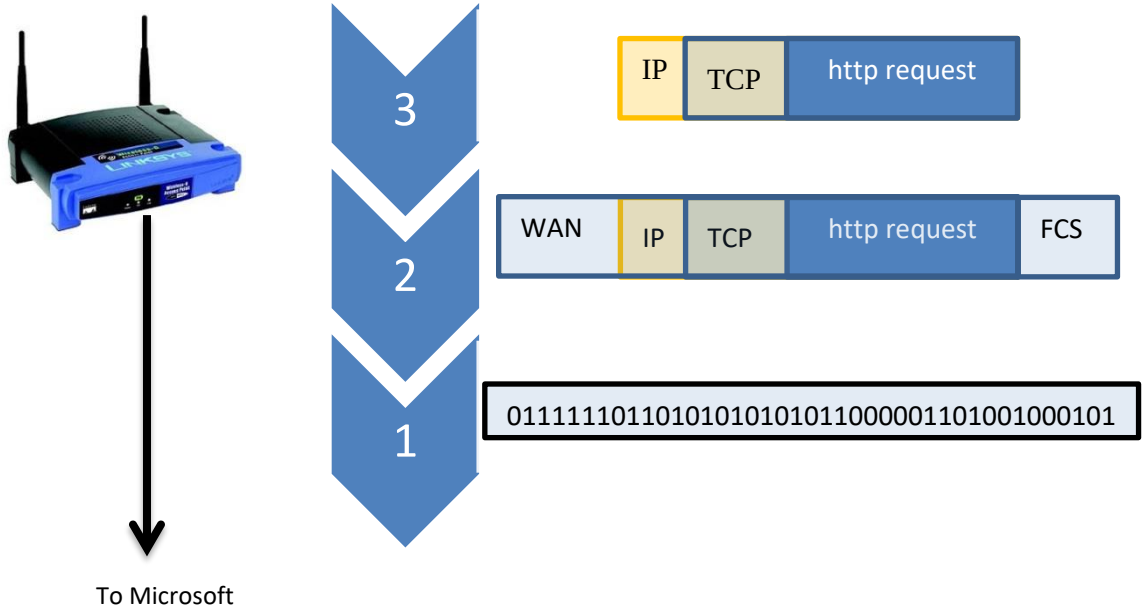
At the access point the frame is checked for errors; if there were errors, the transport layer would ask for a retransmission. If no errors, the access point adds a new DLL header and trailer because, in reality, the access point is a bridge connecting to different networks, a wireless Ethernet network to a wired Ethernet network. The new frame conforms to the Ethernet 802.3 frame format, with the MAC address of the Seneca proxy server and then the FCS is recalculated. The physical layer then transmits the binary information to the proxy server which is sent using CAT 6 UTP cable.



The proxy server removes the Ethernet headers and trailers and creates a WAN header and trailer so the frame can travel over the Internet and be passed from router to router. The proxy server converts the private network address 10.0.0.0 to a public address 142.240.0.0 and re-encapsulates the Internet datagram. The datagram is passed to the DLL layer of the server and a new MAC address is added identifying the nearest router.

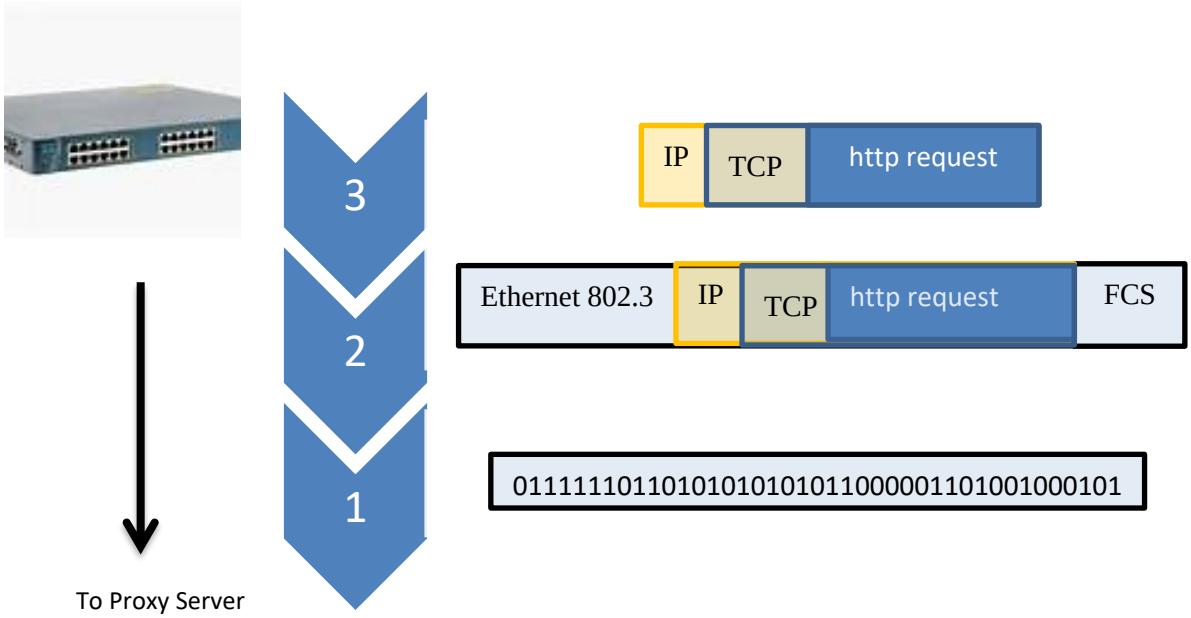


At each router the http message is opened up to review the destination IP address. Based on the destination the router will review its routing table to find the best route to relay the message to the next router. The router rips off the Ethernet LAN headers and trailers because the packet is about to transverse the Internet and recreates a WAN header and trailer to conform to the next leg of the packet's journey.

