# C O M P 3 0 8 / S p r i n g 2 0 2 3

**A S S I G N M E N T # 3 – T R A N S P O R T L A Y E R**

P R O B L E M S

# P1) COMPUTING AN INTERNET CHECKSUM [10pts]

Consider the two 16-bit words (shown in binary) below. Recall that to compute the Internet checksum of a set of 16-bit words, we compute the one's complement sum of the two words. That is, we add the two numbers together, making sure that any carry into the 17th bit of this initial sum is added back into the 1's place of the resulting sum); we then take the one's complement of the result. Compute the Internet checksum value for these two 16-bit words:

|  |  |  |
| --- | --- | --- |
| 11001010 | 00010001 | this binary number is 51729 decimal (base 10) |
| 10011101 | 11100101 | this binary number is 40421 decimal (base 10) |

1. What is the sum of these two 16 bit numbers?

1100101000010001 + 1001110111100101= ~~0~~10110011111110110 (17th bit carry)

0110011111110110 + 0000000000000001 = 01100111 11110111

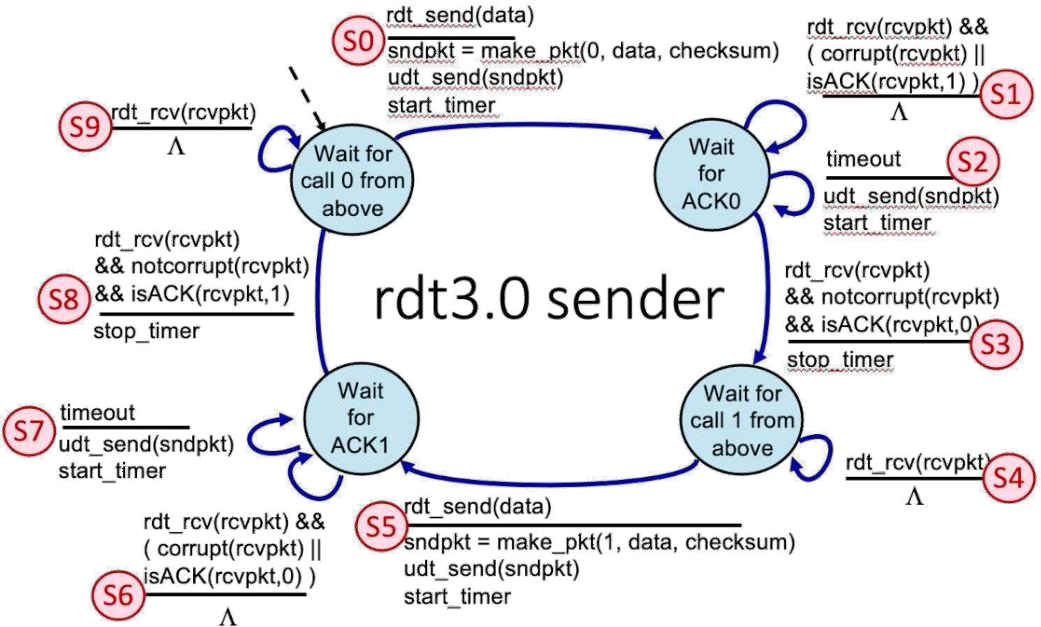
1. Using the sum from question a), what is the checksum?

