Nathan Agbomedarho 400081762

Assignment 1:

Probability of getting a connection = 1/25 Probability of not getting a connection = 24/25 Required Probability: (24/25)^7 = 0.5% The maximum bit rate on a noisy channel can be calculated using the Shannon-Hartley theorem, which states that the maximum bit rate (in bits per second) of a channel is equal to the channel capacity, which is given by: **C = B * log2(1 + S/N)** where C is the channel capacity in bits per second, B is the bandwidth in hertz, S is the signal power, and N is the noise power.

To calculate the maximum bit rate on this channel, we need to know the bandwidth and the signal-to-noise ratio (SNR). The bandwidth is given as the difference between the upper and lower frequencies of the channel, which is 10 MHz - 4 MHz = 6 MHz. The SNR is given in decibels (dB) as SNRdB = 15 dB. We can convert this to a ratio using the formula:
SNR = 10^(SNRdB/10)which gives: SNR = 10^(15/10) = 31.62Now we can calculate the noise power as: N = S/SNR where S is the signal power. Since we don't know the signal power, we'll assume that it is equal to the noise power, so that the SNR is 1 (or 0 dB) and we can calculate the maximum bit rate for this worst-case scenario: N = S/1 S = N Plugging in the values we get: N = S/SNR = N/31.62 N = S = (1/31.62)*10^(-3) watts Finally, we can calculate the maximum bit rate as: C = B * log2(1 + S/N) = 6 * log2(1 + 10^3/31.62) = 6 * log2(31.62 + 1) = 6 * log2(32.62) = 6 * 5 = 30 Mbps (approximately) Therefore, the maximum bit rate on this channel is approximately 30 Mbps.

In a block of bits with 5 rows and 6 columns including horizontal and vertical parity bits, the total number of bits is 5 * 6 + 2 * 6 + 2 * 5 = 40. If exactly 4 bits are inverted due to transmission errors, the probability that the error will go undetected depends on the specific location of the inverted bits. If the inverted bits are located in a way that they do not affect the horizontal or vertical parity, then the error will go undetected. To estimate the probability, we can calculate the number of combinations of 4 bits that do not affect the parity and divide that by the total number of combinations of 4 bits. For example, consider the case where the 4 inverted bits are located in 4 different rows. In this case, the horizontal parity will still be correct. If we choose the 4 bits randomly, the number of combinations is C(5,4) = 5, and the probability of error detection is 1 - 5/C(40,4) = 1 - 5/91390. This is an upper bound on the probability of error detection, since the 4 inverted bits could be located in less than 4 different rows. A more accurate estimate can be obtained by considering all possible combinations of 4 bits and checking if they affect the parity. However, this is a complex calculation and the exact answer depends on the specific design of the block of bits. In general, the probability of error detection is close to 1, and the probability of error going undetected is close to 0. The closest answer to the probability of the error going undetected is 0.05. Since there are horizontal and vertical parity bits in this block of bits, it is possible to detect errors by checking if the parity bits are correct. If exactly 4 bits are inverted, there are 2 possibilities: either all 4 bits are in the same row, or all 4 bits are in the same column. In either case, the error can be detected by the corresponding parity bit. To calculate the probability of failure to detect the error, we need to consider the possibility that the 4 bits are inverted in a way that neither the horizontal nor the vertical parity bits can detect the error. This would happen if 2 bits are inverted in the same row and 2 bits are inverted in the same column, but not in the same row and column. The number of ways to choose 2 bits from a row is C(6, 2) = 15, and the number of ways to choose 2 bits from a column is C(5, 2) = 10. The total number of ways to choose 4 bits is C(30, 4) = 3003. So the probability of failure to detect the error is (15 * 10)/3003 = 0.05 So the probability of failure to detect the error is 0.05, or 5%.