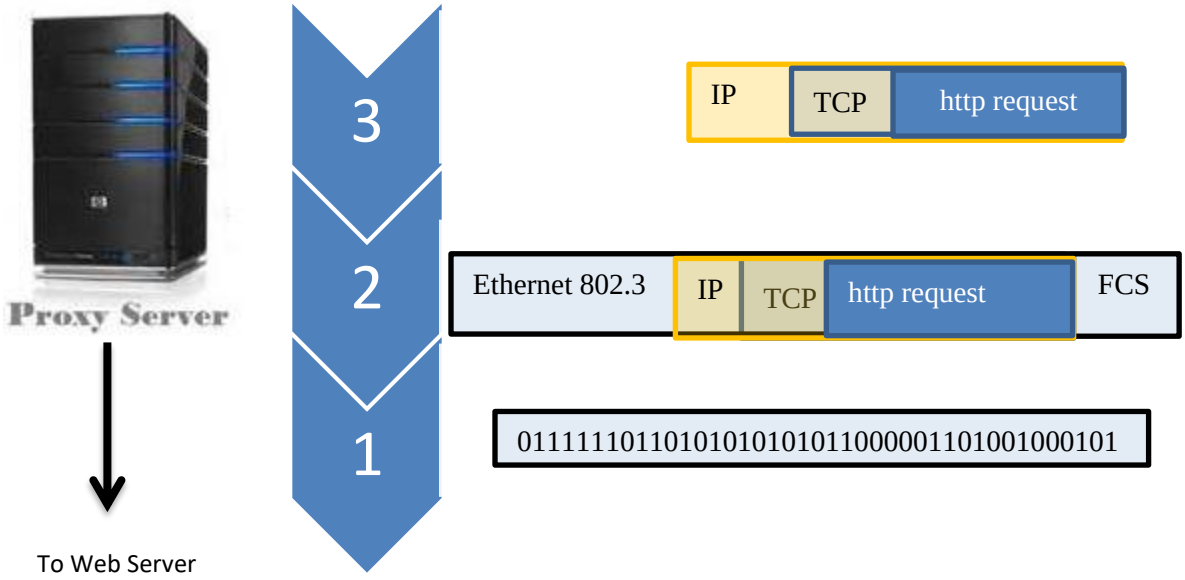


As the http message is passed from “link” to “link”, each router opens the Internet layer header to review the destination address. The router than makes a forwarding decision as to the best route to travel. If it needs to recreate new headers because the next link uses a different technology, it will replace the headers as necessary. Each routing decision gets the message one step closer to the destination.

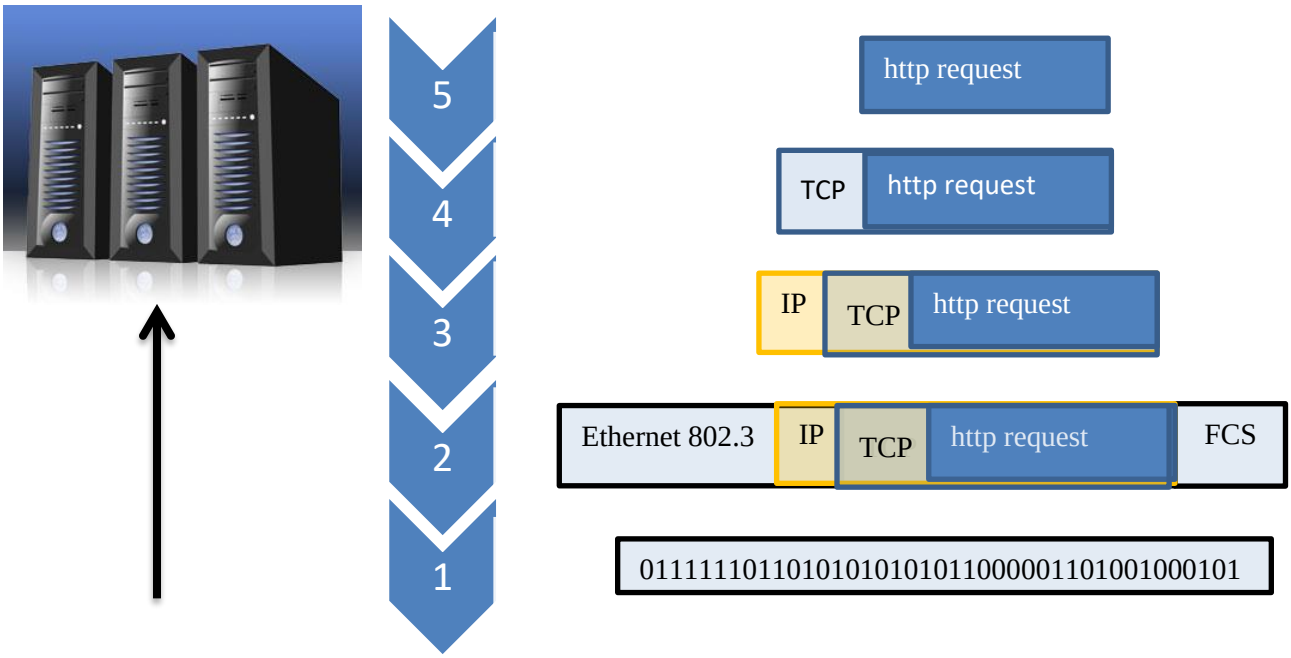
When the WAN packet arrives at the Microsoft network, the border router DLL checks the FCS to ensure no errors in transmission. If no errors, the DLL layer rips off the WAN header and trailer. The frame is forwarded to the Internet layer to add the IP address of the proxy server. The datagram is passed back to the Data Link Layer which recreates LAN headers and trailers so the packet can travel on the Microsoft local area network (assume that Microsoft is using Ethernet 802.3 as well) and adds the MAC address of the proxy server to the header. It then recalculates the FCS and the frame is passed to the physical layer.(assume that Microsoft uses Ethernet 802.3).



