CS 536 (Park) **Assignment VI** **Due: Oct. 21 (Wed.), 1998**

PROBLEM 1

(a) Read Sections 5.1, 5.2 of P & D.

(b) (b) Read "The design philosophy of the DARPA internet protocols" by D. Clark. In *Proc. ACM SIGCOMM '88*, pages 106-114, 1988. Give a one-page summary and critique of the paper (i.e., do you agree with the design paradigm?).

PROBLEM 2

(a) Implement a *barrier* mechanism using the notion of a monitor, in our case, implemented as an iterative server. The *barrier server* is to have a well-known request FIFO to which clients trying to synchronize register by issueing a `barrier` function call. This should enqueue the client's process ID into the FIFO. After *all* clients have issued `barrier` calls, the server should send a suitable *go-ahead* message back to the blocked clients, allowing them to continue. This feedback should also be implemented as a FIFO, similarly to the client/server example of Assignment IV.

The driver portion of the client code should consist of a `for`-loop iterating 10 times, at each iteration outputting its process ID and iteration count followed by the `barrier` call. `barrier` is the only function besides the driver codes for the server and client, and it has type definition

void `barrier`(*void*)

First run the barrier server in the background, then start up five copies of the client in the background. Assume, for now, that the number "five" is hardcoded into the server. Assuming the process IDs of the five processes are 100, 200, 300, 400, and 500, respectively, the output of a sample run may look as follows:

100(1) 200(1) 300(1) 400(1) 500(1)
200(2) 100(2) 300(2) 400(2) 500(2)
100(3) 200(3) 500(3) 400(3) 300(3)
100(4) 300(4) 200(4) 400(4) 500(4)
200(5) 100(5) 300(5) 400(5) 500(5)
200(6) 100(6) 400(6) 300(6) 500(6)
100(7) 200(7) 300(7) 400(7) 500(7)
100(8) 200(8) 300(8) 400(8) 500(8)
200(9) 500(9) 300(9) 400(9) 100(9)
200(10) 100(10) 300(10) 500(10) 400(10)

(b) Remove the hardcoded feature of "five" from the server in the following way. The server detects the number of clients participating in the barrier by noting the time when the first client message arrives and waiting for further client registrations for a time interval of $\mathcal{T}$ seconds where $\mathcal{T}$ is a command-line argument to the server. After $\mathcal{T}$ seconds have elapsed from the first client request, the server notices how many clients have registered and only provides barrier service to those ignoring late arrivals. That is, if others try to join late and make requests subsequently, then by inspecting their request format, they are correctly identified as such and ignored.

Demonstrate the correct working of your modified server by executing your server with four times with command-line arguments 15, 10, 5, 1. Each time, type the commands to run five copies of the clients in the background as fast as you can and show the resulting output trace. Check that your traces demonstrate the correct working of your "timed" barrier server.

PROBLEM 3

(a) Give a precise algorithmic description of DQDB's MAC protocol based on the discussion in class. Recall that the protocol shown in the lecture notes is incomplete. Write the pseudo code for three functions `DQDB_send`(), `DQDB_listen_req`(), and `DQDB_listen_data`(). `DQDB_send`() is called by higher layer protocols which passes a frame to be sent. `DQDB_listen_req`() "listens" for marked request frames passes on the control bus (assume, for simplicity, that data communication occurs in one way only) and is triggered or executed each time a marked request frame passes by. `DQDB_listen_data`() "listens" correspondingly on the data bus and is triggered when empty data frames

are passing by.

(b) Assume you have a DQDB network with 2 hosts attached to the system. Supposing that hosts can be modeled as *infinite sources* and all stations send data in one direction only (the same direction for all stations), what are the throughputs (in units of frames or slots per unit time or second) achieved by the upstream and downstream nodes? Here, throughput is to be measured as time $t \rightarrow \infty$. Does DQDB's MAC protocol give fair access? What is the system utilization on the data bus? Generalize your analysis to $n \geq 1$ hosts.