CS/CE/TE 6378: Project III

Instructor: Ravi Prakash

Assigned on: Nov 16, 2012
Due date and time: Dec 7, 2012, 11:59 pm

This is an individual project and you are expected to demonstrate its operation to the instructor and/or the TA. Your program must run on lab machines (net01-net50.utdallas.edu).

1 Requirements

Design a replicated file system which consists of seven independent nodes[1], communicating via UDP channels. There is a logical file in the system, and a physical copy of it is stored on each of the seven nodes. Each node maintains three data structures for its physical copy of the file:

1. VN_i: Version number of the file, initialized to 0.
2. RU_i: Number of replicas updated during the last write operation, initialized to 7 (number of nodes).
3. DS_i: Distinguished node list, initialized to $null$.

All nodes in the replicated file system must implement some mechanism to know the reachability of other nodes. Each node maintains a local data structure, P, for the set of reachable nodes. A node is always an element of its P set. Additionally, there is a manager node which sends commands to the replicas via persistent TCP channels. The command requests could be –

1. READ: Read from the file.
2. WRITE: Write given text to the end of the file.
3. NODE-DOWN: Make all UDP links incident on the specified node down.
4. NODE-UP: Make all UDP links incident on the specified node up (if down).
5. HALT: Make this node exit from execution after releasing all resources.

The manager reads a command file (commands.txt) which contains a sequence of commands to be sent to the replicated file system. Each line of the file is either -
`<REQUEST> <dest-node> [<message>|<other-node>]`
or
`WAIT <delay>`.
In case of "WAIT", the manager waits for the given time before sending next request. If `<delay>=-1`, then the manager waits for a keyboard hit. Please note that requests are sent irrespective of receiving replies from nodes in the replicated file system to the previous request(s). The manager is expected to collect replies as they arrive. One example of the network topology is shown in Figure. 1.

[1]Your program should be easily extensible for any number of nodes.

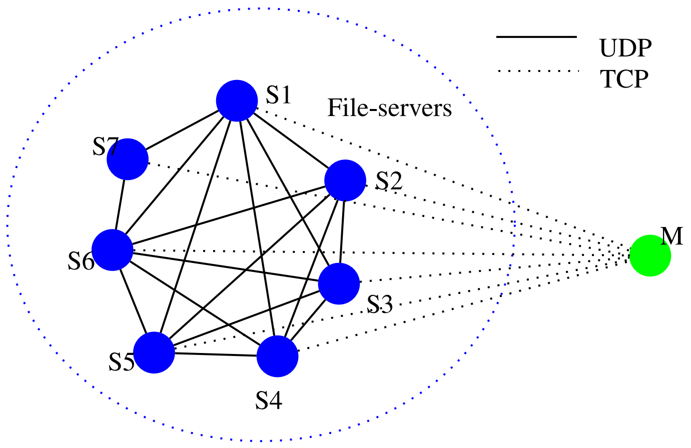

Figure 1: Network Topology

You are required to implement a voting protocol, as described below, to perform READ/WRITE operation to the file.

Reads and Writes

When one of the seven nodes, S_i, receives a READ/WRITE request:

1. S_i sends a VOTE-REQUEST message to all reachable nodes (P_i).
   (a) Upon receiving a VOTE-REQUEST from node S_i, S_j sends $< VN_j, RU_j, DS_j >$ to S.
2. Upon receiving all responses, S_i checks if it belongs to the distinguished partition by calling *Is-Distinguished* procedure.
   (a) If YES: If S_i does not have the latest copy (based on VNs), then obtain the latest copy from a node.
   (b) If NO: ABORT the request.
3. If the request is READ, sends the reply with last line of the file.
4. If the request is WRITE:
   (a) Update $< VN_i, RU_i, DS_i >$ by calling *Do-Update* procedure.
   (b) append the following to the local copy of file: $< VN_i, RU_i, DS_i, text >$.
   (c) Send COMMIT request with $< VN_i, RU_i, DS_i, text >$ to all nodes in P_i besides itself.
5. Upon receiving a COMMIT request from S_j, append the following to the local copy of file: $< VN_j, RU_j, DS_j, text >$ and perform the corresponding updates to the local values of VN, RU and DS.

Fault Tolerance

The replicated file system must be able to handle node failures and recoveries. When a node S_i recovers from failure (NODE-DOWN) then in order to get itself up-to-date, it does the following :

1. It determines if S_i is a member of the distinguished partition by calling *Is-Distinguished* procedure.
2. If YES:If $VN_i < M$, then obtain the latest copy from a node $\in I$.

Is-Distinguished function call

The procedure *Is-Distinguished* for node S_i is defined as follows -

1. Node S_i, upon collecting responses from P_i computes M, N and set I as follows –
   - $M = max\{VN_j : S_j \in P\}$
   - $I = \{S_k : VN_k = M\}$
   - $N = RU_j : S_j \in I$
2. If $|I| > \frac{N}{2}$, return YES.
3. Otherwise, if $|I| = \frac{N}{2}$ and $DS_i \in I$, then return YES.
4. Otherwise, return NO.

Do-update function call

Procedure *Do-Update* for node S_i:

1. $VN_i = M + 1$
2. $RU_i = |P_i|$
3. $DS_i = \begin{cases} P_i & If \quad |P_i| = 3 \\ S' & If \quad |P_i|\ is\ even,\ |P_i| > 3\ and\ S'\ has\ highest\ ID\ in\ P_i \end{cases}$

You can read the paper [1] for more detail understanding of the protocol. However, the instruction given in the project description must be followed.

Your program must display informative log messages on console. You may write a program/script to test the correctness of the program.

2 Submission Information

The submission should be through WebCT in the form of an archive consisting of:

1. File(s) containing the source code.
2. The README file, which describes how to run your program.

DO NOT submit unnecessary files.

References

[1] Jajodia, S.; Mutchler, D., "A hybrid replica control algorithm combining static and dynamic voting," Knowledge and Data Engineering, IEEE Transactions on , vol.1, no.4, pp.459-469, Dec 1989.