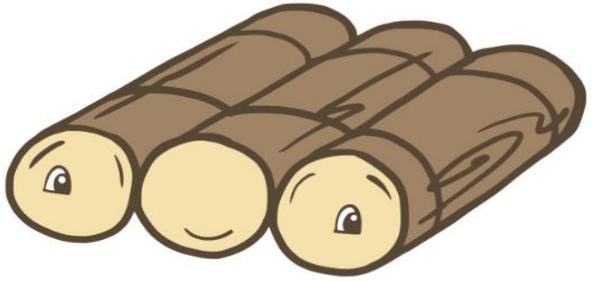
RAFT CONSENSUS ALGORITHM

Miulescu Cristina-Maria

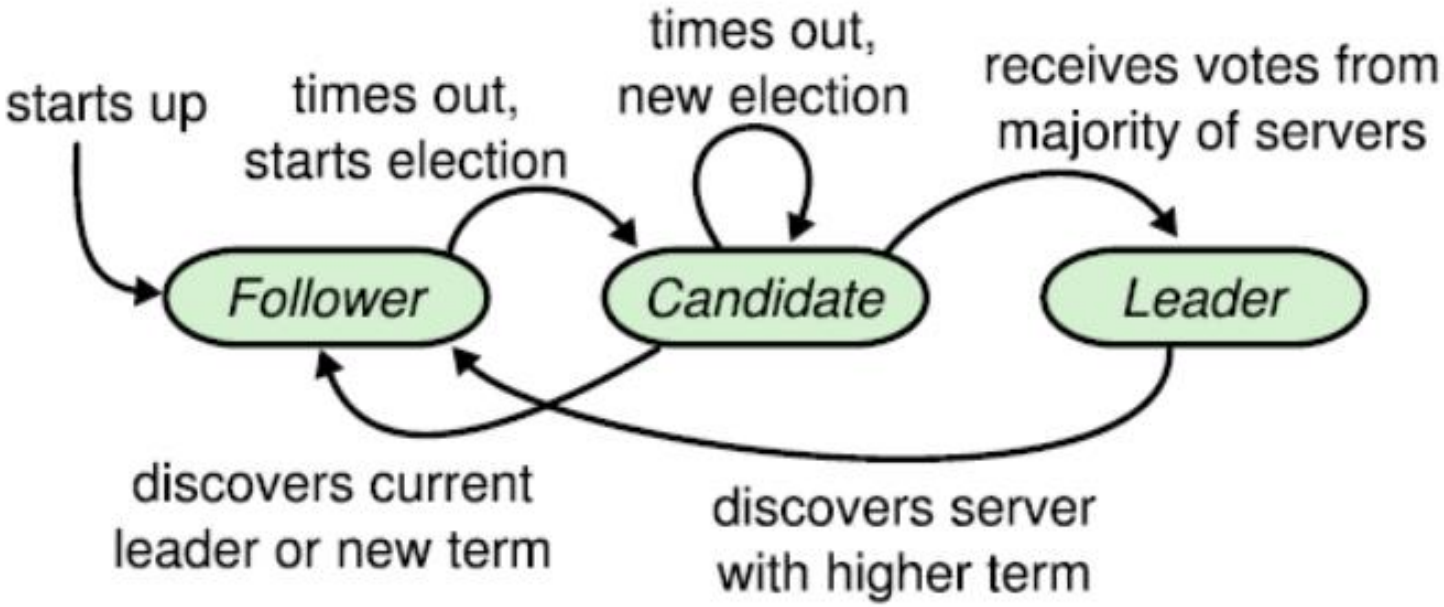
SCPD

WHAT IS RAFT?

- Distributed Consensus Algorithm
- Alternative to Paxos
- Leader-based
- Ensures log replication



MAIN IDEA


starts up
times out,
starts election
times out,
new election
receives votes from
majority of servers
Follower
Candidate
Leader
discovers current
leader or new term
discovers server
with higher term

IMPLEMENTATION

Server-Client

Log Manager

Raft Implementation

SERVER-CLIENT

```
$ python client.py 10000 1
[*] Connecting to localhost port 10000

------------------------------------------------
[*] Type your message:
    >> get ana
[*] Sending: get ana
[*] Received:

>> GET response:
4


------------------------------------------------------------
[*] Type your message:
    >> set d 8
[*] Sending: set d 8
[*] Received:

>> SET response:
d set to 8

------------------------------------------------------------
[*] Type your message:
    >> get d
[*] Sending: get d
[*] Received:

>> GET response:
8
```

