

Course ISE 431: Distributed Information Systems

Recitation 11 Exercise

Michael J. May

December 29, 2009

We have learned about synchronization and ordered communication in the past few lectures. We will be developing a couple of sample applications based on what we have learned, but in order to do so we must first develop a mechanism for multicast communication.

1 Approaches to Multicasting

Multicast communication can be done in one of three ways:

1. Each node subscribes to listen on a special multicast address. The messages are sent via UDP or some other connectionless protocol (meaning that we don't have reliable delivery). The routers/switches flood the multicast communications to all nodes (or a selection of them if they are using a multicast subscription protocol) in case one of them is interested in it.
2. To support connection-oriented protocols, special multicast routers may act as end points for multicast messages and forward them on in a point-to-point manner to all available nodes. Effectively, the routers do an exhaustive flooding.
3. For connection-protocols without router support, the most effective mechanism is a point-to-point multicast tree (a form of an overlay network).

Since I am unsure about whether the router in 202 will block multicast addresses, I am fairly certain it will not act as a distribution point, and in any case we are interested in reliable multicast we will work on the third option: implementing multicast by a point-to-point multicast distribution tree.

2 Designing the Tree

To make a multicast tree, we begin with a single node which will act as the **root** of the multicast tree. The root is unique in that it does not have any parent.

2.1 Joining

When a new node wishes to join the multicast tree it sends a "join" request to the root. The root then has two options:

1. If the root has less n children, it will send back an "OK" message. It will then add the IP address of the new child to its list of children for future communication.

2. If the root has n or more children, it will send back a “No, try c_1 ” message where c_1 is a randomly chosen child of the root. The new node will then send a “join” request to c_1 to attempt to become one of its children.¹

Each node follows the same algorithm above. The result is (probabilistically) a balanced multicast tree.

2.2 Communication

When a node receives a message from its parent:

1. It makes it available to the local application and
2. It forwards it on to its children (if it has any)

This way whenever the root wishes to send a multicast message, it just sends it to its children and eventually all of the subscribers will receive the message.

When node wishes to send a message it does the following:

1. Sends the message to all of its children (if it has any)
2. Sends the message to its parent (if it has any)

When a node receives a message from its child c_1 :

1. It sends the message to all children other than c_1 (if there are any)
2. It sends the message to its parent (if it has one)

3 What to do

Using the above design, implement a multicast tree application which has the following features:

- A node can decide it will be the root of a multicast tree (assume there is only one on the computer for simplicity)
- A node can decide to join an existing multicast tree by sending a “join” message to the root of the tree (it will have to know the IP of the root somehow).
- Nodes which receive join requests behave as described above
- Any node can send messages to the entire multicast tree
- A node which receives a message prints it out on the screen and forwards it as described above.

You may use Java or C# for this framework according to your preferences. We will use it in future labs for experiments with Lamport clocks and vector time stamps.

¹Another option would be to have the root send back a list of all of its children and then let the requestor choose which child to attempt to contact. The approach above is a bit simpler to implement, if a bit less ideal in terms of balancing the tree.