Due data: 1397/10/09

Emulated Bellman-Ford Algorithm

Assume that you are given a file (adj_mat.txt) like the attached file. It shows the adjacency matric of routers in a network (the value on a_{ij} shows the cost of the link between node i and node j. If $a_{ij} = \infty$ it means that there is no direct link between i and j).
There is also another file (which_port.txt), which shows which router (terminal) is listening to which port.

Now you write a program that acts as a router.
So, based on the number of columns in the file (assume **k**), you open **k** terminals and on open you tell the ID of the router that it should emulate. For example, after execution, the terminal asks: which router am I? and you give it a number.

Then the program will go and read "adj_mat.txt" and find out the cost that it has from other routers (it will read only the line that it corresponding to itself).

All terminals also read (which_port.txt) and learn about the port number that each of the other terminals (routers) will listen into.

a) After you setup all the terminals, you press "S" in each of them so they start the Bellman-Ford algorithm. Then they start sending their distance vector to their neighbors and update their own vector upon receipt of a new message from others. It should continue this process until it gets to the stable version (it gets no new update message from others).
After each update, the terminal should show the updated distance vector and the messages it has already received from others (like the tables that I have shown you in class for each router).

b) After it reaches to the stable state, the emulated router (terminal) should perform two tasks:
1) Always listen to its port for possible updates from other routers
2) Every 10 seconds, the emulated router (terminal) should go and again check the corresponding line of the "adj_mat.txt" to see if its link cost has been changed or not. If it has been changed it should see if anything changes in the distance vector and if it has changed it should send the update to its neighbors.

Notes:
1) use poisoned reverse method to prevent the count to infinity problem.
2) The connection between the terminals can be implemented as a TCP or UDP, whichever is easier for you.
You can submit this assignment as a group but each groups should be at most 2 persons.

Good Luck