Replica Voting Protocol Software Execution Steps

Prepared by Arun Adiththan

- SSH to the main server in all 11 terminal windows (1 server, 10 voters): `ssh 134.74.126.104`
- SSH to the 11 machines (in Linux lab NAC 7/105) as follows
  - server: `ssh 134.74.160.101`
  - voter 0: `ssh 134.74.160.102`
  - voter 1: `ssh 134.74.160.103`
  - voter 2: `ssh 134.74.160.104`
  - voter 3: `ssh 134.74.160.105`
  - voter 4: `ssh 134.74.160.106`
  - voter 5: `ssh 134.74.160.107`
  - voter 6: `ssh 134.74.160.108`
  - voter 7: `ssh 134.74.160.109`
  - voter 8: `ssh 134.74.160.110`
  - voter 9: `ssh 134.74.160.112`
- Compile the client & server code using g++ command: `g++ file_name.cpp -o executable_file_name`
- Execute the client & server with arguments
  - Server (format): `./executable_file_name number_of_voters redo vote_mode total_rounds powersave_mode fm fa sprd`
    - e.g.: ./server 10 1 2 1 0 3 5 10
  - Client (format): `./executable_file_name number_of_voters voter_id powersave_mode fault_severity`
    - e.g.: ./client 10 0 0 0.6
- Upon execution, `result.txt` gets generated
  - We're typically interested in `TTC`, message overhead (`m`), data message (`dm`)
  - Based on loss percentage, `TTC`, `m`, `dm` will vary
  - `fm` is the number of faulty voters, `bvn` is the actual number of faulty voters (i.e., `fa`).

******