CSE 586

DISTRIBUTED SYSTEMS PROJECT PHASE 3

Authors: Sarveshwar Singhal(50418642) | UBIT - sarveshw

Abhishek Thosar (50394602) | UBIT - athosar

Date of submission: April 8, 2022

**Introduction:**

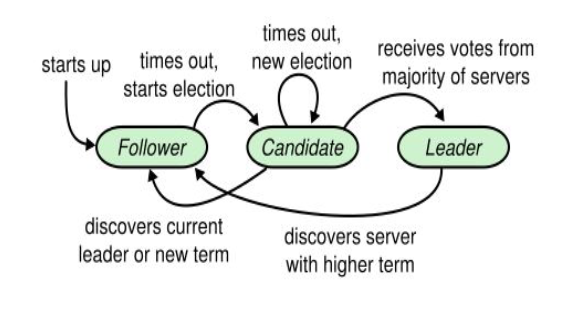
The phase 3 of the project deals with understanding and learning RAFT protocol, socket programming, server-side programming. We learned socket programming, multi-threading, asynchronous call in python to perform the Leader election using the RAFT protocol. Multiple Nodes/servers were created in Docker along with controller to create a distributed system. All the nodes are identical with respect to code.

Initially when the system starts every node is a candidate and when the timeout happens for any one of the node election starts. Post that a leader is elected. Once that leader is elected it periodically sends heartbeats to all other nodes to maintain it’s leadership. In the network if any nodes fail or status of a node changes (through controller) appropriate action is triggered automatically. In case if a leader node is shutdown elections are called or appropriate action is taken based on the controller command.

The main motivation came from looking at the services of big tech giants, their systems never went down, so they must be following distributed systems approach to have data redundancy in case if a server fails, they have other servers to cater the user requests with persistent data.

**Design Overview:**

The below image is the design overview of our Leader election system implementing RAFT algorithm.



**Implementation:**

As part of the project, we created 5 nodes and 1 controller node in the docker container. To perform leader election within the nodes we implemented RAFT algorithm. All the 5 nodes have identical code. All the operations are done on server side. Initially when the system starts (i.e. we start the docker container) all the nodes are candidates as there is no leader. As soon as one of the nodes timed out it request for elections and elections starts.

We have used python to code the server-side logic. Communication within the nodes is done with the help of web sockets. Server-side code used multi-threading to communicate within nodes asynchronously.

**Classes/Methods:**

1. RaftServer: This class returns a RAFT server instance.

a. *get\_election\_timeout*: This method returns the election timeout.

b. *socket\_listener*: Thread to actively listen to socket communication.

c. *become\_candidate*: Method to change node state to candidate and start election process by timeout.

d. *Become\_leader*: After majority vote this method changes server state to leader.

e. *send\_heartbeat*: This method is used to send heartbeat to all other nodes in the distributed network.

f. *send\_append\_entry*: This method is used to send append entry with data to all other nodes in our distributed system.

g. *handle\_vote\_request*: This method is used to handle the voting procedure.

h. *Handle\_vote\_request\_response*: This method parses vote request response *granted or not granted*

i. *Handle\_append\_rpc*: This method handles *heartbeat* signals from leader node

j. *convert\_to\_follower*: This method is used to convert a node to follower state.