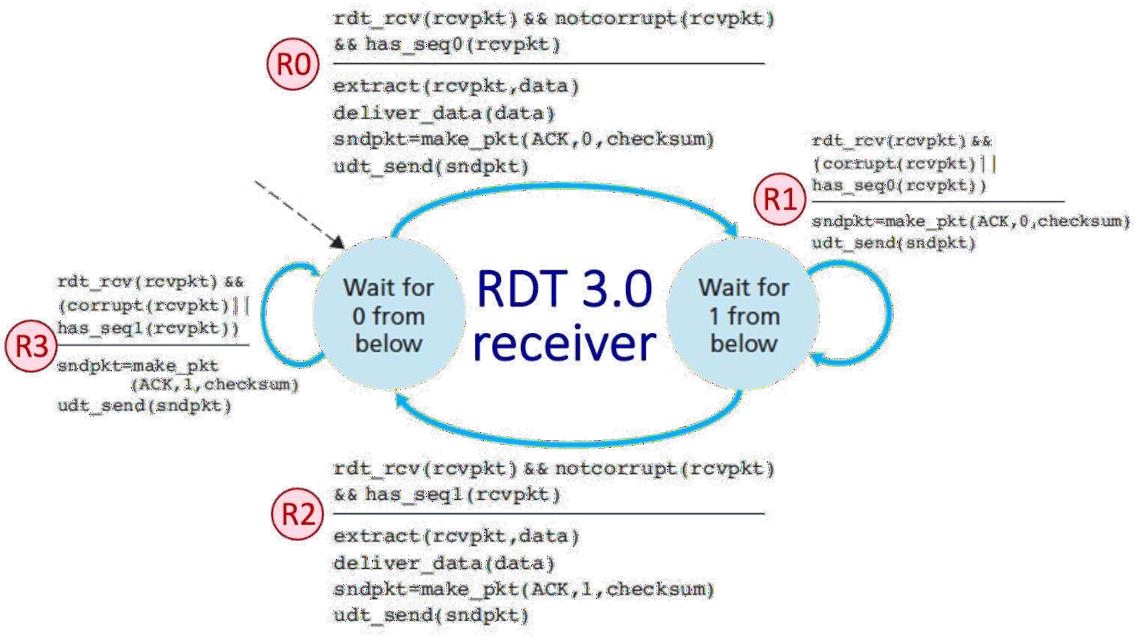
Take one’s complement of the answer for question a (convert 0s to 1s and 1s to 0s)

100111000 00001000

# P2) RELIABLE DATA TRANSFER: RDT 3.0 [30pts]

Consider the RDT 3.0 protocol, for reliably communicating data from a sender to receiver over a channel that can lose or corrupt packets in either direction, and when the maximum delay from sender to receiver and back is not known. The FSMs for the sender and receiver are shown below, with their transitions labeled as SX and RY, respectively.





Now let’s consider the sequence of sender and receiver transitions that would happen when one or more of the following complications occur: a packet (data or ACK) is lost, a timer times out (prematurely or not), or a message is corrupted. One or more of these events has occurred

to produce the sequence of transitions below. Find the missing transition "\*" in the following sequences. To indicate the missing transition, enter S or R, followed by an index.

