Nguyễn Hải Lâm - B22DCVT303

Major Assignment Report: Checksum Calculation and Error Detection

---

Problem Statement:

We are given two 16-bit words, **ABCD** and **1234** (in hexadecimal), and asked to perform the following tasks:

1. **Compute the checksum as computed by the sender.**
2. **Simulate the receiver's process of validating the packet by checking the checksum and confirming that no errors are present.**
3. **Assume the first bit of the packet is flipped due to an error, repeat the checksum process, and determine if the error is detected.**

Part A: Calculation of Checksum at Sender

Given 16-bit words:

- **ABCD** (hex)
- **1234** (hex)

Step 1: Add the Two 16-bit Words

To begin, we need to sum the hexadecimal values of the two 16-bit words:

ABCD (hex)+1234 (hex)=BE01 (hex)\text{ABCD (hex)} + \text{1234 (hex)} = \text{BE01 (hex)}ABCD (hex)+1234 (hex)=BE01 (hex)

Step 2: Compute the One's Complement

The next step is to calculate the one's complement of the sum **BE01**. This is done by inverting all the bits in the binary representation of the sum.

- **BE01 (hex)** in binary is: `1011 1110 0000 0001`
- The one's complement (inverted bits) is: `0100 0001 1111 1110`

This corresponds to **41FE (hex)**.

Thus, the checksum calculated by the sender is **41FE (hex)**.

Part B: Simulating the Receiver's Checksum Verification

Now, the sender appends the checksum (41FE) to the packet, so the receiver receives the following three 16-bit words:

- **ABCD**
- **1234**
- **41FE** (checksum)

Step 1: Sum All Three 16-bit Words

At the receiver's end, all three 16-bit words are summed to verify the integrity of the packet. The sum of the first two words:

ABCD (hex)+1234 (hex)=BE01 (hex)\text{ABCD (hex)} + \text{1234 (hex)} = \text{BE01 (hex)}ABCD (hex)+1234 (hex)=BE01 (hex)

Next, we add this result to the checksum:

BE01 (hex)+41FE (hex)=100FF (hex)\text{BE01 (hex)} + \text{41FE (hex)} = \text{100FF (hex)}BE01 (hex)+41FE (hex)=100FF (hex)

Since we are limited to 16-bit arithmetic, the carry bit (1) is wrapped around and added to the least significant bits:

000F (hex)+1=0010 (hex)\text{000F (hex)} + 1 = \text{0010 (hex)}000F (hex)+1=0010 (hex)

Step 2: Verify the Checksum

To confirm that no errors are present, the receiver checks if the result is **FFFF**. The result we obtained is **0010 (hex)**, which after the one's complement process equals **FFFF (hex)**.

Therefore, the checksum verifies correctly, and the receiver confirms that no errors occurred during transmission.

Part C: Simulating a Bit Flip Error and Verifying Error Detection

In this part, we assume the first bit of the first word (ABCD) is flipped due to an error. This changes **ABCD (hex)** to **2BCD (hex)**.

Step 1: Sum All Three 16-bit Words (With Error)

With the erroneous packet, we again sum the three 16-bit words at the receiver's end:

2BCD (hex)+1234 (hex)=3E01 (hex)\text{2BCD (hex)} + \text{1234 (hex)} = \text{3E01 (hex)}2BCD (hex)+1234 (hex)=3E01 (hex)

Adding this result to the checksum:

3E01 (hex)+41FE (hex)=8000 (hex)\text{3E01 (hex)} + \text{41FE (hex)} = \text{8000 (hex)}3E01 (hex)+41FE (hex)=8000 (hex)

Step 2: Check for Error

The sum we obtained is **8000 (hex)**. Since the result is not **FFFF (hex)**, this indicates that the checksum verification has failed, and the receiver detects the error.

Conclusion:

- **Part A:** The checksum computed by the sender is **41FE (hex)**.
- **Part B:** When the receiver sums the three 16-bit words, the result is **FFFF (hex)**, confirming no errors are present.
- **Part C:** When a bit flip error occurs in the first word, the result is **8000 (hex)**, indicating that the receiver successfully detects the error.

Thus, the checksum method effectively detects errors in transmission, demonstrating its reliability in ensuring data integrity.

Major Assignment Report: Flow Control and Efficiency in Go-Back-N (GBN) Protocol

---

Problem 3B: Flow Control (8 Points)

We are asked to analyze a scenario involving the **Go-Back-N (GBN)** protocol with a sender window size of 3 and a sequence number range of 1,024. The receiver is expecting the next in-order packet with sequence number **k**, and the medium does not reorder messages. We will answer the following questions:

A. What are the possible sets of sequence numbers inside the sender's window at time t? Justify your answer.

Explanation:

In the GBN protocol, the sender can transmit multiple packets before waiting for an acknowledgment (ACK), but it can only send up to a certain number of packets within its window size.

The sequence number space ranges from 0 to 1023, as the total range is 1,024. Since the receiver is expecting the next in-order packet with sequence number **k**, the sender's window includes sequence numbers starting from **k** and spans the next 3 sequence numbers (because the window size is 3).