The frame is forwarded by switches on the Microsoft network until it arrives at the proxy server. The proxy server acts like a firewall and ensures that only legitimate http requests are forwarded to the web server farm. The proxy server DLL checks the FCS for errors and if no errors, it passes the frame to the Internet layer. This layer reviews the destination IP address and ports, converts the public IP address to the private IP address used by Microsoft and recreates the IP header. The datagram is passed down to the DLL layer which adds the MAC address of the web server farm to the DLL header and recalculates the FCS. The physical layer then encodes the 0s and 1s and forwards the signals onto the wired network.



The frame now passes through the switches of the Microsoft network until the web server recognizes MAC address and the physical layer then passes the frame to the DLL layer of the web server. The FCS is checked for errors and if no errors the Ethernet headers and trailers are removed and the datagram is passed to the Internet layer. The IP address is checked and then the Internet header is removed and the data is passed to the transport layer. The latter checks the sequence number to ensure that the TCP packet is a valid and checks the TCP checksum to ensure that there were no errors. The header is then removed and the data passed to the application layer which then processes the http request message. The web server then retrieves the requested web page, creates an http response message and the process now repeats as the response is forwarded back to the originating client at Seneca.



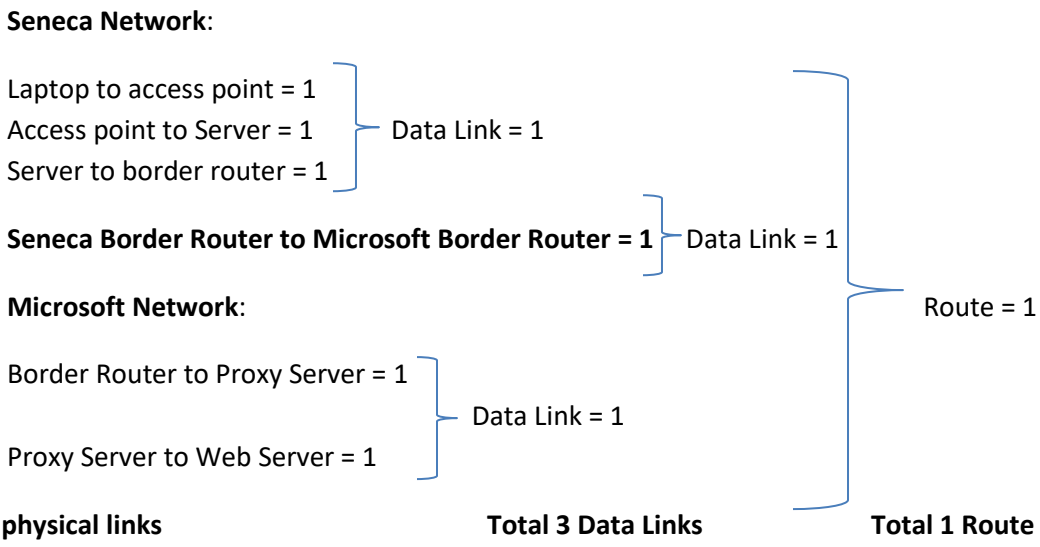
From this case study three points are evident:

1. The Data Link layer is responsible for framing the datagram so that it can travel over a single network. For example, the data must transverse the Seneca switched network to get to the border router. Transverse different WAN technologies across the Internet. At the destination network, the data must be forwarded by the Microsoft switched network to the web server. The data link layer of the host\router prepared the frame for the next step of it's journey by adding/removing LAN/WAN headers and/or trailers