Hamming code is an error detection and correction technique in which extra bits (parity bits) are added to the data to be transmitted to ensure that errors can be detected and corrected. In this case, the data byte to be transmitted is "10011010". To construct the Hamming code, the data byte is first separated into groups of bits and parity bits are added to the positions that correspond to powers of 2 (e.g. 1, 2, 4, 8, etc.). The parity bits are calculated such that the number of 1's in each group, including the parity bit, is always odd. Here's the process to generate the Hamming code for the data byte "10011010": To determine the number of parity bits required, you need to find the smallest number (r) that satisfies the following equation: 2^r >= m + r + 1 To determine whether each Hamming code is valid or not, we need to perform the following steps:

Step 1: Count the number of parity bits (p) in the Hamming code. Step 2: Calculate the value of each parity bit using even parity (i.e., the parity bit is 0 if the number of 1s in the corresponding data bits is even, and 1 otherwise). Step 3: Compare the calculated parity bits with the parity bits in the Hamming code. If they match, the Hamming code is valid; otherwise, it contains errors. Let's apply these steps to the given Hamming codes: Hamming code 1: 0 1 0 1 0 1 1 0 0 0 1 0 Step 1: There are 4 parity bits in this Hamming code. Step 2:Parity bit P1 = D3 ⊕ D5 ⊕ D7 ⊕ D9 ⊕ D11 = 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 1 = 0Parity bit P2 = D3 ⊕ D6 ⊕ D7 ⊕ D10 ⊕ D11 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 = 1Parity bit P3 = D5 ⊕ D6 ⊕ D7 ⊕ D12 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1 Parity bit P4 = D9 ⊕ D10 ⊕ D11 ⊕ D12 = 0 ⊕ 0 ⊕ 1 ⊕ 0 = 1 Step 3: The parity bits in the Hamming code are 1 1 1 1, which do not match the calculated parity bits 0 1 1 1. Therefore, this Hamming code contains errors. Hamming code 3: 0 1 0 1 0 1 1 0 0 0 1 1 Step 1: There are 4 parity bits in this Hamming code.
Step 2: Parity bit P1 = D3 ⊕ D5 ⊕ D7 ⊕ D9 ⊕ D11 = 0 ⊕ 01 ⊕ 1 ⊕ 1 = 0Parity bit P2 = D3 ⊕ D6 ⊕ D7 ⊕ D10 ⊕ D11 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 = 1Parity bit P3 = D5 ⊕ D6 ⊕ D7 ⊕ D12 = 1 ⊕ 1 ⊕ 1 ⊕ 1 = 0 Parity bit P4 = D9 ⊕ D10 ⊕ D11 ⊕ D12 = 0 ⊕ 0 ⊕ 1 ⊕ 1 = 0Step 3: The parity bits in the Hamming code are 0 1 0 0, which match the calculated parity bits 0 1 0 0. Therefore, this Hamming code is valid. Therefore, only the third Hamming code (0 1 0 1 0 1 1 0 0 0 1 1) is valid. Assignment 2:

The sliding window protocol uses sequence numbers to keep track of the order in which frames are sent and received, and to detect and recover from lost or corrupted frames. The number of bits needed for the sequence number depends on the size of the sliding window, which in turn depends on the round trip time (RTT) and the maximum transmission rate (in bits per second) of the network. To calculate the sliding window size, we first need to determine the maximum number of unacknowledged frames that can be in transit at any given time, which is given by: window_size = (transmission_rate * RTT) / frame_sizewhere transmission rate is the maximum transmission rate of the network in bits per second, RTT is the round trip time in seconds, and frame size is the size of each frame in bytes. In this case, we have: transmission_rate = 10Mbps = 10,000,000 bps RTT = 0.5 sec frame_size = 20KB = 20,000 bytes Plugging these values into the formula, we get: window_size = (10,000,000 * 0.5) / 20,000 = 25 So the maximum number of unacknowledged frames that can be in transit at any given time is 25. To accommodate this, we need a sequence number that can represent at least 25 different values. Since the sequence number is used to uniquely identify each frame, we require the sequence number to wrap around after it reaches its maximum value. Therefore, we require a sequence number that can wrap around at least 25 times. The minimum number of bits required for this sequence number is given by: bits = ceil(log2(25 * 2)) = ceil(log2(50)) = 6 So we require at least 6 bits for the sequence number to support a sliding window protocol with a window size of 25 frames on a 10Mbps network with a round trip latency of 0.5 second and 20 KB frame size. The following statement is true about the routing algorithms: If the cost on all the edges are the same, the shortest path from a source node to all other nodes computed by Dijkstra's algorithm is the breath first tree rooted at the source.

