Assignment 4: Broadcasting Methods

Ang Deng, Akshay Nagendra, Paul Yates

Assignment Summary

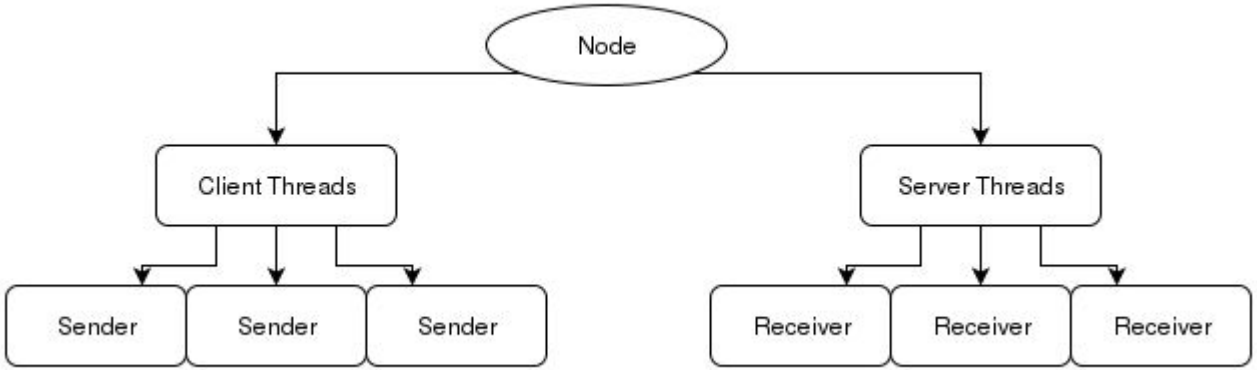
- In this assignment we will let the students implement the broadcasting methods learned in class
  - Reliable Broadcast
  - FIFO Broadcast
  - Causal Broadcast
- Working with practical implementations of these algorithms beyond the simple pseudocode will give students a deeper and more lasting understanding of how they should work

Requirements and Specifications

- Language: Python 2.7
- Skeleton code is provided with instructions on where to insert the missing broadcast algorithm code
  - E.g. the network layer socket setup will already be there
- The student must demonstrate successful completion of the algorithm by submitting correct code and output logs which satisfy the requirements of the broadcast algorithm used
  - For causal broadcast, students must also submit proof that their implementation is correct via an explanation (PDF) and any supporting files

Skeleton Code Features: Server/Client Node Model

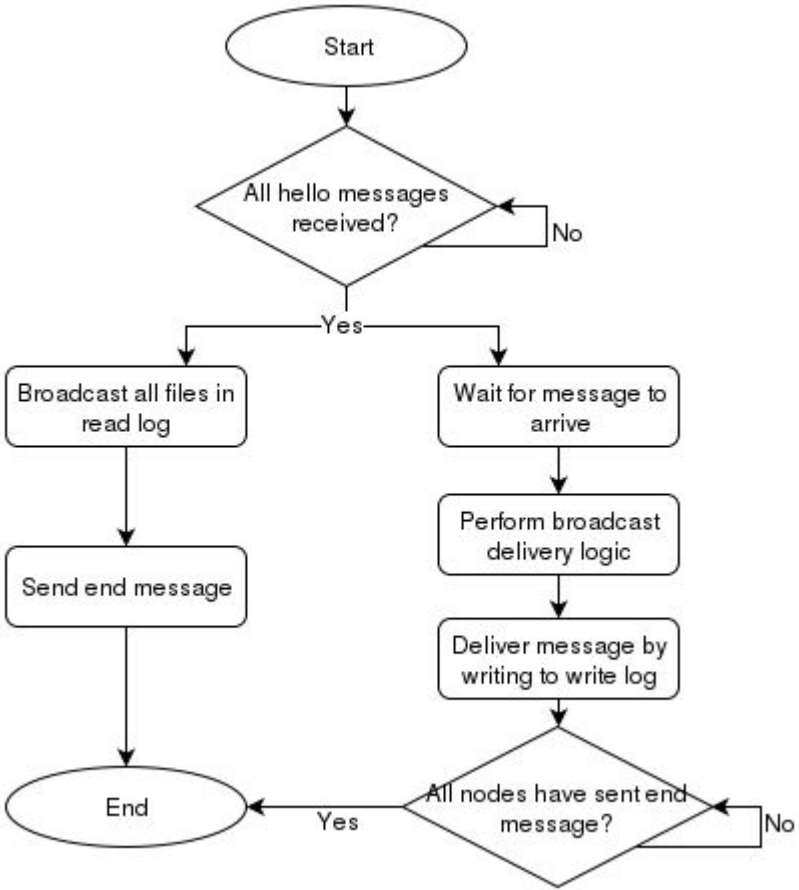
- Each system is modeled as a node which acts as both a server and a client
- Each node sends all of its messages as a client, and receives all messages as a server
- Low-level sending and receiving is provided by the skeleton code
- Python implementation using threads



Skeleton Code Features: Read/Write Logs

- An input file corresponding to each node contains the messages to be sent and their timestamps
  - Messages must be sent only when the node's tick or clock value is >= than the timestamp for a message
- When each message is delivered by a node, it must be written to an write/delivery log unique for that node with the following information
  - Node Clock Value/Tick Value
  - Tag of Message
  - Message Contents

Flowchart for Client/Server Model



Skeleton Code Features: Channel Delay and Failure

- If desired, a channel delay feature may be enabled
  - Each transmission takes a random amount of time (bounded to 3 seconds)
  - This affects order of message received
  - FIFO and Causal rules must still be obeyed
- Broken channels are defined by the input file
  - Each node may have some of its communication channels fail
  - Broadcast algorithms should still be successful as long as there exists at least one send and receive path between all nodes

Solution: Reliable and FIFO Broadcast

- For Reliable Broadcast, students must implement:
  - Unique Tags
  - Broadcast and Receive Handler
  - If a message is received for the first time, it is delivered and re-broadcasted if not sent from receiving node itself
  - Each message should appear once and only once in the output file
- For FIFO Broadcast, students must implement:
  - Unique tags with sequence numbers that increase from broadcast to broadcast
  - Must complete and use r_delivery_handler()
  - Students may notice that they can potentially use same reliable broadcast function for send
  - Each message should appear in sequence number order (per sender node) in the output file

Solution: Causal Broadcast

- **For Implementation:**
  - Whenever a message is broadcasted, all previous delivered messages must be sent alongside and the *prevDelivered* list must be emptied
  - Whenever receiving a message list from *f_deliver*, any message that hasn't been *c_delivered* is delivered to the application layer and appended to the *prevDelivered* list

- **Sample Verification**
  - Whenever a message is broadcasted, record it and the current state of *prevDelivered*
  - If all messages are delivered after all the messages in their corresponding *prevDelivered* list, then the implementation would be correct and maintain causal behavior

Thank you!

- Demo Video (time permitting)
- Any questions?