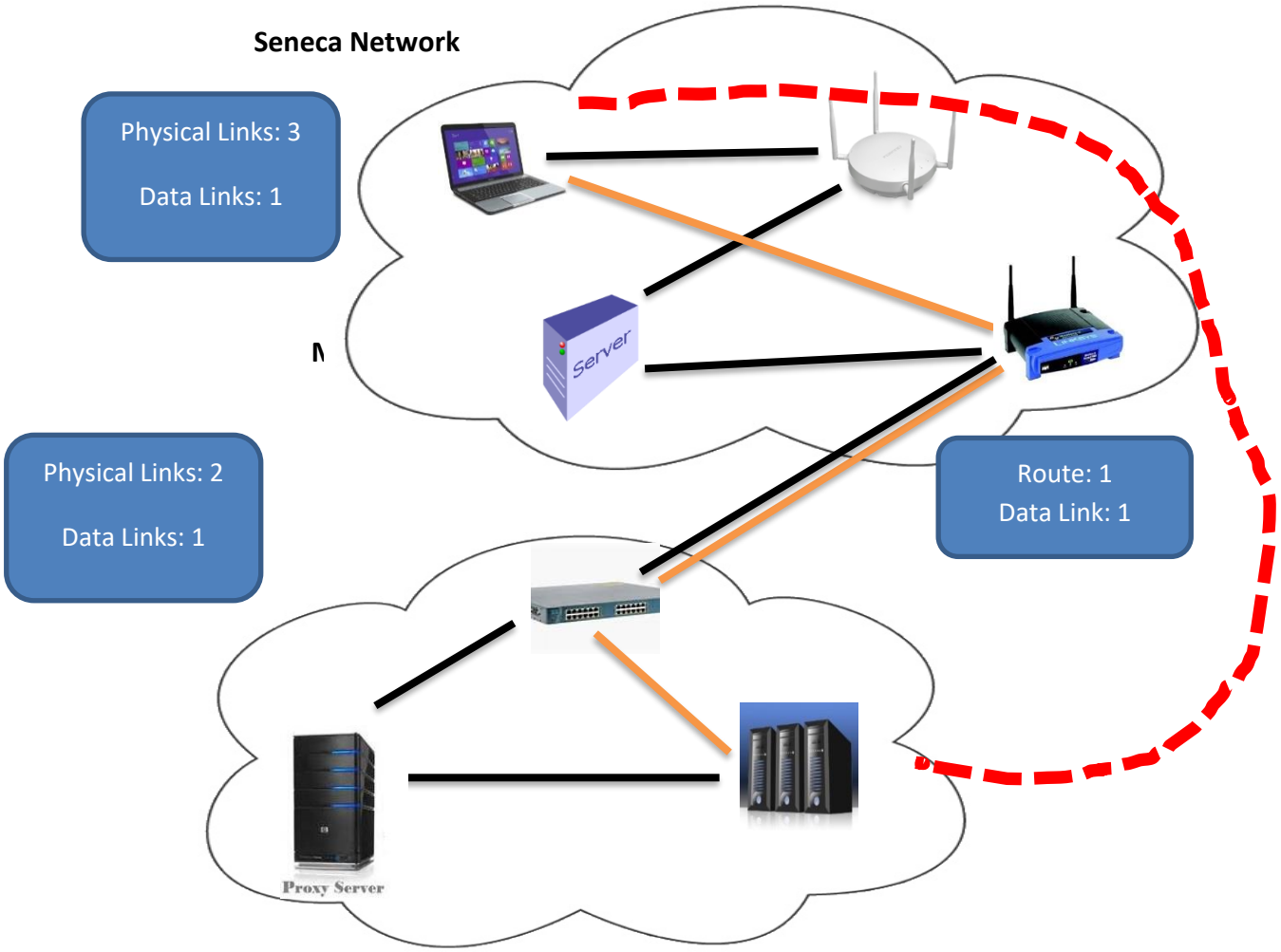
2. Although we like to think that the message is sent directly from the originating host to the destination host, using the IP address, this connection is virtual; in reality the message travels down 5 layers of the TCP/IP protocol stack, across each data link, link by link, (called a physical link) and then up 5 layers of the protocol stack at the receiving computer

3. Two addresses are required because all LAN/WAN technologies use the MAC/hardware address, or similar address, to move across a data link. A data link is the means used to interconnect one location with another to send information. Always remember routing is built on top of switching to allow different switched networks to communicate. This is the innovation Bob Kahn and Vint Cert had to connect different switched networks with a globally unique address.

We can see how to two addresses work together in the diagram below. This is a simplified diagram of how your message was sent from the Seneca open lab to the Microsoft web server. First, the message travels link by link. If we count the physical links there are 6 physical links (shown in black)



However, across a single switched network, there is only 1 data link, regardless of how many switches are used (show in orange). A data link is the means used to interconnect one location with another to send information. Across the Seneca switched network is a data link, across all the routers between Seneca and Microsoft is another data link; and from the Microsoft router to the web server is another data link. In total, there are 3 data links shown in orange. At each data link, the MAC address is used to move the frame from across the network. The 48-bit MAC address is unique with each device. Since many networks are using private IP addresses, it is very possible for 2 networks to being using the same IP addresses (32 or 128 bit address) for a host. This logical connection is called a route, and there is only 1 route for the packet from sending to receiving computer shown in red.



