Robert Judka
CS550
Programming Assignment 3

Verification

System Testing

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
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’), remote folder has ‘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), remote folder has both files (with their respective names)	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
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
refreshing downloaded file for newest version	1. enter ‘r’ into cli 2. enter the stale file’s origin node 3. enter the filename to refresh	config file, files in ‘nodes/n0/’, ‘nodes/n1/’	output showing the new version of the refreshed file	P
origin node and version is tracked from downloaded files	1. enter ‘o’ into cli 2. enter other node’s id into cli 3. enter ‘a.txt’ into cli 4. enter ‘f’ into cli	config file, files in ‘nodes/n0/’, ‘nodes/n1/’	output showing remote files which are being tracked (with correct file stats)	P
downloading a node’s remote file give you the origin node and version	1. enter ‘o’ into cli 2. enter other node’s id into cli 3. enter ‘a.txt’ into cli 4. enter ‘f’ into cli	config file, files in ‘nodes/n0/’, ‘nodes/n1/’	output shows new downloaded file, however origin node and version are consistent between the two	P

PUSH Consistency Testing

Test Scenario	Test Steps	Test Data	Expected Results	Pass/ Fail
invalidate messages get broadcasted to all nodes when editing a file	1. modify file 'a.txt'	config file, files in 'nodes/n0/'	invalidate message reaches all nodes (through their respective peers)	P
stale version of files get removed from all node's directories	1. modify 'a.txt' 2. enter 'l' into cli	config file, files in 'nodes/n0/'	every node which had the origin node's 'a.txt' downloaded gets it removed and updates its super peer	P
invalidate messages get broadcasted to all nodes when deleting a file	1. delete file 'a.txt'	config file, files in 'nodes/n0/'	invalidate message reaches all nodes (through their respective peers)	P

PULL FROM NODES Consistency Testing

Test Scenario	Test Steps	Test Data	Expected Results	Pass/ Fail
node polls origin node for downloaded files after the *TTR* value expires	1. enter 'o' into cli 2. enter other node's id into cli 3. enter 'a.txt' into cli 4. enter 'f' into cli periodically	config file, files in 'nodes/n0/', 'nodes/n1/'	remote files get their last checked time updated every *TTR*	P
node removes its remote file if it has been marked invalid after a poll	1. modify origin node's 'a.txt' 2. enter 'l' into cli	config file, files in 'nodes/n0/', 'nodes/n1/'	file 'a.txt' gets removed from node's remote folder and updates its peer	P

PULL FROM PEERS Consistency Testing

Test Scenario	Test Steps	Test Data	Expected Results	Pass/ Fail
peer sends invalidate message to other peers after a file has been modified and the *TTR* value expires	1. modify file 'a.txt'	config file, files in 'nodes/n0/'	invalidate message reaches all peers	P
peers check their respective file indexes and send invalidate messages to any nodes which have the file	1. modify file 'a.txt' 2. enter 'l' into cli	config file, files in 'nodes/n0/'	invalidate message reaches any nodes with a stale 'a.txt' and removes them, then updates their respective peer	P
peer tracks files which have been modified and their *TTR* value	1. modify file 'a.txt' 2. enter 'd' into cli	config file, files in 'nodes/n0/'	output shows 'a.txt' as being modified	P