Explanation: Dijkstra's algorithm is a shortest path algorithm used in network routing protocols to calculate the shortest path from a source node to all other nodes in a network. If the cost on all the edges are the same, then Dijkstra's algorithm will generate a breadth-first tree rooted at the source, where each node in the tree is the same distance from the source.

The other statements are false: Distance vector routing (DVR) requires more bandwidth, whereas link state routing (LSR) requires less bandwidth. This statement is false. In fact, the opposite is true. Distance vector routing requires less bandwidth because routers only need to exchange their routing tables with their immediate neighbors, while link state routing requires more bandwidth because routers exchange more detailed information about the network topology. Count to infinity problem is present in both DVR and LSR. This statement is false. The count-to-infinity problem is only present in distance vector routing protocols, not in link state routing protocols. In distance vector routing, a router can incorrectly believe that it has found a shorter path to a destination node and continue to broadcast this information to its neighbors, causing a routing loop. Routing tables of a router keeps track of distributed IP addresses to network devices. This statement is false. Routing tables of a router keep track of network addresses and the next hop to reach that network, not individual IP addresses of devices.

In a sliding window protocol, the sender can transmit multiple frames at once without waiting for the acknowledgment of the previous frames. The receiver maintains a window of the frames it expects to receive next. In this case, we have a sliding window protocol with a window size of 10, meaning that the receiver is expecting frames with sequence numbers 21 through 30. The receiver can receive any frame within this window and acknowledge it. Out of the options given, the only impossible frame to arrive is 10. This is because the receiver is currently expecting frames with sequence numbers 21 through 30, and any frame with a sequence number less than 21 has already been acknowledged and is no longer expected. Frames 15, 20, 25, and 30 are all within the receiver's current window and can be received and acknowledged. However, frame 10 is not within the current window and has already been acknowledged, so it is impossible for it to arrive again.

When host A sends a message to host C, the message will be initially received by bridge B1. Since bridge B1 does not know the MAC address of host C, it will flood the message out of all its ports except for the port that the message was received on. The message will then be received by bridge B2 on two of its ports (one from B1 and the other from B4). Bridge B2 will flood the message out of all its ports except for the ports that the message was received on. The message will also be received by bridge B4 on one of its ports (from B2). Bridge B4 will flood the message out of all its ports except for the port that the message was received on. Finally, the message will be received by bridge B3 and forwarded out of the port connected to host C. Therefore, bridges B1, B2, B4, and B3 will forward the message from host A to host C. The path taken by the message is as follows: A -> B1 (flood) -> B2 (flood) -> B4 (flood) -> B2 -> B3 -> C. if host C sends a message to host D, all the bridges in the network will forward the message.

When host C sends a message to host D, the message will be initially received by bridge B3. Since bridge B3 does not know the MAC address of host D, it will flood the message out of all its ports except for the port that the message was received on. The message will then be received by bridge B2 on one of its ports (from B3) and bridge B4 on another port (from host D). Both bridges B2 and B4 will flood the message out of all their ports except for the ports that the message was received on. Finally, the message will be received by bridge B2 on one of its ports (from B4) and forwarded out of the port connected to bridge B1. Bridge B1 will then forward the message out of the port connected to host D. Therefore, all the bridges in the network (B1, B2, B3, and B4) will forward the message from host C to host D. The path taken by the message is as follows: C -> B3 (flood) -> B2 (flood) -> B4 (flood) -> B2 -> B1 -> D. When host A sends a message to host C, the message will be forwarded by bridges B1, B2, and B3. The forwarding tables in the bridges will be updated as follows: Bridge B1 will learn that the MAC address of host A is on port 1. Bridge B2 will learn that the MAC addresses of host A and B1 are on port 2 and that the MAC address of host C is on port 3. Bridge B3 will learn that the MAC addresses of host B2 and C are on port 2 and 3. When host A sends a message to host C, the message will be initially received by bridge B1. Since bridge B1 does not know the MAC address of host C, it will flood the message out of all its ports except for the port that the message was received on. The message will then be received by bridge B2 on two of its ports (one from B1 and the other from B4). Bridge B2 will forward the message out of the port connected to bridge B3. Finally, the message will be received by bridge B3 and forwarded out of the port connected to host C. Therefore, bridges B1, B2, and B3 will forward the message from host A to host C. The path taken by the message is as follows: A -> B1 (flood) -> B2 -> B3 -> C.