1. Transition Sequence: S0, S2, \*, S1, S2, R1, S1, S2, R1, S3, S5, R1, S6, S7, R2, S8.

R0

1. Transition Sequence: S0, R0, S3, \*, S7, R2, S8.

S5

1. Transition Sequence: S0, R0, S1, S2, R1, S1, S2, \*, S3, S5, R2, S7, R3, S8

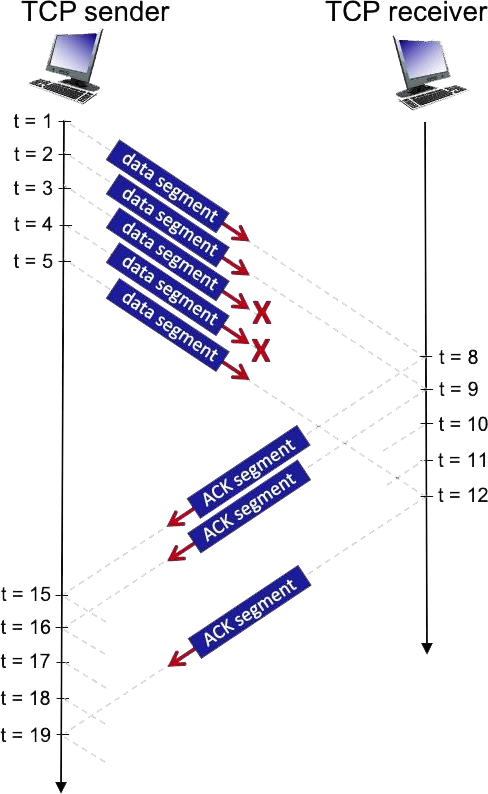
R1

1. Transition Sequence: S0, R0, S3, S4, S5, R2, \*, R3, S8

S6

# P3) TCP RETRANSMISSIONS (RELIABLE DATA TRANSMISSION WITH ACK LOSS) [20pts]

Consider the figure below in which a TCP sender and receiver communicate over a connection in which the segments can be lost. The TCP sender wants to send a total of 10 segments to the receiver and sends an initial window of 5 segments at t = 1, 2, 3, 4, and 5, respectively. Suppose the initial value of the sequence number is 36 and every segment sent to the receiver each contains 276 bytes. The delay between the sender and receiver is 7 time units, and so the first segment arrives at the receiver at t = 8, and an ACK for this segment arrives at t = 15. As shown in the figure, 2 of the 5 segments is lost between the sender and the receiver, but *none* of the ACKs are lost. Assume there are no timeouts and any out of order segments received are thrown out.



1. What are the sequence numbers of the segments sent at each t in [1,2,3,4]?

Sequence number of the segment sent at t=1 is 36

Sequence number of the segment sent at t=2 is 36 + 276 = 312

Sequence number of the segment sent at t=3 is 36 + 2\*276 = 312 + 276 = 588

Sequence number of the segment sent at t=4 is 36 + 3\*276 = 588+ 276 = 864

Sequence number of the segment sent at t=5 is 36 + 4\*276 = 864 + 276 = 1140

1. What are the values of the ACKs sent at each t in [8,9,10,11,12]? (If segment lost, write 'x')

The value of the ACK sent at t=8 is 312

The value of the ACK sent at t=9 is 36 + 276 = 588

The value of the ACK sent at t=10 is x

The value of the ACK sent at t=11 is x

The value of the ACK sent at t=12is 36 + 276 = 588

1. What are the sequences numbers of the segments sent at each t in [15,16,17,18,19]? (If ACK never arrives, write 'x')

Sequence number of the segment sent at t=15 is 1140 + 276 = 1416

Sequence number of the segment sent at t=16 is 1416 + 276 = 1692

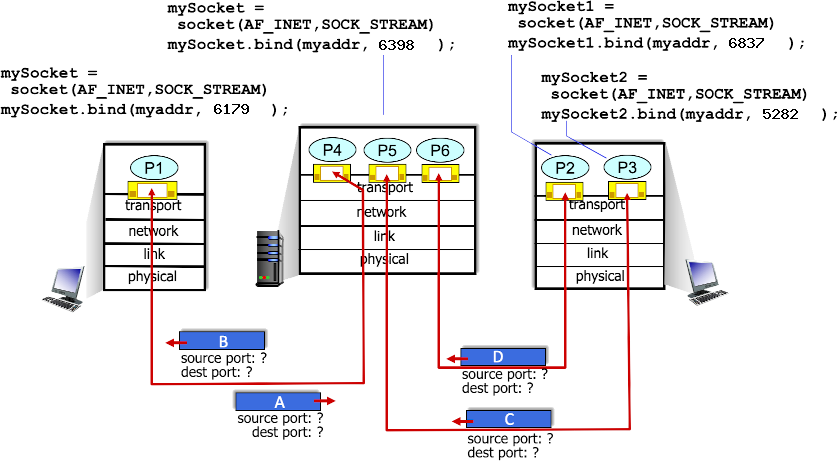
Sequence number of the segment sent at t=17 is x

Sequence number of the segment sent at t=18 is x

Sequence number of the segment sent at t=19 is x

# P4) TCP MULTIPLEXING AND DEMULTIPLEXING [10pts]

In the scenario below, the left and right TCP clients communicate with a TCP server using TCP sockets. The Python code used to create a single welcoming socket in the server is shown in the figure (the welcoming socket itself is not shown graphically); code is also shown for the client sockets as well. The three sockets shown in server were created as a result of the server accepting connection requests on this welcoming socket from the two clients (one connection from the client on the left, and two connections from the client on the right).



