MIDTERM EXAM (10/22/98)

COP 6617 Distributed System Design

1. (25 pts) Implement the following recursive equation using DCDL:

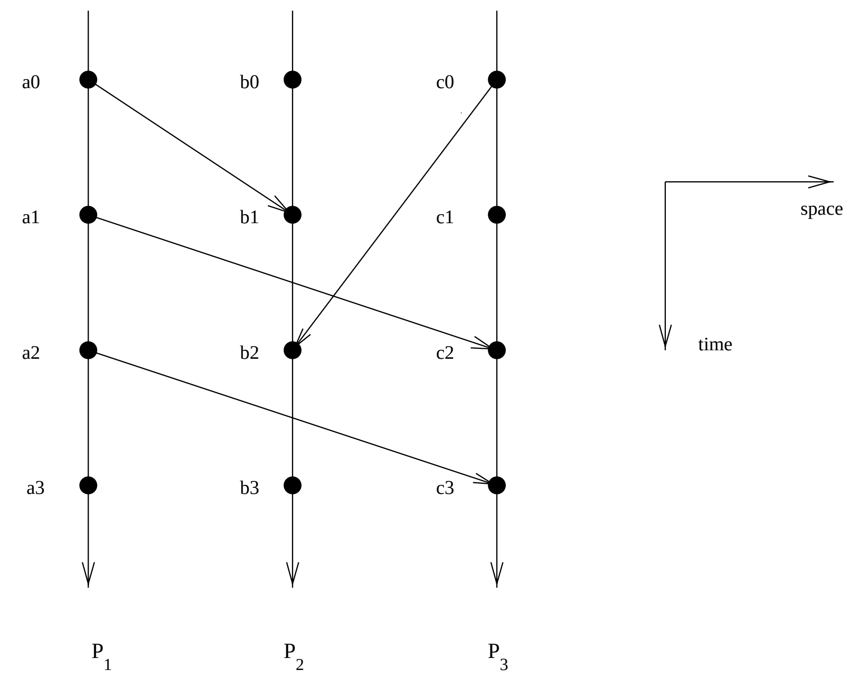
$$f(n) = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ f(n) = f(n-1) \times f(n-1) & n > 2 \text{ and } n \text{ is odd} \\ f(n) = f(n-2) \times f(n-1) & n > 2 \text{ and } n \text{ is even} \end{cases}$$

   That is, $f(3) = f(2) \times f(2), f(4) = f(2) \times f(3), f(5) = f(4) \times f(4), f(6) = f(4) \times f(5), f(7) = f(6) \times f(6)$, etc.

   (a) Use DCDL to write a code for $f(n)$. You are required to **use one process in computing** $f(i)$ and one process P_0 for the user (who provides n). It is assumed that each P_i can only communicate with its two neighbors P_{i-1} and P_{i+1}.

   (b) Demonstrate your algorithm by calculating $f(6)$.

2. (25 pts) For the distributed system shown in the figure below,



(a) provide logical time for all the events using

(i) linear time, and

(ii) vector time.

Assume that each LC_i is initialized zero and $d_1 = 1$, $d_2 = 2$. d_3 is a variable which is initialized one. After each event at P_3, d_3 increments by one, i.e., $d_3 = 2$ after c_0 and $d_3 = 3$ after c_1.

(b) does condition $a \rightarrow b \Rightarrow LC(a) < LC(b)$ still hold? For any other set of d's that includes one or more d's that are variables? and why?

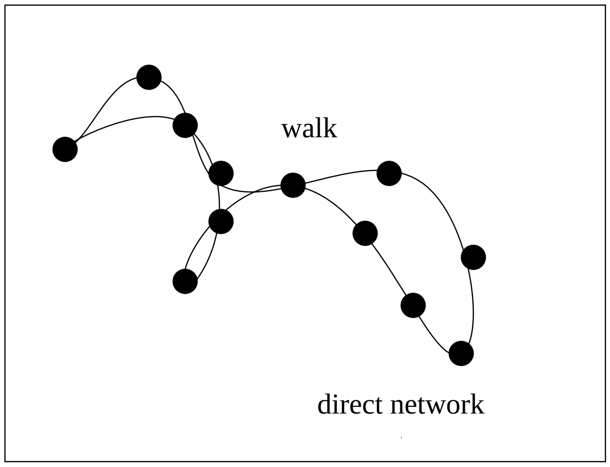


Figure 1: A walk in a given direct network.

3. (25 pts) Misra's fault-tolerant ping-pong algorithm can be directly applied to any direct network that is Hamiltonian (a graph in which there exists a path that visits each node once and only once; moreover, this path starts from and ends with the same node). In a direct network that is not Hamiltonian, we can construct a *walk* that starts from and ends with the same node. However, some nodes may be visited more than once (see the example in the figure).

   (a) What are the potential problems if the regular ping-pong algorithm is directly applied by using a talk as its logical ring?

   (b) Propose an extension of the ping-pong algorithm. Describe your extension in **plain English**. There is no need to provide the actual code. However, explicitly list variables needed at each process (node).

process id	priority	1st request time	length	retry interval	resource(s)
P_1	3	1	1	1	A
P_2	4	1.5	2	1	B
P_3	1	2.5	2	2	A, B
P_4	2	3	1	1	B, A

Table 1: A system consisting of four processes.

4. (25 pts) Show the resource allocation time for each resource (A and B) and for each of the four processes in the given table when the **wait-die scheme** is used. Note that P_3 requests both resources A and B (in the order A and then B). P_4 also requests both resources but in a reversed order. Assume a process immediately requests the second resource once the first resource is granted. However, execution starts only when both resources are acquired. Once a process is killed it should release all the acquired locks (if any).