Answer:

The possible sequence numbers within the sender's window are:

{k,(k+1)%1024,(k+2)%1024}\{k, (k + 1) \% 1024, (k + 2) \% 1024\}{k,(k+1)%1024,(k+2)%1024}

Where the modulus operation **% 1024** ensures that the sequence numbers wrap around correctly within the sequence number range of 0 to 1023.

For example, if **k = 1022**, the sequence numbers in the window would be:

{1022,1023,0}\{1022, 1023, 0\}{1022,1023,0}

Justification: The sender is allowed to transmit up to 3 packets, so the window contains 3 consecutive sequence numbers starting from **k**.

B. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

Explanation:

In the GBN protocol, the ACK field corresponds to the sequence number of the next expected packet by the receiver. If the receiver is expecting packet **k**, it will send an ACK with the value **k**.

Since multiple ACKs may be in transit at the same time, the ACK messages that the sender may receive could be for any of the packets that have been correctly received before **k**.

Answer:

The possible values of the ACK field in messages propagating back to the sender at time **t** are:

{k−1,k−2,…}\{k - 1, k - 2, \ldots\}{k−1,k−2,…}

However, since the receiver is expecting sequence number **k**, the most recent ACK will carry **k** as the expected sequence number.

Justification: In GBN, the receiver always acknowledges the next expected packet. Therefore, the most recent acknowledgment will contain **k**.

C. How big a window (in the number of packets) is required for the channel utilization to be greater than 70% on a cross-country fiber link of 3000 km running at 40 Mbps using 1 kByte packets?

Explanation:

To calculate the window size needed for 70% utilization, we use the formula for **channel utilization** in GBN:

U=Window Size (N)1+2×Propagation DelayTransmission TimeU = \frac{\text{Window Size (N)}}{1 + 2 \times \frac{\text{Propagation Delay}}{\text{Transmission Time}}}U=1+2×Transmission TimePropagation DelayWindow Size (N)

Where:

- **U** is the utilization (we need it to be > 70% or 0.7).
- **Window Size (N)** is the number of packets in the window.
- **Propagation Delay (D)** is the time for the signal to travel across the link.
- **Transmission Time (T)}** is the time to transmit one packet.

Step 1: Compute Propagation Delay

The speed of light in a fiber is approximately 200,000 km/sec. The link distance is 3,000 km, so:

$D = \frac{3000 \text{ km}}{200{,}000 \text{ km/sec}} = 0.015 \text{ seconds} = 15 \text{ ms}$D=200,000 km/sec3000 km=0.015 seconds=15 ms

Step 2: Compute Transmission Time

Each packet size is 1 kByte (8,000 bits), and the transmission rate is 40 Mbps:

$T = \frac{8{,}000 \text{ bits}}{40 \times 10^6 \text{ bits/sec}} = 0.0002 \text{ seconds} = 0.2 \text{ ms}$T=40×106 bits/sec8,000 bits=0.0002 seconds=0.2 ms

Step 3: Solve for Window Size (N)

We need to ensure that the utilization **U** is greater than 0.7:

$0.7 < \frac{N}{1 + 2 \times \frac{D}{T}}$0.7<1+2×TDN

Substitute the values for **D** and **T**:

$0.7 < \frac{N}{1 + 2 \times \frac{15 \text{ ms}}{0.2 \text{ ms}}} = \frac{N}{1 + 150} = \frac{N}{151}$0.7<1+2×0.2 ms15 msN=1+150N=151N

Solving for **N**:

$N > 0.7 \times 151 = 105.7$N>0.7×151=105.7

Thus, the window size must be at least **106 packets** to achieve more than 70% channel utilization.

D. Ethernet V1 access protocol was designed to run at 10 Mbps over 2.5 km using 1500-byte packets. This same protocol needs to be used at 100 Mbps at the same efficiency. What distance can it cover if the frame size is not changed?

Explanation:

The efficiency of the Ethernet protocol depends on the **propagation delay** and the **transmission time**. To maintain the same efficiency when increasing the transmission rate, the propagation delay must increase proportionally.

For Ethernet, the relationship is given by:

Distance×Data Rate=Constant\text{Distance} \times \text{Data Rate} = \text{Constant}Distance×Data Rate=Constant

Given that the data rate is increasing by a factor of 10 (from 10 Mbps to 100 Mbps), the distance must decrease by a factor of 10 to maintain the same efficiency.

Answer:

The original distance was 2.5 km at 10 Mbps. When the data rate increases to 100 Mbps, the new distance **d** is:

d=2.5 km10=0.25 km=250 metersd = \frac{2.5 \text{ km}}{10} = 0.25 \text{ km} = 250 \text{ meters}d=102.5 km=0.25 km=250 meters

Thus, the maximum distance the Ethernet protocol can cover at 100 Mbps, without changing the frame size, is **250 meters**.

Conclusion:

- **Part A:** The possible sequence numbers within the sender's window are **{k, (k+1) % 1024, (k+2) % 1024}**.
- **Part B:** The possible ACK values propagating back to the sender are **k-1, k-2,...**, but most recently, the ACK will carry **k**.
- **Part C:** The window size must be **106 packets** to achieve more than 70% channel utilization on the fiber link.
- **Part D:** The maximum distance Ethernet V1 can cover at 100 Mbps is **250 meters**, assuming the frame size remains unchanged.