* 1. What is the source and destination port # for packet B?

the source port # for packet B: 6398

the destination port # for packet B: 6179

* 1. What is the source and destination port # for packet D?

the source port # for packet D: 6837

the destination port # for packet D: 6398

* 1. What is the source and destination port # for packet C?

the source port # for packet C: 5282

the destination port # for packet C: 6398

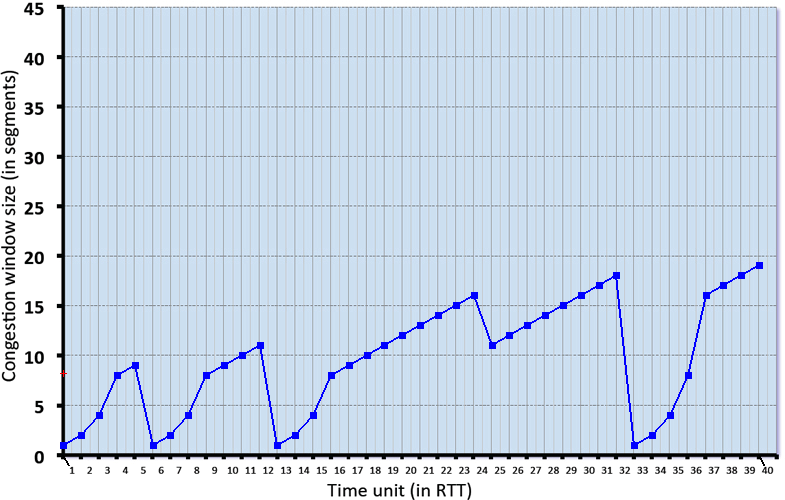
* 1. What is the source and destination port # for packet A?

the source port # for packet A: 6179

the destination port # for packet A: 6398

**P5) TCP IN ACTION: SLOW START, CONGESTION AVOIDANCE, AND FAST RETRANSMIT [30pts]**

Consider the figure below, which plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT); see Figure 3.53 in the text. In the abstract model for this problem, TCP sends a "flight" of packets of size *cwnd* at the beginning of each time unit. The result of sending that flight of packets is that either *(i)* all packets are ACKed at the end of the time unit, *(ii)* there is a timeout for the first packet, or *(iii)* there is a triple duplicate ACK for the first packet. In this problem, you are asked to reconstruct the sequence of events (ACKs, losses) that resulted in the evolution of TCP's *cwnd* shown below.





**Threshold** (red line)

Until threshold point, cwnd will be increased exponentially (1,2,4,8) (aka slow start). After passing that threshold, it will be increased linearly (aka congestion avoidance). Threshold changes when avoidance or timeout event occurs. (I did not include all thresholds)

**Timeout 🡪** Cut cwnd to 1. (Purple line)

**3 Duplicate ACKs 🡪** Cut cwnd in half.

**Purple Rectangles** **🡪** congestion avoidance

**Green Rectangles 🡪** slow start

**Fast recovery 🡪** after triple ACKs occurred. (Orange line)

Consider the evolution of TCP's congestion window in the example above and answer the following questions (Format your answer like: 1,3,5,9). The initial value of *cwnd* is 1 and the initial value

of *ssthresh* (shown as a red +) is 8.

* + 1. Give the times at which TCP is in slow start.

1,2,3,6,7,8,13,14,15,33,34,35,36

* + 1. Give the times at which TCP is in congestion avoidance.

4,9,10,11,16,17,18,19,20,21,22,23,24,25,26,27,28,29,39,31,37,38,39,40

* + 1. Give the times at which TCP is in fast recovery.

25

* + 1. Give the times at which packets are lost via timeout.

5,12,32

* + 1. Give the times at which packets are lost via *triple ACK*.

24

* + 1. Give the times at which the value of *ssthresh* changes (if it changes between t=3 and t=4, use t=4 in your answer).

6,13,25,33