Seven Layer OSI network model: Application - Application specific protocols, Presentation: Format of exchanged data Session: Connection management, Transport: Process-to-process channel, Network: Host-to-host packet delivery, Data Link: Framing of data bits, Physical: Transmission of raw bits. Benefits include reduces complexity, standardizes interfaces, facilitates

Nathan Agbomedarho 400081762

modular engineering, facilitates interoperability, etc. **Name the OSI layer or layers in which medium access control (MAC) is addressed and state whether MAC is typically handled in hardware, in software, or in both in the Internet architecture**: Data link layer -> Hardware**. Explain how a receiver detects the end of a frame with length-based framing?** The length is indicated in the header of the frame. **Explain how a receiver detects the end of a frame with sentinel-based framing.** Special String/Characters. **Describe the benefits of error correction over error detection, and vice versa.** When error is detected, there is no need of retransmission. It will achieve better performance when error probability is relatively large. **What is the major advantage of CRC check? Under what circumstances will error detection using CRC fail? CRC** (Cyclic Redundancy Check) is a widely used error detection technique in digital communication systems. It adds redundant information (checksum) to the data that is transmitted and compares it with the checksum sent by the sender to detect errors. CRC is simple, fast, and can detect a wide range of errors. However, it can fail to detect errors if they occur within the same CRC block, due to malicious attacks, or if the noise or interference in the communication channel is high. Despite these limitations, CRC remains a widely used error detection technique due to its efficiency and effectiveness. **Describe the problem solved by medium access control (MAC). Medium** Access Control (MAC) is a sublayer of the Data Link Layer in the OSI model that solves the problem of sharing a single communication channel among multiple network devices. The MAC sublayer determines which device has access to the shared channel at any given time, and regulates the transmission of data to avoid collisions and ensure efficient use of the channel. The MAC protocol defines rules for how devices access the shared channel, including how they initiate and terminate transmissions, and how they handle collisions when they occur. By controlling access to the shared channel, the MAC sublayer ensures that each device gets a fair chance to transmit its data and that the channel is used efficiently. **Why does Ethernet use binary exponential backoff during contention resolution?** To avoid collisions and ensure fair access to the shared channel. When two or more devices attempt to transmit data on the same Ethernet network at the same time, a collision occurs, and the devices back off and try again later. To prevent repeated collisions, Ethernet uses a binary exponential backoff algorithm that increases the delay between retransmissions exponentially each time a collision occurs. This means that devices that experience frequent collisions will wait longer before attempting to transmit again, while devices that experience fewer collisions will wait less time. **Why does Ethernet have a minimum packet size? How is it determined?** To ensure that a sender can detect a collision, so it should still be sending when the collision signal propagates to the sender. The packet size = maximum RTT * bandwidth of the link**. Suppose packets on a wireless link consist of N data bits and H header bits each, where H is fixed. Suppose bits are received in error with probability P, independently of each other, and that N is adjusted to maximize the throughput of data in bits per second. If P gets larger, does the optimal value of N get larger or smaller? Why?** If packets on a wireless link consist of N data bits and H header bits each, where H is fixed, and bits are received in error with probability P, independently of each other, the optimal value of N to maximize the throughput of data in bits per second will depend on the specific details of the system, such as the bandwidth of the wireless link, the processing power of the devices, and the transmission power used. However, in general, as P gets larger, the optimal value of N will get smaller. This is because when the error rate is high, the likelihood of a packet being received in error is also high. Therefore, if the packet size is too large, the probability that the entire packet will be received in error also increases. This can lead to a waste of bandwidth and reduce the throughput of the system. On the other hand, if the packet size is too small, the overhead of the header bits will become a larger proportion of the total packet size, reducing the amount of data that can be transmitted in each packet and again reducing the overall throughput of the system. Therefore, there is an optimal value of N that balances the trade-off between maximizing the amount of data transmitted per packet and minimizing the probability of errors. As P gets larger, the optimal value of N will decrease to reduce the probability of receiving an entire packet in error and improve the overall throughput of the system. **Consider a frame consisting of two characters of four bits each. Assume that the probability of error is 10^-3, independent for each bit. What is the probability that the frame is received correctly? Add a parity bit to each character. Now what is the probability?