22C: 166 Distributed Systems and Algorithms
Homework 2, Total points = 60

Assigned 9/20/12 due 9/27/12

Please submit typewritten solutions through ICON, preferably in the pdf format. Late assignments will not be accepted without prior approval.

Question 1 (10 points): A *round-robin scheduler* guarantees that between two consecutive actions by the same process, every other process with enabled guards executes their action at least once.

Consider a *completely connected network* of n processes numbered 0 through n-1. Assuming that only one process is scheduled at any time, implement a round-robin scheduler. Provide brief arguments in support of your implementation.

Question 2 (25 points): The *dining philosophers' problem* is a classic problem in concurrent programming. Dijkstra designed this problem in 1965 as a class exercise for students, and since then numerous researchers have studied problem and its variations. Here is a statement of the problem:

Five philosophers sit at a table around a bowl of spaghetti. A fork is placed between each pair of adjacent philosophers. Each philosopher must alternately *think* and *eat*. Eating is not limited by the amount of spaghetti left: assume an infinite supply. However, a philosopher can only eat while holding both the forks to the left and to the right. Each philosopher can pick up an adjacent fork, when available, and put it down, when done with it. These are separate actions: forks must be picked up and put down one by one. The problem is how to design a discipline of behavior (by specifying the program for a philosopher) such that each philosopher can forever continue to alternate between eating and thinking, and thus does not starve to death. Do not use global variable, or a global cache, since that would spoil the fun. (Feel free to look at the Wikipedia page for discussion on this problem).

(a) First, think of a simple approach: let each philosopher, when he/she becomes hungry, grabs the left fork first and then the right fork (as they become available). After finishing eating, each philosopher puts down the right fork first and then the left fork. Does it work? Why or why not?

(b) Now, using the *message-passing model*, propose a solution of the dining philosophers problem. You can assume that a philosopher can send a time-stamped request to the left neighbor for the left fork and to the right neighbor for the right fork. Note that each philosopher can only communicate with his/her right and left neighbors. Briefly argue why your solution will work. Also, comment on whether your solution is starvation free, and add a brief justification.

Question 3 (20 points): Consider a system of *unbounded clocks* ticking in unison, i.e. at the *same rate*, and displaying the same value. Due to electrical disturbances, the phases of these clocks might occasionally be perturbed. The following program claims to synchronize their phases in a bounded number of steps following a perturbation:

{program for clock i}
define c[i] : integer {non-negative integer representing value of clock i}
{N(i) denotes the set of neighbors of clock i}
do true $\rightarrow$ c[i] := 1 + max {c[j] : j $\in$ N(i) $\cup$ i} **od**

Assuming a synchronous model where all clocks execute the above action with each tick, and this action requires zero time to complete, verify if the claim is correct. Prove using a *well-founded set* and an appropriate variant function that the clocks will be synchronized in a bounded number of steps. What is the round complexity of the algorithm?