The synchronization problem

Computers work on clock cycles. When a computer is sending information internally, from RAM to the console, there is no synchronization problem because all parts of the computer are working in phase with each other. Data is sent on one clock cycle, the binary data is sampled in the middle of the next clock cycle. However, when computers are separated by distance it is difficult for computers to stay in

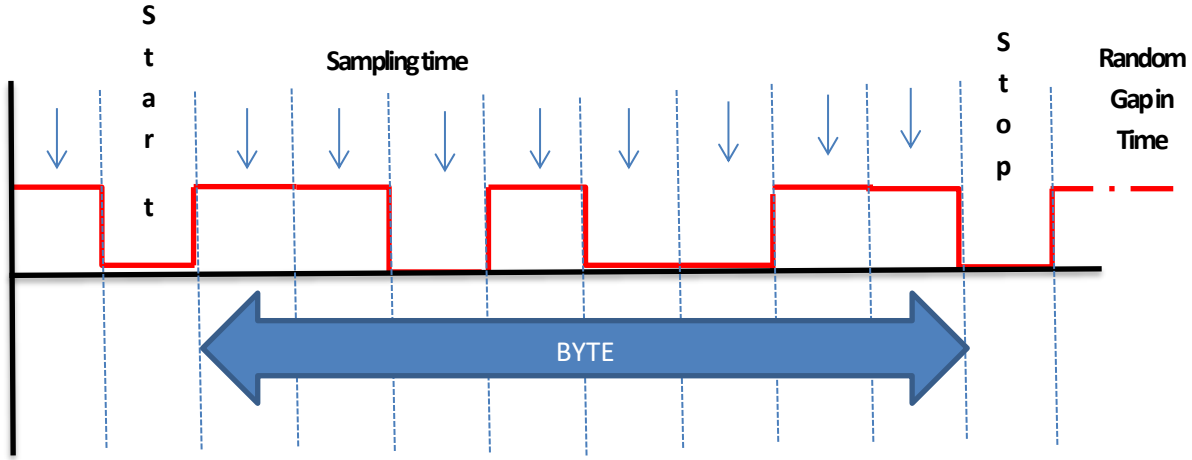
time with each other. For example, to send 1 bit across Canada on a fiber optic cable takes 30 ms. This delay is called "propagation delay" and can't be eliminated. In addition, each device has to process the data, read it into memory, execute instructions and send it out one of its ports. This process takes a few milliseconds for each device. As the data travels from link to link, this delay can build up, called "latency", which refers to the delay as the data travels from source to destination. In addition, routers can get congested and develop a backlog of packets which further adds to the latency. If the variance in time is too great when packets are received, jitter can occur. We experience jitter when our TV screen flickers during a show, or when we lost part of a conversation when using VoIP. How to keep remote computers in synch with each other has been a major goal of data communications.

Three methods have been developed to keep remote computers in time - even if they are in different parts of the world.

1. The best and most expensive way to keep computers in synch is to run a separate clocking wire to between them. If the sender and receiver are both working in time to the same clock source, then transmission can take place forever with the assurance that signal sampling at the receiver will always be in perfect synchronization with the transmitter. Some MAN (Metropolitan Area Network) such as SONET or FDDI work this way, but a separate clocking wire is impractical when the source and destination are separated by great distances.

2. The second method is to develop an encoding system which is "self-clocking". If the receiving computer can sample the signal in phase with the sending computer they will always be in synch. Manchester encoding does this by forcing the receiving computer to reset its clock two times per bit. For example, a transition from low to high indicates a one and a transition from high to low indicates a zero. This ensures synchronization, but is inefficient and can't be used for Gigabit Ethernet. NRZI is more efficient because at change in voltage at the beginning of a clock cycle is a one and no change is a zero. When a one is following by a zero, 2 bits are being sent per clock cycle. However, long strings of zeros, when the voltage is kept constant, can cause jitter.

3. The third method is to transmit data asynchronously so that timing is less of an issue. This is the most common method used internally for a PC. The sending computer starts a transmission by a drop in signal level. This indicates the "start bit" and the beginning of a transmission. After the start bit comes 8 bits of data followed by a "stop bit". The sending computer waits a random amount of time and the process repeats. The receiver tries to sample the signal in the middle of each bit time and with only 8 bits being sent jitter is minimized. Tests have shown that the sender and receiver can be out of synch by as much as 5% and the bit will still be interpreted correctly. The problem with this approach is that for every 8 bits sent you have 2 bits or 25% of overhead. Also, sending one byte at a time does not provide enough throughout for intensive applications like real time streaming or large data bases.



4. The better approach is to create a synchronous transmission which is a large data block (for Ethernet the block is 1500 bytes – Payload is much larger than indicated in the diagram). The address field holds the MAC address of the sending and receiving devices. The Control field is one or more bytes and contains information about the type of frame. For example, if the frame is user data or supervisory information needed by the data link devices. The Payload of the frame is the data field sent from the higher layer and is completely transparent to the DLL. The CRC refers to the "Cyclical Redundancy Check" which is a two byte field. The value of these bytes is the result of polynomial arithmetic based on every bit of data between the flags. When the frame is received, the calculation is repeated and compared with the sending computer's CRC value. If the values match then the receiving computer knows that there were no errors in transmission. Otherwise the frame is discarded. CRC is effective in detecting 99.9% of all network errors and has less than 1% of overhead making it very efficient. Finally, a flag of "01111110" (value 126 in decimal or 7E in hexadecimal) is added as delimiters of the frame.

