Georgia Southwestern State University

CSCI_6230 Fall 2023

Lee_Brunovsky_HW3.pdf

Q1. According to Kurose & Ross, UDP protocol provides end-to-end error detection via the 1's complement of the sum of each 16-bit word in a segment (2013). Accordingly, we can split the given segment of '11101111101010111011101110010100' into 16-bit words and then calculate the sum of each. Once this is completed, we can then compute the checksum by taking the complement of the total of the three words as follows:

Step 1: Split the segment into 16-bit words:

1110 1111 1010 1011

1011 1011 1001 0100

Step 2: Add the 16-bit words together starting from the left:

1110 1111 1010 1011

\+ 1011 1011 1001 0100

---

1 1010 1011 0011 1111

Note: There is a carry over from the most significant bit (17th bit in red), so add it back to the sum starting at the LSB:

1010 1011 0011 1111

\+ 0001

1101 1011 1100 0000 (carry over added to LSB)

Step 3: Finally, take the ones' complement of the result (invert each bit), which is the checksum to be included in the UDP segment, which in this case is **0010 0100 0011 1111**.

Q2. With Go-Back-N the sender can send multiple segments before waiting for acknowledgments. With this protocol, if any segment is lost then all subsequent segments must be re-transmitted; However, if a segment is received correctly, then the receiver acknowledges it and the sender can continue sending the next segments (Kurose & Ross, 2013). Given the following parameters stipulated in the problem, we can break down the communication into *segment and acknowledge* rounds per below:

1. Sender sends segments 1, 2, 3, 4, 5 (window size N=5) initially.
2. Receiver acknowledges segments 1, 3, 4, 5.
3. Sender re-transmits segment 2, 3,4, 5, 6.
4. Receiver acknowledges segment 2, 3, 4, 5, 6.
5. Sender continues sending segments 7, 8, 9, 10, 11.
6. Receiver acknowledges segments 7, 8, 9, 10, 11.
7. Sender continues sending segments 12, 13, 14, 15, 16.
8. Receiver acknowledges segments 12, 13, 14, 15, 16.
9. Sender continues sending segments 17, 18, 19, 20, 21.
10. Receiver acknowledges segments 17, 18, 19, 20, 21.
11. Sender sends segments 22, 23, 24.
12. Receiver acknowledges segments 22, 23, 24.

Next, we can calculate the time for each phase:

- Initial transmission and ACK shown in steps 1 & 2 above (segments 1-5) = 1 timeout (30 ms)
- Re-transmission of segment and ACK 2, 3, 4, 5, 6 shown in step 3 & 4 above = 1 RTT (10 ms)
- Sending & ACK if segments 7-11 shown in step 5 & 6 above = 1 RTT (10 ms)

- Sending & ACK segments 12-16 shown in step 7 & 8 above = 1 RTT (10 ms)

- Sending & ACK segments 17-21 shown in step 9 & 10 above = 1 RTT (10 ms)

- Sending & ACK segments 22-24 shown in step 11 & 412above = 1 RTT (10 ms)

Total time = 5 RTTs + 30 ms timeout = 5 * 10 ms + 30 ms

Total time = 80 ms

Next we can calculate the total amount of data transmitted:

- Each segment size = 1500 bytes

- Total segments sent = 24 segments

Total data transmitted = 24 * 1500 bytes = 36,000 bytes

Finally, we can calculate throughput:

Throughput (in bytes per second) = Total data transmitted / Total time (in seconds)

Throughput = 36,000 bytes / (80 ms / 1000) = 36,000 bytes / 0.08 seconds = **450,000 bytes/second or 3.6 Mbps**

However, per my discussion board post to Remesh, this is very confusing indeed...

References

Kurose, J. F., & Ross, K. W. (2013). *Computer networking: A top-down approach* (6th ed.). Pearson.