**Q1:** Suppose that Selective Repeat ARQ is modified so that ACK messages contain a list of the next m frames that it expects to receive

1. How does the protocol need to be modified to accommodate this change?
2. What is the effect of the change on protocol performance?

**Answer:**

a. To modify Selective Repeat ARQ so that ACK messages contain a list of the next m frames that it expects to receive, the protocol needs to be modified as follows:

* The sender maintains a sliding window that includes all unacknowledged frames, as in the standard Selective Repeat ARQ protocol.
* When the receiver receives a frame, it checks to see if it is the next expected frame in the sequence. If it is, the receiver sends an ACK message that contains a list of the next m frames that it expects to receive.
* If there are gaps in the received frames, the receiver sends an ACK message that requests retransmission of the missing frames. The sender then retransmits the requested frames.

b. The effect of this change on protocol performance depends on the value of m and the characteristics of the network.

* One potential benefit of this modification is improved efficiency, particularly in networks with high latency or high error rates. By including a list of expected frames in each ACK message, the receiver can help reduce the number of unnecessary retransmissions. For example, if the sender knows that the receiver is expecting frames 10-20, it can prioritize those frames for transmission instead of sending other frames that may not be needed.
* However, there are also potential drawbacks to this modification. One concern is increased overhead due to the larger size of the ACK messages. Depending on the value of m, the size of each ACK message could be significantly larger than in the standard protocol, which could impact network performance. Additionally, the more frames that are included in each ACK message, the greater the risk of errors in the ACK message itself, which could lead to further retransmissions and delays.

**Q2:** A 1.5 Mbps communications link is to use HDLC to transmit information to the moon. What is the smallest possible frame size that allows continuous transmission? The distance between earth and the moon is approximately 375,000 km, and the speed of light is 3 x 108 meters/second.

**Answer:**

To determine the smallest possible frame size that allows continuous transmission, we need to calculate the round-trip time (RTT) for a signal to travel from Earth to the Moon and back.

RTT = 2 \* distance / speed of light

Converting the distance between Earth and the Moon to meters:

375,000 km \* 1000 m/km = 375,000,000 meters

Plugging in the values:

RTT = 2 \* 375,000,000 / 3 x 10^8

= 5 seconds

To achieve continuous transmission on a 1.5 Mbps link with HDLC, we need to calculate the minimum frame size that can be transmitted within this RTT.

Minimum frame size = (Link speed \* RTT) / Efficiency

The efficiency factor takes into account the protocol overhead, such as header and trailer bits.

Assuming an efficiency factor of 80%, we get:

Minimum frame size = (1.5 Mbps \* 5 sec) / 0.8

= 11.25 megabits

To convert to bytes:

11.25 megabits / 8 bits/byte = 1.41 megabytes

Therefore, the smallest possible frame size that allows continuous transmission on a 1.5 Mbps HDLC link to the Moon is approximately 1.41 megabytes.

**Q3:** Consider the transfer of a single real-time telephone voice signal across a packet network. Suppose that each voice sample should not be delayed by more than 20ms.

1. Discuss which of the following adaptation functions are relevant to meeting the requirements of this transfer: handling of arbitrary message size; reliability and sequencing; pacing and flow control; timing; addressing; and privacy, integrity and authentication.
2. Compare a hop-by-hop approach to an end-to-end approach to meeting the requirements of the voice signal.

**Answer:**

a. To meet the requirement of transferring a single real-time telephone voice signal across a packet network with a maximum delay of 20ms, the following adaptation functions are relevant:

- Timing: The timing adaptation function is critical in ensuring that each voice sample is delivered within the required deadline. The network must be able to synchronize its clock with the sender and receiver to maintain the required time intervals between packets.

- Reliability and sequencing: To ensure that each voice sample is delivered without loss or mis-ordering, the reliability and sequencing adaptation function is necessary. This requires the use of error detection and correction mechanisms, as well as sequencing and resequencing of packets at the receiver end.

- Pacing and flow control: To prevent packet loss due to congestion, pacing and flow control mechanisms are necessary. These mechanisms regulate the rate at which packets are transmitted and received to match the capacity of the network.

- Addressing: Addressing is necessary to identify the source and destination of each voice sample. It also enables routing of packets through the network.

b. There are two approaches for meeting the requirements of a real-time telephone voice signal over a packet network: the hop-by-hop approach and the end-to-end approach.

