CS 603 (Park) **Assignment II** **Due: Feb. 6 (Fri.), 1998**

PROBLEM 1

Read Section 3.1 from Tanenbaum. Read "Distributed Computing" by J. Stankovic in *Readings in Distributed Systems*, pp. 6–30, and give a 1-page summary/critique. Make sure to include your point-of-view regarding the pros/cons of the paper.

PROBLEM 2

(a) Give a complete proof of the safety property for the decentralized total order multicast algorithm described in class. Make sure to consider all cases.

(b) Give a proof of the liveness property for the decentralized total order multicast algorithm.

(c) Why is showing safety and liveness equivalent to showing that the algorithm works correctly? First, define what it means for an algorithm to work correctly in the multicast context, and then relate the latter to safety and liveness.

PROBLEM 3

Implement the centralized total order multicast algorithm discussed in class. Create two library routines

`total_order_multicast_send`(), `total_order_multicast_recv`(),

through which an application interfaces to the multicast enabling middleware. Both functions take three arguments, the first indicating a 4-byte integer group ID, the second indicating a pointer to a read/write buffer, and the third indicating the size of the multicast message to be read/written. `total_order_multicast_send`() is a nonblocking call whereas `total_order_multicast_recv`() is blocking. Both are to return a -1 status if an error condition occurs, and the number of bytes received/sent, otherwise.

The multicast enabling middleware consists of a single daemon, `tom_server_d`, which performs the function of the multicast manager. `total_order_multicast_send`() triggers the sending of a multicast message request to `tom_server_d` where it is suitably queued and processed. All communication are to use UDP with a timeout mechanism set at 2 seconds to trigger retransmission of lost messages. Sequence numbers must be used to detect and discard duplicate messages. `total_order_multicast_recv`() opens an UDP socket on port $\mathcal{P}_1$ waiting for a multicast message to arrive. Upon proper processing of an arriving multicast message, `total_order_multicast_recv`() returns to the application closing down the UDP socket.

Test your application in the following set-up. The application to be tested has the basic form

```
for i ← 1 to 20 do
    buf ← "i"
    total_order_multicast_send(10, buf, |buf|)
    total_order_multicast_recv(10, buf, |buf|)
    print(buf)
```

Run the multicast server on xinu_i and run five multicast group clients on xinu_{i+1}, xinu_{i+2}, xinu_{i+3}, xinu_{i+4}, xinu_{i+5}. For this set-up, hardcode the mapping from the group ID "10" to the multicast manager's IP address/port number in the library function `total_order_multicast_send`(). Similarly, hardcode the multicast group members' IP addresses/port numbers inside `tom_server_d`. Repeat the experiment three times and verify that total order multicast is correctly achieved.