CSCE 463/612: Networks and Distributed Processing

Homework 3 Part 2 (25 pts)

Due date: 4/8/25

1. Description

This part implements rdt 3.0 with dynamic RTOs, closely following the algorithm in the slides. *The only difference is that your sequence numbers are 4 bytes rather than 1 bit.* To verify the transfer is good, you need to compute a CRC-32 checksum across the sent buffer and compare that value to the one provided by the receiver.

1.1. Code (25 pts)

The seven input parameters are exactly the same as before. You should suppress per-packet printouts (unless debugging) and add output from a stats thread every 2 seconds:

```
Main:   sender W = 1, RTT 0.100 sec, loss 0 / 0, link 14 Mbps
Main:   initializing DWORD array with 2^15 elements... done in 1 ms
Main:   connected to s3.irl.cs.tamu.edu in 0.102 sec, pkt 1472 bytes
[ 2] B     18 (  0.0 MB) N     19 T 0 F 0 W 1 S 0.105 Mbps RTT 0.102
[ 4] B     36 (  0.1 MB) N     37 T 0 F 0 W 1 S 0.105 Mbps RTT 0.102
[ 6] B     55 (  0.1 MB) N     56 T 0 F 0 W 1 S 0.111 Mbps RTT 0.102
[ 8] B     73 (  0.1 MB) N     74 T 0 F 0 W 1 S 0.105 Mbps RTT 0.102
[10.03] <-- FIN-ACK 90 window FC694CF3
Main:   transfer finished in 9.818 sec, 106.80 Kbps, checksum FC694CF3
Main:   estRTT 0.102, ideal rate 114.41 Kbps
```

The first stats line (in bold) shows that 2 seconds have elapsed since the `SenderSocket` constructor was called, the sender base is currently at 18 packets, 0.0 MB of data has been ACKed by the receiver, the next sequence number is 19, there have been 0 packets with timeouts and 0 with fast retransmission, the current *effective* window size is 1 (i.e., the minimum between sender and receiver window), the speed at which the application consumes data at the receiver (i.e., *goodput*) is 0.105 Mbps, and the estimated RTT is 102 ms. Note that the speed is averaged over the period since the last printout (i.e., by subtracting the base values, then multiplying the result by `8 * (MAX_PKT_SIZE - sizeof(SenderDataHeader)`).

Upon receipt of a FIN-ACK, `ss.Close()` prints the FIN's sequence number 90, which is the total number of data packets delivered in this exchange, and the CRC-32 checksum of the data provided by the receiver in the `recvWnd` header of the FIN-ACK packet (in all other cases, `recvWnd` is valid and should be used in flow control). The *elapsed time* reported by main is slightly different from Part 1 – it is the duration between the transmission of the first data packet and receipt of the last data ACK (i.e., both SYN and FIN phases are excluded). Since the application may call `ss.Close()` before the last packet has been acknowledged, this function must pause to collect all outstanding data ACKs (and let the sender perform retransmission as needed) before moving ahead with the FIN. It can then return the elapsed time to main():

```
double elapsedTime;
if ((status = ss.Close (&elapsedTime)) != STATUS_OK)
```

The final line of the trace shows that the latest value of estimated RTT is 102 ms (which is updated using TCP formulas) and the ideal rate under no loss is 114 Kbps (i.e., window / estRTT).

Another example with moderate packet loss:

```
Main:   sender W = 1, RTT 0.100 sec, loss 0.2 / 0, link 14 Mbps
Main:   initializing DWORD array with 2^15 elements... done in 0 ms
Main:   connected to s3.irl.cs.tamu.edu in 0.103 sec, pkt 1472 bytes
[ 2] B      9 (  0.0 MB) N     10 T 4 F 0 W 1 S 0.053 Mbps RTT 0.102
[ 4] B     20 (  0.0 MB) N     21 T 9 F 0 W 1 S 0.064 Mbps RTT 0.102
[ 6] B     29 (  0.0 MB) N     30 T 14 F 0 W 1 S 0.053 Mbps RTT 0.102
[ 8] B     42 (  0.1 MB) N     43 T 18 F 0 W 1 S 0.076 Mbps RTT 0.102
[10] B     60 (  0.1 MB) N     61 T 18 F 0 W 1 S 0.105 Mbps RTT 0.102
[12] B     69 (  0.1 MB) N     70 T 21 F 0 W 1 S 0.053 Mbps RTT 0.102
[14] B     84 (  0.1 MB) N     85 T 23 F 0 W 1 S 0.088 Mbps RTT 0.102
[15.11] <-- FIN-ACK 90 window FC694CF3
Main:   transfer finished in 14.894 sec, 70.40 Kbps, checksum FC694CF3
Main:   estRTT 0.102, ideal rate 114.46 Kbps
```

1.2. ACK Numbers

Note that ACKs are similar to those in TCP – they acknowledge the base of the receiver window, i.e., the next expected packet. Unlike TCP, however, SYN-ACK and FIN-ACK packets do not increment the sequence number. A valid exchange will proceed as following:

```
--> SYN 0
<-- SYN-ACK 0                       // expects packet 0
--> data 0
<-- ACK 1                           // expects packet 1
--> data 1
<-- ACK 2                           // expects packet 2
--> FIN 2
<-- FIN-ACK 2 window = CRC32        // still expects seq 2
```

Use the following function to compute checksums on the buffer you've transmitted:

```
// checksum.cpp
Checksum::Checksum ()
{
    // set up a lookup table for later use
    for (DWORD i = 0; i < 256; i++)
    {
        DWORD c = i;
        for (int j = 0; j < 8; j++) {
            c = (c & 1) ? (0xEDB88320 ^ (c >> 1)) : (c >> 1);
        }

        crc_table[i] = c;
    }
}

DWORD Checksum::CRC32 (unsigned char *buf, size_t len)
{
    DWORD c = 0xFFFFFFFF;
    for (size_t i = 0; i < len; i++)
        c = crc_table [(c ^ buf[i]) & 0xFF] ^ (c >> 8);

    return c ^ 0xFFFFFFFF;
}

// main.cpp
void main (void)
{
    ...                         // send the buffer, close the connection
    Checksum cs;
```

```
    DWORD check = cs.CRC32 (charBuf, byteBufferSize);
}
```

1.3. RTO

Set the RTO for the SYN packet to the maximum of 1 second and `2*lp.RTT`. For RTO estimation during the rest of the transfer, prevent the RTT deviation from falling below 10 ms:

```
RTO = estRTT + 4 * max (devRTT, 0.010);
```

This should keep the RTO at least 40 ms above the mean; otherwise, you are likely to end up with `devRTT = 0` and `RTO = estRTT`, which will lead to frequent spurious timeouts.

463/612 Homework 3 Grade Sheet (Part 2)

Name: _____

Function	Points	Break down	Item	Deduction
Printouts	25	1	Fails to print the stat timer every 2 sec	
		2	Incorrect base (B)	
		2	Incorrect bytes sent	
		2	Incorrect next sequence (N)	
		2	Incorrect timeouts (T)	
		2	Incorrect window (W)	
		2	Incorrect speed (S)	
		2	Incorrect RTT	
		2	Incorrect FIN-ACK (seq, checksum)	
		3	Incorrect summary (delay, rate, checksum)	
		1	Incorrect ideal rate	
		2	Fails to perform as shown in the first example of section 1.1 (no loss)	
		2	Fails to perform as shown in the second example of section 1.1 (20% loss)	

Total points: _____