k. Check\_majority\_votes: Verifies node majority nodes

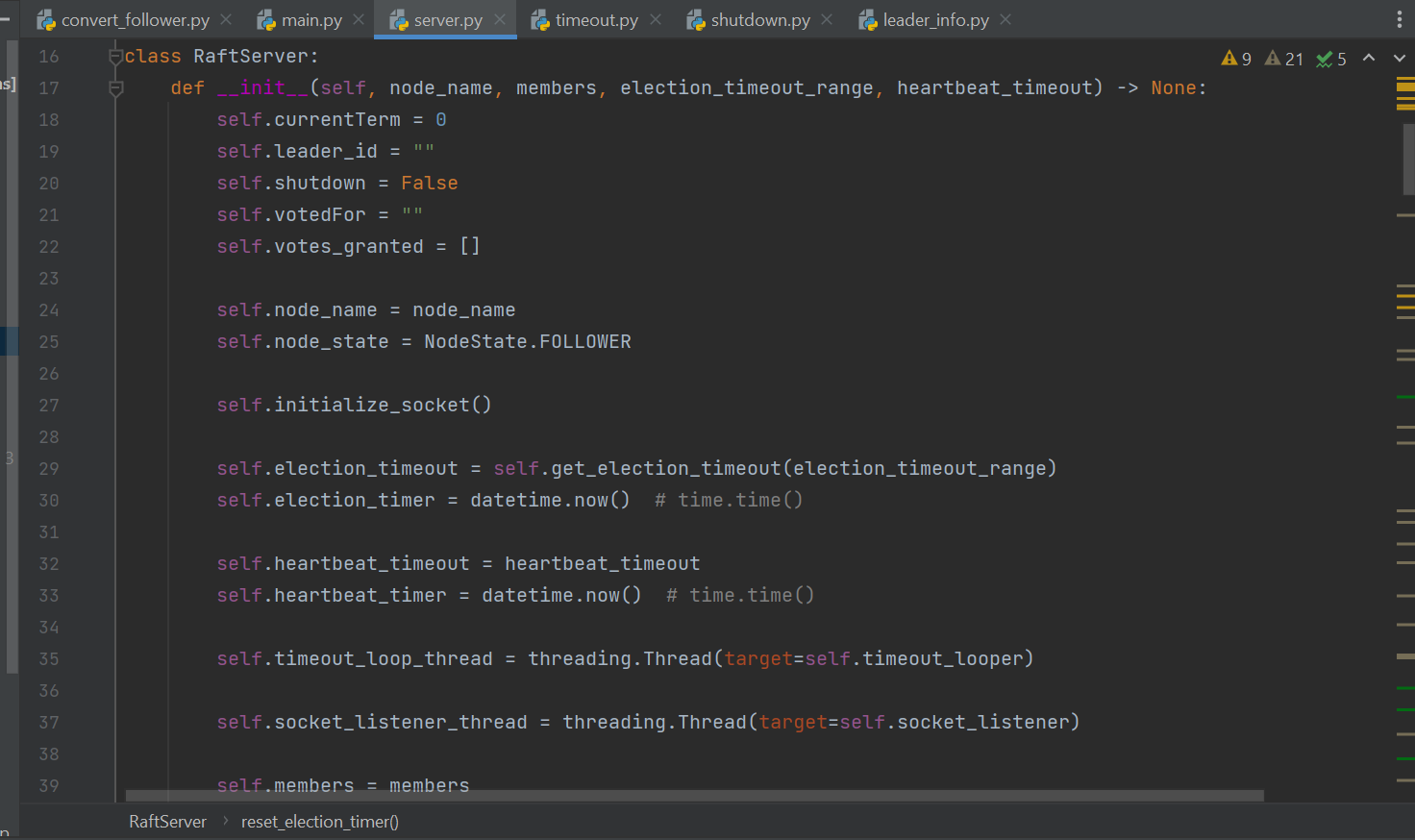
l. Set\_election\_term: Updates node’s current term

m. reset\_election\_timer: Reset election timer after starting election timeout

n. Reset\_heartbeat\_timer: Resets heartbeat timer clock.

2. Controller: This method is used to test the implementation of Leader Election.

**Validations/Screenshots:**



The above image shows the initialization of RAFT server class, where all the variable like heartbeat\_time, node\_state, election\_timeout etc. are set.

Text

Description automatically generated

The above image shows the socket\_listener method which is used to listen all the incoming request on socket.