** Without parity bits, the probability of receiving the frame correctly can be calculated as follows: The probability of a single bit being received in error is 10^-3. Since the frame consists of two characters, each with four bits, the total number of bits in the frame is 8. Therefore, the probability of receiving all 8 bits in the frame correctly is (1 - 10^-3)^8, which is approximately 0.9921. With parity bits, the probability of receiving the frame correctly can be improved. Assuming even parity (where the parity bit is set to 1 if the number of 1's in the character (excluding the parity bit) is odd, and set to 0 if the number of 1's is even), the probability can be calculated as follows: Each character now has a parity bit, making the total number of bits in the frame 10. For each character, the probability of receiving the 5 bits correctly is (1 - 10^-3)^5, which is approximately 0.9950. Therefore, the probability of receiving both characters, including their parity bits, correctly is (0.9950)^2, which is approximately 0.9900. If you add parity chances of bit errors increase, but chances of undetected errors decrease. **What does 4B/5B encoding accomplish, besides expanding the number of bits by 25%?** Limits length of 0-runs, guaranteeing a transition (in NRZI) every four bits transitions are necessary for clock recovery. **Explain the hidden terminal problem and how it is solved. The** hidden terminal problem is a scenario that can occur in wireless networks where two or more devices are out of range of each other, but within range of a central access point (AP). In this scenario, each device can communicate with the AP but not with each other, which can lead to collisions when the devices attempt to transmit data at the same time. This is because each device is unaware of the other devices' transmissions, and may therefore start transmitting its own data at the same time, leading to a collision at the AP. To solve the hidden terminal problem, a mechanism called the Request-to-Send/Clear-to-Send (RTS/CTS) handshake can be used. This mechanism involves a device that wants to transmit data first sending an RTS frame to the AP, requesting permission to transmit. The AP responds with a CTS frame, granting permission to transmit. Other devices that hear the RTS and CTS frames will know that the channel is occupied and wait until it is available again. This mechanism helps to prevent collisions and improve the overall efficiency of the network. Alternatively, a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol can also be used to solve the hidden terminal problem. This protocol involves devices sensing the channel before transmitting data, and waiting for a brief period of time before transmitting to avoid collisions. Additionally, the CSMA/CA protocol can also use a backoff mechanism to further avoid collisions by randomly selecting a delay time before reattempting transmission if a collision occurs. This mechanism helps to reduce the likelihood of multiple devices trying to transmit at the same time and causing collisions, thus improving the overall efficiency of the network. **Explain the exposed terminal problem and how it is solved.** The exposed terminal problem occurs in wireless networks when a device refrains from transmitting data even though it is within range of another device that is transmitting data to a third device. This can occur when the first device incorrectly senses the channel as busy due to the transmission of the second device, even though it is out of range of the second device's receiver. This can result in decreased network performance due to unnecessary delays in data transmission. To solve this problem, the Clear-to-Send (CTS) mechanism or the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol can be used to prevent collisions and improve the overall efficiency of the network. **In CSMA/CA, when transmission may occur**? In Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), transmission may occur only after a device senses that the channel is idle for a specified period of time called the Distributed Inter-Frame Space (DIFS). The DIFS is a minimum time interval that must elapse before a device can transmit data, and it allows other devices to sense the channel and determine whether it is busy. If the channel is busy, the device will wait for a random amount of time before attempting to transmit again. Once the channel is idle for a DIFS period, the device can transmit data with the confidence that there are no other devices transmitting at the same time, which reduces the likelihood of collisions and improves the overall efficiency of the network. **What is the Hamming Distance? What is the Hamming Distance of Hamming's (7, 4) code? Hamming** distance