The hop-by-hop approach involves implementing the required adaptation functions at each intermediate node in the packet network. Each node processes the packets it receives before forwarding them to the next node. This approach can introduce additional delays and overhead due to processing at each node. Furthermore, if a node fails, the entire communication may become compromised.

The end-to-end approach involves implementing the required adaptation functions only at the endpoints of the communication path, i.e., the sender and receiver of the voice signal. The packets are transmitted through the network without modification, and any required processing is performed at the endpoints. This approach minimizes delays and overhead, but it may not be suitable for networks with high packet loss rates or variable delays.

In general, the end-to-end approach is preferred for real-time voice communications over packet networks because it minimizes delays and overhead. However, the hop-by-hop approach may be necessary in some situations, such as when the network has high delay or loss rates, or when additional processing is necessary at intermediate nodes.

**Q4:** In this activity, you are given the network address of 192.168.100.0/24 to subnet and provide the IP addressing for the Packet Tracer network. Each LAN is the network requires at least 25 addresses for end devices, the switch and the router. The connection between R1 to R2 will require an IP address for each end of the link.

1. Base on the topology, how many subnets are needed?
2. How many bits must be borrowed to support the number of subnets in the topology table?
3. How many subnets does this create?
4. How many usable hosts does this create per subnet?

**Answer:**

a. Based on the topology, we need 4 subnets - one for each LAN and one for the link between R1 and R2.

b. To support 4 subnets, we need to borrow two bits from the host portion of the IP address. This is because 2^2 = 4 (remember that the formula for calculating the number of subnets is 2^n, where n is the number of borrowed bits).

c. Borrowing two bits creates four subnets: 192.168.1.0/26, 192.168.1.64/26, 192.168.1.128/26, and 192.168.1.192/26.

d. Each subnet has 62 usable host addresses. This is because a /26 subnet provides 64 total addresses, but two of those are reserved for the network and broadcast addresses, leaving 62 usable addresses per subnet.

**Q5:**

1. Use the Dijkstra algorithm to find the set of shortest paths from node 4 to other nodes.
2. Find the set of associated routing table entries (Destination, Next Hop, Cost)

**Answer:**

* Step 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration** | **Node 1** | **Node 2** | **Node 3** | **Node 4** | **Node 5** |
| **Initial** | (-1, ) | (-1, ) | (-1, ) | (-1, ) | (-1, ) |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |

* Step 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration** | **Node 1** | **Node 2** | **Node 3** | **Node 4** | **Node 5** |
| **Initial** | (-1, ) | (-1, ) | (-1, ) | (-1, ) | (-1, ) |
| **1** | (-1, ) | (-1, ) | (6,1) | (-1, ) | (6,2) |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |

* Step 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration** | **Node 1** | **Node 2** | **Node 3** | **Node 4** | **Node 5** |
| **Initial** | (-1, ) | (-1, ) | (-1, ) | (-1, ) | (-1, ) |
| **1** | (-1, ) | (-1, ) | (6,1) | (-1, ) | (6,2) |
| **2** | (3,3) | (5,6) | (6,1) | (3,3) | (6,2) |
| **3** |  |  |  |  |  |

* Step 4:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration** | **Node 1** | **Node 2** | **Node 3** | **Node 4** | **Node 5** |
| **Initial** | (-1, ) | (-1, ) | (-1, ) | (-1, ) | (-1, ) |
| **1** | (-1, ) | (-1, ) | (6,1) | (-1, ) | (6,2) |
| **2** | (3,3) | (5,6) | (6,1) | (3,3) | (6,2) |
| **3** | (3,3) | (4,4) | (6,1) | (3,3) | (6,2) |

* The result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **N** | **D1** | **D2** | **D3** | **D5** | **D6** |
| **Initial** | {1} | 3 | 2 Ѵ  ↘ | 5 | ↘ |  |
| **1** | {1,3} | 3  Ѵ | 2 | 4  ↘ |  | 3 |
| **2** | {1,2,3} | 3 | 2 | 4 | 7 | 3  Ѵ |
| **3** | {1,2,3,6} | 3 | 2 | 4  Ѵ | 5 | 3 |
| **4** | {1,2,3,4,6} | 3 | 2 | 4 | 5  Ѵ | 3 |
| **5** | {1,2,3,4,5,6} | 3 | 2 | 4 | 5 | 3 |