CS 536 (Park) **Assignment VI** **Due: Oct. 10 (Fri.), 1997**

PROBLEM 1

Read Sections 4.4, 5.1, 5.2, and 5.6 from P & D.

PROBLEM 2

Extend the client/server application of Problem 2(a) of Assignment V such that client and server processes are allowed to run on different hosts and communication is achieved via socket calls invoking UDP/IP. Thus the FIFO IPC structure needs to be replaced by the appropriate networking calls. Note that UDP is a connectionless (i.e., stateless), unreliable transport protocol. Test your application using 4 machines in the Xinu Lab, one on which the server runs, and three on which client processes run. Assume the server listens on a well-known port, say 7997. A client should accept two command-line arguments

% `remote_exec` *IP-address command*

where *IP-address* is in dotted decimal form. To prevent indefinite waiting from occuring when packets are lost or the server is down, the client, upon making a request to the server, should set an alarm to go off in x msec. When SIGALRM is caught, the client should resend the request with the entire procedure repeating at most 5 times. On your host machine, open four windows, one for each separate host, running the server on one machine and three copies of the client on the other machines. Test the system by giving `host`, `date`, and `ls -ld` as arguments to the clients (run the three commands consecutively on each host). For the first set of runs, set $x = 5$ sec. For the second set of runs, set $x = 50$ msec and compare the two trace behaviors. In addition to the code and script files, hand in a screen dump which shows the contents of the four windows.

PROBLEM 3

Repeat Problem 2, however, with socket calls invoking TCP in place of UDP. Do not remove the timeout mechanism since it can still check for the server being down or not. Which communication mechanism is more suited for this client/server application? Justify your answer.

PROBLEM 4

Implement your design of Problem 4 of Assignment V. Use UDP as your transport protocol where a token is just a UDP packet that is sent from station to station in a logical ring. Assume a 5-station logical ring and test your system by having each station i try to send 1 MB of data to station j where $j = (i+2) \bmod 5$. Measure the average, max, and min completion time for tranferring the files where the average/max/min operations are taken over the set of clients partaking in the run. Note that recovery from token loss is an integral component of this system. Your solution need not engage in a full-fledged leader election process to work.