To keep the computers in synch a periodic pulse of voltage is sent on a separate wire called a "Clock" or "Strobe" which tells the receiver "the current data bit is 'valid' at this moment in time". Synchronous transmission is used internally by your PC for parallel communication. For example, when the computer retrieves a system address or read/write data using the control bus.

Summary Table: Asynchronous and Synchronous

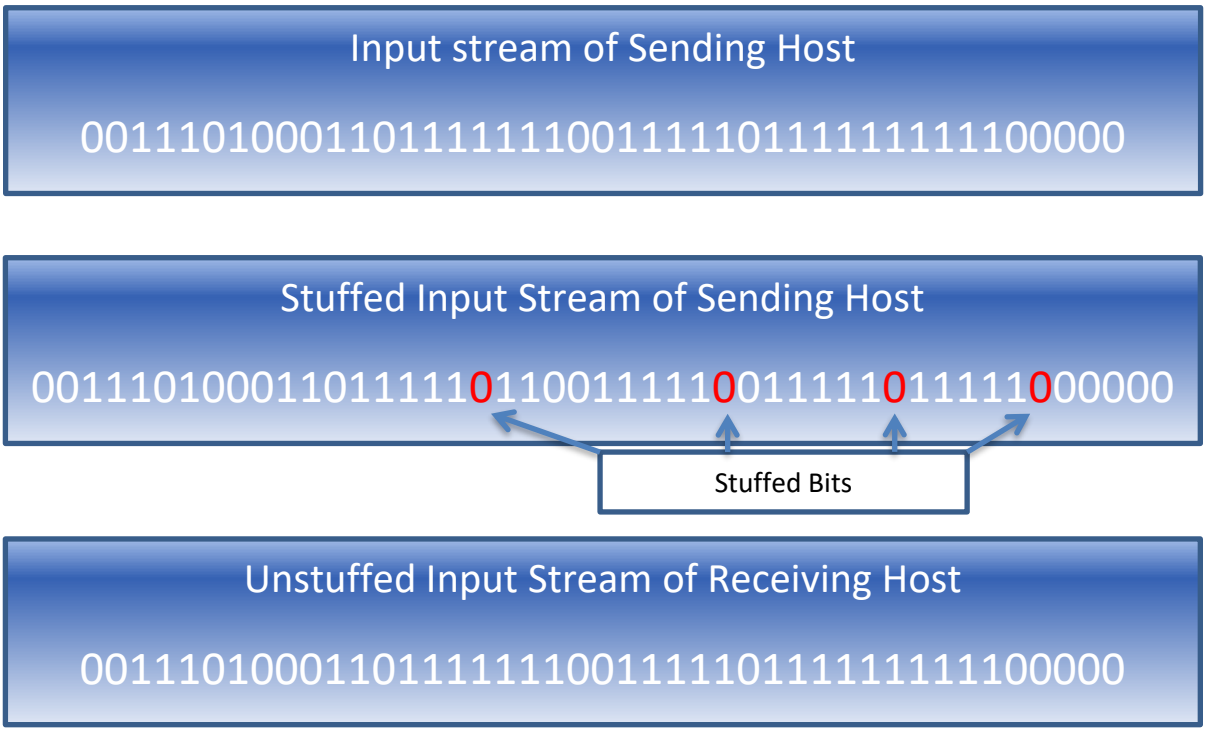
Type of Transmission	Advantages	Disadvantages
Asynchronous	Simple, uses less hardware and programming. Equipment less expensive.	Low throughput because of high overhead and slower speed
Synchronous	High throughput because of less overhead and larger frame size	Requires more hardware and programming.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

When constructing frames, the DLL layer can create an Asynchronous or Synchronous frame. Today, almost all communication is synchronous. However, if the line noise is high, the DLL layer will default to asynchronous to ensure proper transmission. There is also a WAN technology called Asynchronous Transfer Mode (ATM) which uses asynchronous transmissions(the frame size is based on 53 bytes). ATM networks are the core technology of the public switched telephone network (PSTN); it is also used on DSL and multiplexed lines where fast, low latency transmission is required. ATM is ideal for content such as voice and video. ATM networks, however, have been in decline which the growth of IP based technologies such as MPLS and increasing speeds of Ethernet. We will discuss these technologies in later lectures.

Data Link Layer Programming

Synchronous transmissions are the norm of network traffic. Each synchronous transmission has a flag of 126 in decimal or 7E in hexadecimal at the beginning and at the end of the frame. The flags are essential because they tell the receiving computer when a transmission starts and ends. The problem is if this value appears in anywhere else in the frame, the receiving computer could mistakenly interpret the value as the end of the frame. To avoid this program, the DLL layer has a build in programming routine called "bit stuffing".

The routine is relatively simple, but highly effective. Before the DLL adds, the flags, it scans the data stream. If it detects five consecutive "1s"'s , it adds or "stuffs" a "0" into the outgoing data stream. Then the DLL adds the flags. When the frame arrives at the DLL of the receiving host, it deletes the two flags, and scans the data stream for five consecutive "1s", followed by a "0"; when it finds this combination it "unstuffs" the "0" bit, before sending the data stream to the network layer.



The bit stuffing routine ensures that the flags are in fact the delimiters of the frame, avoiding any errors in the interpretation of the frame.

