C S 380D (53640)
DISTRIBUTED COMPUTING I
Spring 2013

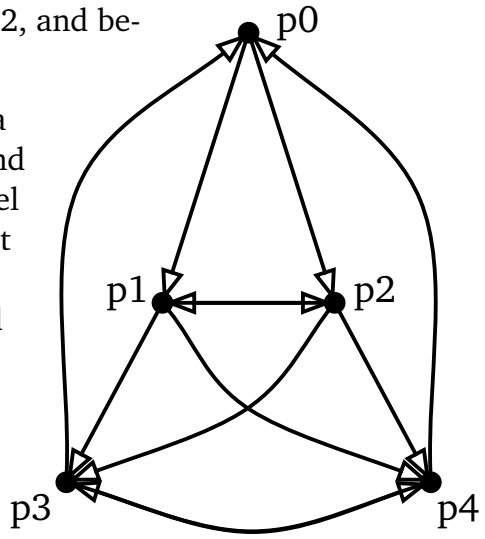
Project 1

Create an Orc language program that conforms to the following specification. When complete, turn in the .orc file using the `turnin` utility.

Specification

[Suggested implementation approach: get items 1–5 working, then add subsequent items one at a time.]

1. Create the network shown in the diagram below. Each vertex is a process, and each (directed) edge is a channel. Each channel is a unbounded FIFO (order-preserving) unidirectional link between two processes. Note that there are two edges, one in each direction between p1–p2, and between p3–p4.



2. A process executes a *round* by: (a) generating a random number (in a range of, say, 0...100), and (b) sending it once along each outgoing channel at random moments in time. A process does not read its incoming channels during a round. Choose the random times such that each round is completed within 50 ms.
3. The p0 node is the initiator (master) process, and the remaining ones are slaves. Node p0 starts the computation by running a round. Subsequently, it behaves like a slave (see below).
4. A slave waits to receive a message from any of its input channels. Upon receiving a message from some channel, it starts a round. It does not process a message from any other channel until the round is complete. Make sure that no incoming message is dropped or ignored.
5. Modify the program so that each message carries a single integer logical time stamp, so that each transmitted message is of the form (*time-stamp*, *value*).
6. Modify the program, yet again, so that each process has a state, an integer, so that the following invariant is maintained whenever *all* processes have completed their rounds (in a perpetual computation, like the one described here, such moments are rare, but you have to account for the possibility):
   sum of all process states = sum of values of all messages that are still in channels.
7. Modify the program, yet again, so that a global snapshot is initiated after 1 second by the initiator. (Processes do snapshot computations only between rounds, never within a round.)

8. Modify the program, yet again, so that after 1.5 seconds, the initiator sends a message to the network, in "diffusing computation style", requesting that all processes start recomputing from the snapshot state. "Diffusing computation style" means that each process notifies its neighbors. This is similar to a distributed snapshot protocol.

Resources

The *Orc Reference Manual* contains descriptions of all the language features and the standard library sites.
URL: http://orc.csres.utexas.edu/documentation/html/refmanual/index.html

The *Structured Concurrent Programming* book has several chapters that are useful for this project. In particular, see chapter 8, Programming with Channels.
URL: http://www.cs.utexas.edu/~misra/temporaryFiles.dir/Orc.pdf

Submission

Submit the project on on a department-managed Linux host by running the command "`turnin --submit jthywiss cs380d_proj1 file...`", where *`file...`* is a space-separated list of files to submit.

Run "`man turnin`" for more information on the `turnin` utility.

Evaluation

The submitted program will be evaluated for correctness, design quality, and appropriate use of the language and library.