CMSC 417 Computer Networks **Spring 2009**

Second Third-Term Exam

Closed book and notes; In class *Thursday, April 9th*

⊕ *Do not forget to write your name on the first page. Initial each subsequent page.*
⊕ *Be* **neat** *and* **precise**. **I will not grade answers I cannot read.**
⊕ *You should draw simple figures if you think it will make your answers clearer.*
⊕ *Good luck and remember, brevity is the soul of wit*

- All problems are mandatory
- I cannot stress this point enough: **Be precise**. If you have written something incorrect along with the correct answer, you should **not** expect to get all the points. I will grade based upon what you **wrote**, not what you **meant**.
- Maximum possible points: 50.

Name: _____

Problem	Points
1	
2	
3	
4	
5	
Total	

1. Describe the following terms (2 points each)

   (a) Incarnation number

   (b) Go-back N

   (c) Self-clocking

   (d) Flow Control

   (e) AIMD

2. Multicast

(a) How does DVMRP reverse-path broadcast differ from link-state flooding? (3 points)

(b) What shortcoming of DVMRP does PIM-SM solve? (3 points)

(c) How are (S,G) joins used in PIM-SM? Why? (4 points)

3. Transport Protocols

(a) Why is "source port" part of the UDP header? (2 points)

(b) What is the difference between congestion control and congestion avoidance? (2 points)

(c) Give an example where the Alternating Bit Protocol fails if ACKs do not have sequence numbers. (3 points)

(d) Give an example where a selective acknowledgment performs better than a cumulative acknowledgment. (3 points)

4. TCP

(a) Describe the state transitions and messages sent during a TCP simultaneous open. (3 points)

(b) Why is the `TIME_WAIT` state required in TCP? (2 points)

(c) How does Fast Retransmit improve TCP performance? (3 points)

(d) When is slow-start required during a TCP connection? (2 points)

5. You're implementing a reliable packet transport protocol over UDP. Your protocol implementation runs as an application over POSIX and uses the sockets interface to communicate with UDP. Applications (that use your protocol) communicate with your process using two queues kept in shared memory. For each connection, you maintain the following protocol state in a connection specific PCB (protocol control block):

   `output`, `input` : queues used to communicate with application
   each queue supports `q.empty?` `q.full?` `q.enque()` and `q.deque()` calls

   `sd` : UDP socket

   `SWS, RWS, LAR, LAF, LFS, LRF` : variables for sliding windows

   ...

   - Describe other variables you need to maintain for each PCB
   - Write the prototypes and a short description for auxiliary functions you will need to implement reliable delivery (no need to provide implementations)
   - Assume PCBs for active connections are kept in a global array `C`. You can access the input queue for the ith active connection using the notation `C[i].input`

     Write psuedocode for the main packet processing loop. You may refer to any function you have defined, any C library call, and any system call. Your code should allow a `housekeeping()` function to be called every 500 milliseconds (10 points)

     Alternately, instead of an array of PCBs, provide the solution assuming *only one* active PCB `C`. The input queue in this case would be accessed using `C.input`. Again, `housekeeping()` should be called every 500 milliseconds. (**7** points)

     In neither case does your solution need to cover connection setup or teardown.