Simplex Half-Duplex and Full Duplex

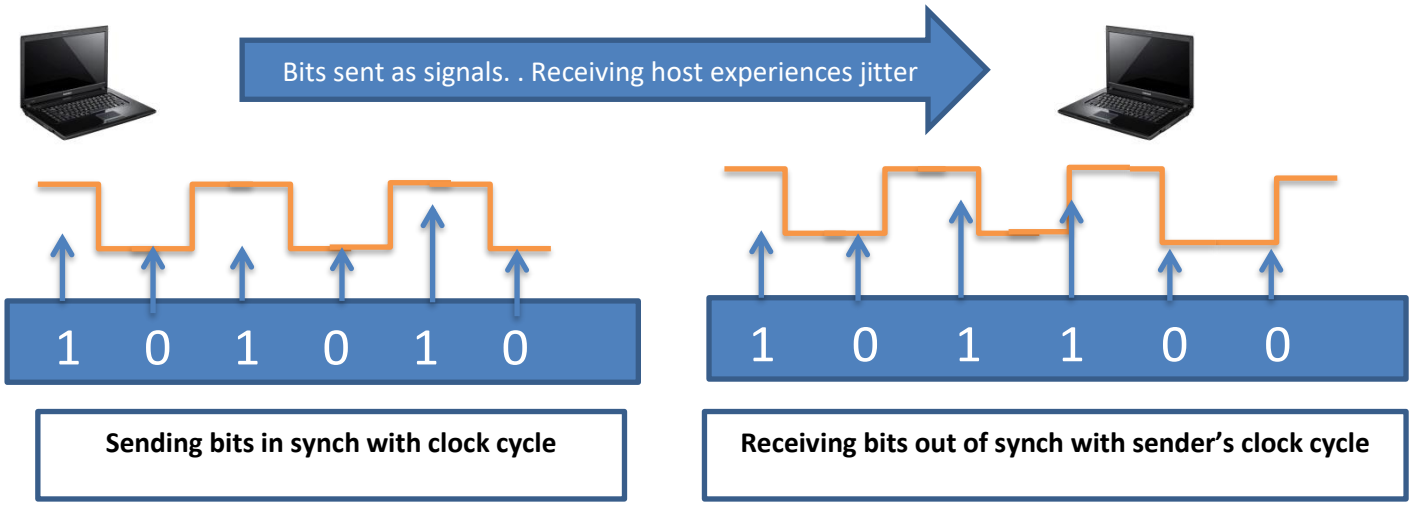
The outgoing data stream can be sent in one of three ways: simplex, half-duplex and full-duplex. For example, simplex means that the data can only travel in one direction. For example, internally, your keyboard can only send information to the CPU, the keyboard can never receive information from the CPU. Light travels in fiber optic cable in only one direction which makes fiber optic cable simplex. If you had only one wire, you could receive information, only if you stopped sending information. This is an example of half-duplex, you can send and receive, but not at the same time. The best form of

communication is full duplex which means you can send and receive at the same. Using fiber optic cable, you need to bundle 2 wires, one for sending and the other for receiving to make the cable full duplex. Coaxial cable can send full duplex communication by dividing the copper core into multiple channels using technology. One channel can send and the other could receive. Ethernet cables are all full duplex.

Common Transmission Errors:

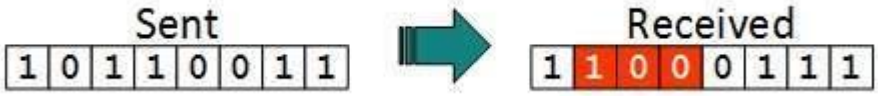
There are four common causes for transmission errors: noise, electromagnetic interference (EMI), crosstalk and jitter. Network noise is generated as electrons move down a cable, and can't be eliminated in copper cabling. This type of noise is called "thermal noise. It can't be eliminated which is why EIA/TIA has strict limits to the distance a signal can travel without regeneration. Noise can also be introduced from an external source, such as when lightning strikes, or random increases in line voltage, called "spikes", are conducted along the cable. EMI causes errors when cabling is located too close to noise sources which generate their own electromagnetic field, such as lights, elevator motors or other heavy machinery. We have experienced EMI when you are listening to the AM radio and you drive under high voltage lines, the music is disrupted with loud "SHSHSH" sound coming from the car speakers. Crosstalk is a special type of electromagnetic interference. Cables with electrons travelling in opposite directions can pull electrons off one cable, and they travel on the other cable, in the opposite direction. We have all experienced crosstalk on the telephone when you are talking to a friend, but you hear the echo of another conversation in the background. To avoid crosstalk, the wires are twisted so that the electromagnetic fields cancel each other. However, the ends of the cable which are straightened to connect to the RJ-45 connector can result in crosstalk if the ends are untwisted more than a ½". Jitter occurs when two computers "drift" out of synch with each other. This can happen if the receiving computer begins to sample the voltage too near the end of a clock cycle, it can misinterpret the value.

Jitter



Error Detection and Correction:

The Data Link Layer is the first layer that detects and corrects errors. Errors are a fact of life in network communications. The most common type of error is called "burst errors where multiple bits are lost or miss interpreted by the receiving host.



To detect errors two common procedures are used by the data link layer: parity check and Cyclic Redundancy Check (CRC). In both cases the data link layer adds redundant bits so the receiving computer can check for errors.

