Distributed Systems

Assignment

Rik Sarkar
Edinburgh Fall 2014

University of Edinburgh

Programming Assignment

- Simulations
- Part 1:
  - Input: Coordinates of wireless nodes
  - Output: Minimum spanning tree
  - Method: GHS (synchronous version)

- Part 2:
  - Series of large broadcasts from individual node to all other nodes
  - Keep track of energy
  - A node dies when it runs out of energy

Simulation

- Often a preparatory stage before deploying read distributed programs
- A real system is hard to configure, deploy, maintain
- Important to be confident that your algorithm works and know its properties before taking the trouble of installing it
- Also makes it possible to try many different systems without actually building them

Simulation

- Simulate the effects of having multiple processes and message-based communication
- Using only one computer
- So you have to consider how the processes, communication etc will behave in a real distributed system

Task 1

- Read the input file
- Contains:
  - Node id, coordinates, energy
  - Broadcast transmission list

- Sample files would be uploaded so you can check your input.

- Also check web page regularly and use the most up to date version of the pdf. We may update it with small changes/clarifications.

Task 2

- Create "nodes" according to input file
  - Use suitable data structure

Communication model

- Each node can communicate with nodes in distance 10
  - Ensure that this is guaranteed
- Types of communication:
  - "discover" message: Goes to all nodes in R=10
    - For finding out who your neighbors are
    - Cannot be used for anything else
  - "neighbor" or point to point messages
    - Useful in all other communications
- Note: we are assuming perfect communication in both cases
- You do not have to simulate MAC layer issues
  - Collisions, message loss, retransmissions etc..

Task 3

- Build the network
- Every node needs to “know” its neighbors
  - Constant amount of info per neighbor

Synchronous operation

- See slides from last class
- Operate in rounds
- Assume all messages are delivered simultaneously for the same round

Synchronous operation

- See slides from last class
- Operate in rounds
- Assume all messages are delivered simultaneously for the same round
- Does not have to actually happen simultaneously, you just have to “simulate” that effect

Distributed operation

- Options:
  - Use one thread for each process
  - Simulate multiple processes inside the same thread with suitable data structure and procedures

Network simulation

- May be useful to have specialized code to simulate operation of network and communication

We are assuming a “base station” exists

- Can broadcast to all nodes
- Can automatically detect end of certain things like a flood or a level in GHS

Task 4

- Compute MST [See Lynch for description of SynchGHS]
- Start with n connected components
- Run in levels
  - Each connected component is a tree
  - Find the smallest weight edge (weight=length) to a different component
  - Add it to the MST
  - Use convergecast for the data collection to leader
    - (use flood in the tree if convergecast feels hard)

Task 4

- Compute MST
- Assume that base station knows when a level has ended, and broadcasts a message to all nodes to start next level
- This can be done with termination detection, but we are skipping that for the assignment, and assuming base station knows when level has terminated
- It is also possible to program GHS without this termination detection. You are free to do that too!
- If you are using a broadcast in a tree to spread some kind of info, you can assume that the bsae station knows when this has terminated

Task 5

- Produce Log file

Your simulator should produce a log file with the following information describing the progress of the algorithm:

when the base-station alerts the leaders to proceed to the next level:

 bs {NodeID1}, {NodeID2}, ….

{NodeID} being the id of each node the base-station is communicating to.

when a new leader is elected:

 elected {NodeID}

 {NodeID} is the id of the newly elected leader.

when a new edge is added to a connected component

 added {NodeID1} - {NodeID2}

 {NodeID} is the id of each the nodes at the end of the edge.

Your log file should follow the exact format given in assignment, including keywords. These will be evaluated by an automated checker script which will not be able to read the logs if they do not conform to the format.

Part 2: Energy bounded broadcast

- Problem: several nodes have large data to broadcast
- The broadcast needs to go to all nodes
- But not all nodes are within range 10 of source
- So we need to send messages going in multiple hops
- How you do this is up to you

Part 2: the problem:

- Each node has limited energy in its battery
- Sending the data from one node to another consumes energy proportional to the edge weight

- When a node's energy goes below a certain threshold, it "dies" and is no longer part of the network

Part 2

- When a node dies, its neighbor know
- E.g. they can find out by sending messages and not getting replies
- Or the node can send them messages before shutting down.
- Assume that base station knows when a broadcast has completed
- It initiates the next broadcast only after that

Part 2

- Your task: keep nodes alive as long as possible

Submission

- Source code
  - Must be C, C++, Java or Python
  - Must run on DICE
  - Not executable
  - We will evaluate your code
  - So make it readable
  - Write clearly structured code that is easy to understand
  - Write comments that help to understand your code
  - Write comments and good code from the start. Do not keep it for the last day

Submission

- A shell script called run.sh that compiles and runs your code.
- Takes input file as command line argument
  - Eg. ./run.sh input.txt

Submission

- A shell script called run.sh that compiles and runs your code.
- Takes input file as command line argument
  - Eg. ./run.sh input.txt

README file

- Explain the design
- What modeling decision you made
- What your algorithmic decisions are
- What strategy you are using for part 2
- What strategy you are using when a node dies
- …
- Anything else you would like us to know

- **Max 1 page in 12 point font, 1 inch margin**

Theory submission

- Pdf file
- With your uun (see document for format)

Submission

- Put everything: theory & programming
- In 1 zip file with you uun (see format)

- Use submit command on DICE

- Resubmit: Just submit again in time. Old file will be overwritten

Feedback

- In 3 weeks
- Programming
  - Short description of where you lost points
  - Anything specific that comes to notice
  - We may decide to call you discuss your code and its performance
    - But we cannot grade your answers, so make sure your README, code, comments are good.
- Theory:
  - Usual marking for written answers

Collaboration Vs Plagiarism

- Make sure your code is your own
  - "You can Google. So can we."
  - You need to be able to explain it in comments and description and when asked in discussion
- You are allowed to discuss with classmates
  - Have to disclose it
  - But submissions should not be too similar
  - It can have "general" similarity but not "pivotal" ones
  - Suggestion: if you feel the discussion is getting too specific so that two of you will have the same basic startegies, then stop the discussion.

Assignment

- Start NOW
- Programming, debugging will take time.
- GHS is is not very easy algorithm
- Finding a strategy for part 2 will take time
- Don’t leave the theory part for the last day
- Make sure you code runs on DICE