is a measure of the difference between two strings of equal length. It is defined as the number of positions in which the corresponding symbols of two strings are different. Hamming's (7,4) code is a type of error-correcting code in which 4 data bits are encoded into 7-bit codewords. The codewords are designed to have a minimum Hamming distance of 3, which means that any two codewords differ by at least 3 bit positions. To calculate the Hamming distance of the Hamming's (7,4) code, we can look at the minimum distance between any two valid codewords. Since the code is designed to have a minimum distance of 3, the Hamming distance of the code is 3. This means that any two codewords in the Hamming's (7,4) code differ in at least 3 bit positions. **Given a fixed data size in a frame, how many parity bits are needed to achieve 1-bit error correction?** If data size is N, and M is the number of parity bits, they should satisfy N+M+1<= 2^M. To achieve 1-bit error correction, we need to choose the minimum value of p that satisfies this equation. For example, if we have a data frame with 8 bits (m = 8), then we can calculate the minimum number of parity bits needed as follows: m + 1 + p ≤ 2^p, 8 + 1 + p ≤ 2^p, p ≤ 4 Therefore, we need at least 4 parity bits to achieve 1-bit error correction in a frame with 8 data bits. **What is the channel capacity of a noisy channel? When computing SNR how to convert it a decibel value?** The channel capacity of a noisy channel is the maximum rate at which error-free data can be transmitted over the channel, given a certain amount of noise or interference. It is a measure of the maximum amount of information that can be transmitted over the channel. The channel capacity of a noisy channel can be calculated using the Shannon-Hartley theorem, which states that the channel capacity C in bits per second is given by the formula: C = B * log2(1 + S/N) where B is the bandwidth of the channel in Hertz, S is the signal power in watts, and N is the noise power in watts. To compute SNR (Signal-to-Noise Ratio) in decibels (dB), we can use the following formula: SNR (dB) = 10 * log10(S/N) where S is the signal power and N is the noise power, both in watts. **How the forwarding with datagram, virtual circuit, and source routing work?** In summary, datagram forwarding forwards each packet independently based on the destination address in the header, virtual circuit forwarding establishes a logical connection between the source and destination before forwarding data packets based on the virtual circuit identifier, and source routing forwarding allows the source node to control the path of the packet by including the complete path information in the packet header. **How virtual circuits are established and tear down**? Virtual circuits are established and torn down in a connection-oriented network such as the Asynchronous Transfer Mode (ATM) network. The process of establishing a virtual circuit involves sending a connection request message from the source node to the destination node, checking the quality of service requirements, and allocating resources for the virtual circuit. Tearing down a virtual circuit involves sending a connection release message from the source node to the destination node, receiving the connection release acknowledgement message, and releasing the allocated resources. Although virtual circuits provide a reliable and efficient way to transmit data in a connection-oriented network, the process of establishing and tearing down virtual circuits can result in additional overhead and delay compared to connectionless networks. **Explain the main drawback of the stop-and-wait ARQ algorithm.** By only allowing one outstanding packet, bandwidth is not used efficiently (i.e. stopand-wait does not keep the pipe full) **Relationship between Receive Window Size (RWS), Next Frame Expected (NFE) and Last Frame Acceptable (LFA) in a sliding window protocol? LFA – NFE + 1 ≤ RWS. Determine min sequence number in Sliding window algo? Maximum number of outgoing frames = Throughput delay product / frame size. Maximum needed sequence number is two times of the above value. Then, log_2(MaxSeqNum) will be needed. (We take the ceiling) What do "learning" bridges actually learn? What do they use this information for?** "Learn" which hosts live on which LAN, Maintain forwarding table, Only forward when necessary, Reduces bridge workload. **How do the distance vector routing and link state routing algorithm work? What is the major drawback of these algorithms?** Distance vector and link state routing are two popular routing algorithms used in computer networks. Distance vector routing involves each router maintaining a table of distances to all other routers in the network and periodically broadcasting its routing table to its neighboring routers. Link state routing involves each router maintaining a map of the entire network topology and broadcasting a link state packet when a change occurs in the network. The major drawback of both algorithms is that they can suffer from routing loops and count-to-infinity problems, which can be overcome using additional techniques.