PROJECT KEY POINTS

SUMMARY & OPERATION DESCRIPTION

a) **PARAMETERS** (N,R,W,T)

- N = replication
- R = read QUORUM
- W = write QUORUM
- T = timeout

b) **ITEM** (fields):

- Key
- Value-String

b) **NODE**:

b.1) FIELDS:

- Key
- List of items (key, value)
- List of node (key)

b.2) METHODS:

- **JOIN**:
  1) CONTACT A NODE
     - get nodes information from command line (java Node join remote_ip remote_port)
     - ping the node to check if it is present or not
     - send a request for getting list of nodes
  2) GET THE RESPONSE
     - store the list of nodes
  3) REQUEST ITEMS & GETTING ITEMS
     - ask to the next node (in the ring) to send the list of items for which the sender is responsible for
     - receive the list and store it
  4) ANNOUNCING MSG & DELETE
     - sender broadcasts a "announcing" msg to all other nodes
     - all the other nodes have to remove items (they are not responsible for anymore)
- **RECOVERY**:
  1) RECOVERY NODE
     - from a node terminal send a request to recover to a specific node (java Node recover remote_ip remote_port)
     * (———-Note: I have some doubts here———)
  2) REQUEST LIST OF NODES
     - node (recovered) sends a request to the node specified from command line in order to get the list of nodes (now in the network)
  3) UPDATE LISTS
     - if node recovers and nothing happened (no join, no leaves, no items added/deleted) lists remain the same
     - if node recovers and someone joined with a smaller key, remove the item/s acquired by the node with smaller key and insert it/them in the list of the node recovered
     - if node recovers and someone joined with a higher key, remove from the list of the recovered node the item for which it is not more responsible.

b.3) LOCAL STORAGE:

- FIELDS
  1) Key
  2) Value
  3) Version

c) **CLIENT**:

c.1) FIELDS:

– Key

c.2) METHODS:

– **READ**:
  1) SEND READ MSG
     - client sends a read msg to one of the node (we call this node CORDINATOR) of the network
  2) CHECK CORDINATOR
     - if CORDINATOR is not present, node sends back a "not present" msg
     - if CORDINATOR is present, node sends the read request to the N clockwise nodes and waits.
  3) SEND RESPONSE
     · if quorum is reached (R less than N), node sends back to the client the item with the highest version number
     · if quorum is not reached after a timeout T, node sends back to the client a msg to inform it.

– **WRITE**:
  1) SEND WRITE MSG
     - client sends a write msg to one of the node (we call it CORDINATOR) of the network
  2) CHECK CORDINATOR
     - if CORDINATOR is not present, node sends back a "not present" msg
     - if CORDINATOR is present, node sends the write request to the N clockwise nodes and waits.
  3) UPDATE/RESPONSE
     · if quorum is reached (max(R,W)), node sends back to the client a success msg and to the other N nodes the update version of the item (version := version + 1)
     · if quorum is not reached after a timeout T, node sends back to the client a msg to inform it.

TECHNICAL PART

- key = (16-bit unsigned integers)
- Value = (16-bit unsigned integers)
- List items/values = HashSet
- Local storage = File

For point A we can think to store them in a file..... Remember to check R+W greater than N

DOUBTS :
How can we implement crash mechanism? — a crash emulation is present in the 2PC lab exercise
When contacting a node -¿ suppose to node a priori the ip/port of contacted node? (SOLVED)