Text

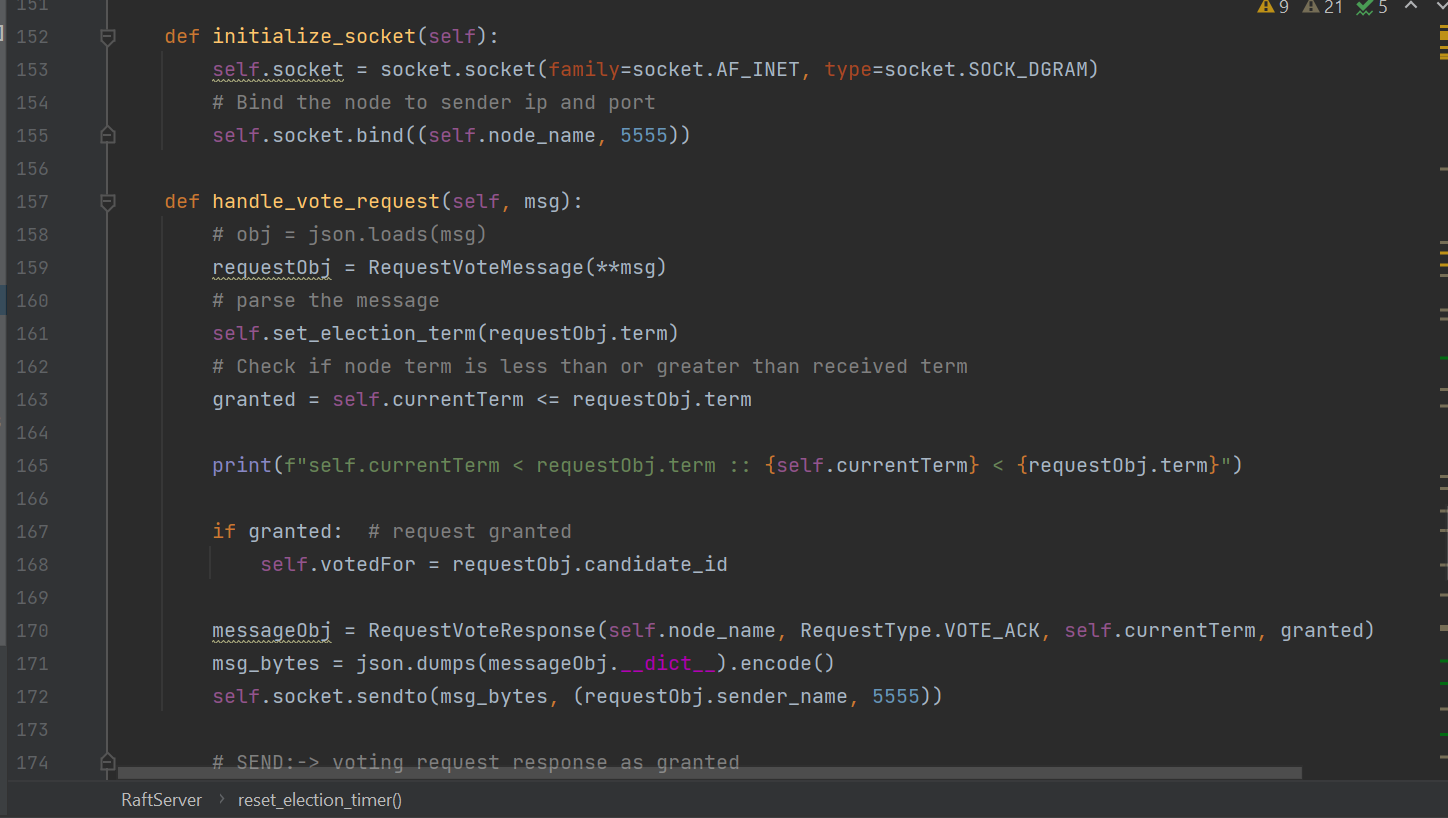
Description automatically generated

The above image shows become\_candidate method which is used to send vote request to other candidates (i.e. nodes)

Text

Description automatically generated

The image shows various methods like become\_leader, send\_heartbeat, send\_append\_entry. These methods are used change a node to state to leader, send heartbeat to other nodes and send data(append) entry to all other nodes.



The above image shows handle\_vote\_request method which is used to check the voting condition for a node. If a node will vote in favor or in against.

Text

Description automatically generated

The above image shows the implementation of controller.

**References:**

<https://raft.github.io/>

<https://raft.github.io/raft.pdf>

<http://thesecretlivesofdata.com/raft/> (Highly recommended)

<https://www.youtube.com/watch?v=YbZ3zDzDnrw>

<https://docs.python.org/3/library/socket.html>

<https://docs.python.org/3/library/threading.html>