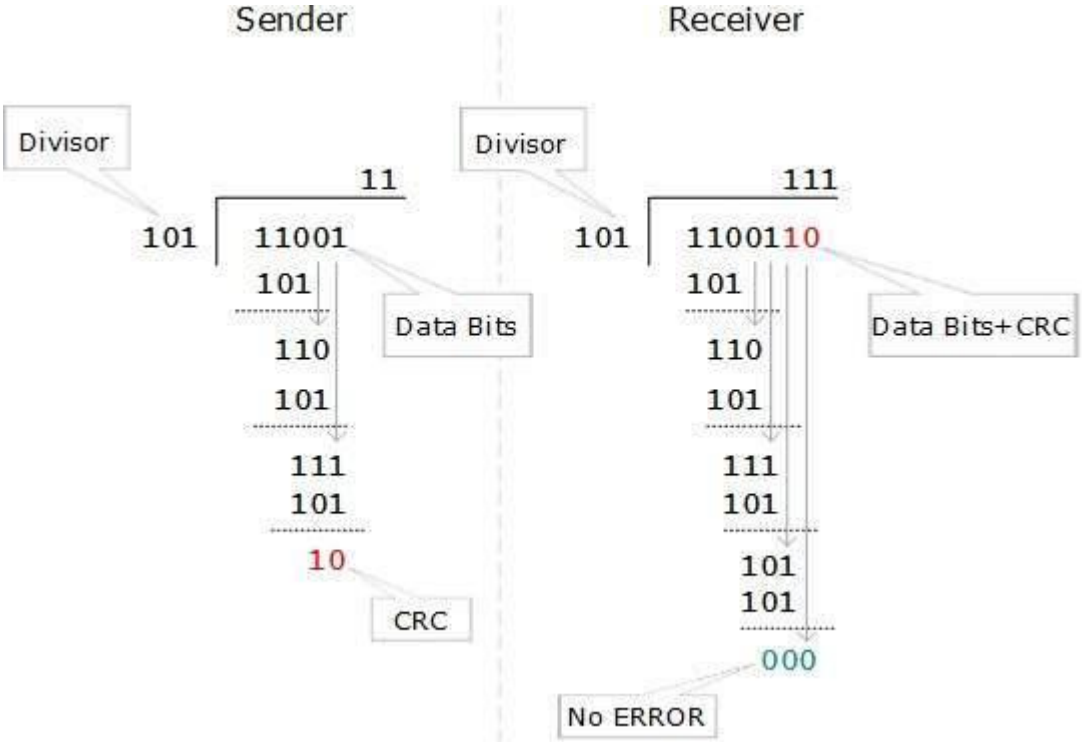
Parity Check

One extra bit is sent along with the original bits to make the number of 1s either even, in case of even parity, or odd, in case of odd parity. The sender counts the number of 1s and if the number of 1a is odd, the layer adds a bit to keep the parity even, if even parity is used (even parity is more common than odd parity)

The receiver simply counts the number of 1s in a frame if the number matches the parity bit used the frame is accepted. However, if 2 bits are flipped, this error will not be detected by parity because the parity bit is correct, but the message is lost. For this reason, there are multiple error checking, at the transport and application layers.

Cyclic Redundancy Check (CRC)

CRC uses binary division combined with polynomial arithmetic to detect if the received frame contains valid data. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits.



The receiver performs a division operation using the same CRC divisor. If the remainder contains all zeros the data bits are accepted, otherwise it is considered as there some data corruption occurred in transit. The CRC seems complicated and slow, but it is very fast because it is built into the hardware. It all has the lowest overhead, less than 1% (16 check bits for a transmission of 1500 bytes). Lastly, studies have shown that the CRC can detect 99.99% of all errors.

Error Correction

In the digital world, error correction can be done in two ways:

- Backward Error Correction: When the receiver detects an error in the data received, it sends a request back to the sender to retransmit the data unit.
- Forward Error Correction: When the receiver detects some error in the data received, it executes error-correcting code, which helps it to auto-recover and to correct some kinds of errors.

Backward Error Correction is simple and can only be efficiently used where retransmitting is not expensive. For example, fiber optics uses this approach. But in case of wireless transmission, or real-time streaming retransmitting is not practical. (Nobody wants to see a blank screen on the TV while a retransmission is in progress)

Backward Correction Pseudocode

Machine A sends to B.

NA = Network layer on A.
DA = Data Link layer on A.

NB = Network layer on B.
DB = Data Link layer on B.

If error, assume detected via CRC.
Receiver sets a timer to receive a response and sends ack packet indicating the next packet to receive. If no ack, sender re-sends.

Sender maintains `next_frame_to_send`
Receiver maintains `frame_expected`

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

Forward correction is based on hamming codes and the use of parity bits. A parity bit is added as previously discussed based on the number of 1s in the frame. In addition, redundant bits are added by the data link layer for error correction which in turn have a parity bit. This block of error correction bits with parity are called “Hamming codes” and allow the receiving computer to detect which bits in the data stream have errors and fix them on the fly. For this course, you do not need to learn how hamming codes work. If you are interested there is an excellent explanation at:

https://www.youtube.com/watch?v=JAMLuxdHH8o