­­­­­­Robert Judka

CS550

Programming Assignment 3

Verification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Scenario | Test Steps | Test Data | Expected Results | Pass/Fail |
| read configuration file | 1. run ‘./super\_peer 0 ../config/ata.cfg’ | config file | peer is started on correct port and knows its neighbors | P |
| run peer | 1. run ‘./super\_peer 0 ../config/ata.cfg’  2. run ‘lsof -i :55000’ in Linux shell | config file | peer listening on port 55000 | P |
| run node | 1. run ‘./leaf\_node 0 ../config/ata.cfg nodes/n0/’  2. enter ‘l’ into cli | config file, files in ‘nodes/n0/’ | peer lists all files in ‘nodes/n0/’ | P |
| add new file to node | 1. add new file to ‘nodes/n0/’  2. enter ‘l’ into cli | config file, files in ‘nodes/n0/’,  file ‘foo.txt’ | peer lists all files in ‘nodes/n0/’, including ‘foo.txt’ | P |
| delete file from node | 1. delete file ‘b.txt’ from ‘nodes/n0/’  2. enter ‘l’ into cli | config file, files in ‘nodes/n0/’ | peer lists all files in ‘nodes/n0/’, not including ‘b.txt’ | P |
| existing file search | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli | config file, files in ‘nodes/n0/’ | output listing current node owning ‘a.txt’ | P |
| non-existent file search | 1. enter ‘s’ into cli  2. enter ‘foo’ into cli | config file, files in ‘nodes/n0/’ | output stating file not found | P |
| file download from current node | 1. enter ‘o’ into cli  2. enter node’s current id into cli | config file, files in ‘nodes/n0/’ | output stating no retrieval perform because the node is the current node | P |
| run 2 nodes | 1. run ‘./leaf\_node 1 ../config/ata.cfg nodes/n1/’  2. run ‘./leaf\_node 2 ../config/ata.cfg nodes/n2/’  2. enter ‘l’ into either cli | config file, files in ‘nodes/n1/’,  ‘nodes/n2/’ | peer lists all files in ‘nodes/n1/’ and ‘nodes/n2/’ | P |
| existing file search for file owned by other node in same peer group | 1. enter ‘s’ into cli  2. enter ‘x.txt’ into cli | config file, files in ‘nodes/n1/’,  ‘nodes/n2/’ | output listing other node owning ‘x.txt’ | P |
| existing file search with both nodes sharing that file in same peer group | 1. enter ‘s’ into cli  2. enter ‘j.txt’ into cli | config file, files in ‘nodes/n1/’,  ‘nodes/n2/’ | output listing both nodes owning ‘j.txt’ | P |
| existing file download from other node | 1. enter ‘o’ into cli  2. enter other node’s id into cli  3. enter ‘x.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output showing original name of file downloaded and the name of the new file (both are ‘x.txt’) | P |
| non-existent file download from other node | 1. enter ‘o’ into cli  2. enter other node’s id into cli  3. enter ‘foo’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output stating other node does not have file | P |
| existing file download from other node with both nodes sharing that file | 1. enter ‘o’ into cli  2. enter other node’s id into cli  3. enter ‘j.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output showing original name of file downloaded and the name of the new file (new file with name ‘j-origin-{other node’s id}.txt) | P |
| file search while other node making sequential requests | 1. run script that loops other node making search requests  2. enter ‘s’ into cli  3. enter ‘x.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output listing other node owning ‘x.txt’ | P |
| file download while other node making sequential requests | 1. run script that loops other peer making search requests  2. enter ‘o’ into cli  3. enter other node’s id into cli  4. enter ‘x.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output showing original name of file downloaded and the name of the new file (both are ‘x.txt’) | P |
| 10 nodes all making 200 sequential file search requests across 10 peers | 1. run ‘python node\_simulation.py 10’ | config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | logs showing 200 sequential start/end search requests for each node | P |
| node quitting network | 1. enter ‘q’ into cli  2. enter ‘l’ from other node | config file, files in ‘nodes/n0/’, other node directory | peer showing disconnection and cleanup message, lists only files from other node directory | P |
| killed node process | 1. enter ^C into cli | config file, files in ‘nodes/n0/’, other node directory | peer showing disconnection and cleanup message, lists only files from other node directory | P |
| existing file search for file owned by other node in different peer group | 1. enter ‘s’ into cli  2. enter ‘k.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n1/’ | output listing other node owning ‘k.txt’ | P |
| large file (10 MB) download from other node | 1. enter ‘o’ into cli  2. enter other node’s id into cli  3. enter ‘100.txt’ into cli | config file, files in ‘nodes/n0/’,  ‘nodes/n18/’ | output showing original name of file downloaded and the name of the new file (both are ‘100.txt’) | P |
| *all-to-all* topology message path | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. examine logs from each peer | ‘ata.cfg’ config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | each peer should receive a message from peer 0 at least once and broadcast that message to all other peers | P |
| *linear* topology message path | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. examine logs from each peer | ‘l.cfg’ config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | the message should only be sent to the peers to the left and/or right of the broadcasting peer | P |
| message reaches all peers in *linear* topology | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. examine logs from each peer | ‘l.cfg’ config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | each log should have received the same message at least once (each with a different *TTL* value also) | P |
| *TTL* value decreases between message hops | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. examine logs from each peer | config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | the message should start with a *TTL* value of 2 at peer 0 and then each other peer should broadcast the message with a *TTL* value of 1 | P |
| peer does not re-forward message already seen | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. examine logs from each peer | config file, files in ‘nodes/n0/’,  …,  ‘nodes/n18/’ | once a peer has sent/forwarded a message, the next time it receives that same message it should just send it back | P |
| message ids are properly tracked and maintained | 1. enter ‘s’ into cli  2. enter ‘a.txt’ into cli  3. enter ‘m’ into cli  4. wait 2 minutes  5. enter ‘m’ into cli | config file, files in ‘nodes/n0/’ | the first time ‘m’ is entered into the cli, you should see the message id from the request on step 2. After waiting at least 2 minutes, you should see the old message has been cleaned | P |