LOG MANAGER

Execute

- Execute commands from the client:
  - Get
  - Set
  - Delete
  - Show

Keep

- Keep track of each node's logs

RAFT IMPLEMENTATION

APPEND ENTRIES RPC

Leader:

- Heartbeat messages
- Replicate log entries

- Current term
- Previous term in log
- Previous index in log

REQUEST VOTE RPC

Candidate:

- Request votes

- Current term
- Previous term in log
- Previous index in log

FOLLOWER

RECOVER LOGS

```
$ python server.py s1 10000
C:\Users\crist\Documents\Projects\DA\raft\src\logs\server-configuration.txt
[*] Recovering logs...
[*] Current term: 6
[*] Server started on ('localhost', 10000) with timeout 15.0 seconds
```

RECEIVE APPEND ENTRIES RPC

```
GiveEntriesAfterIndex 1
Sending message
[*] Connection from ('127.0.0.1', 58804)
[*] Received from s1 : AppendEntries 6-9-5-[{'index': '2', 'term': '3', 'c
ommand': 'get b'}, {'index': '3', 'term': '3', 'command': 'set b 5'}, {'in
dex': '4', 'term': '3', 'command': 'get b'}, {'index': '5', 'term': '4', '
command': 'get ana'}, {'index': '6', 'term': '5', 'command': 'get ana'}, {
'index': '7', 'term': '5', 'command': 'set d 8'}, {'index': '8', 'term': '
5', 'command': 'get d'}, {'index': '9', 'term': '5', 'command': 'get d'}]
Last log:  1 2 set b 30
```

CANDIDATE

```
crist@cristina MINGW64 ~/Documents/Projects/DA/raft/src (raft-dev-2)
$ python server.py s1 10000
C:\Users\crist\Documents\Projects\DA\raft\src\logs\server-configuration.txt
[*] Recovering logs...
[*] Current term: 6
[*] Server started on ('localhost', 10000) with timeout 15.0 seconds
True
[*] Starting election...
Sending message
[*] Connection from ('127.0.0.1', 60971)
[*] Received from s2 : Count on me
[*] I am Leader
{'s1': True, 's2': True, 's3': False, 's4': False}
Sending heartbeat...
Sending message
Sending message
[*] Connection from ('127.0.0.1', 60975)
[*] Received from s3 : Count on me
[*] I am Leader
{'s1': True, 's2': True, 's3': True, 's4': False}
```

LEADER

HANDLE CLIENT CONNECTIONS

```
[*] Received from client : get ana
Sending heartbeat...
Sending message
Sending message
Sending message
```

MAINTAIN LOG REPLICATION THROUGH HEARTBEAT

```
[*] Received from s4 : GiveEntriesAfterIndex 1
[{'index': '2', 'term': '3', 'command': 'get b'}, {'index': '3', 'term': '
3', 'command': 'set b 5'}, {'index': '4', 'term': '3', 'command': 'get b'}
, {'index': '5', 'term': '4', 'command': 'get ana'}, {'index': '6', 'term'
: '5', 'command': 'get ana'}, {'index': '7', 'term': '5', 'command': 'set
d 8'}, {'index': '8', 'term': '5', 'command': 'get d'}, {'index': '9', 'te
rm': '5', 'command': 'get d'}]
```

WHEN LEADER FAILS…

```
<class 'str'>
[*] Connection from ('127.0.0.1', 61227)
[*] Received from s1 : AppendEntries 7-11-6-[]
Last log:  9 5 get d

<class 'str'>
True
[*] Starting election...
```

WHEN CLIENT CONNECTS TO A FOLLOWER...

```
crist@cristina MINGW64 ~/Documents/Projects/DA/raft/src (raft-dev-2)
$ python client.py 10001 1
[*] Connecting to localhost port 10001

------------------------------------------------
[*] Type your message:
    >> get ana
[*] Sending: get ana
[*] Received:

Sorry, I am not the leader. Last leader I heard from is: s1

------------------------------------------------
[*] Type your message:
    >>
[*] Closing socket
```

MULTIPLE CLIENTS

```
crist@cristina MINGW64 ~/Documents/Projects/DA/raft/src (raft-dev-2)
$ python client.py 10000 1
[*] Connecting to localhost port 10000

------------------------------------------------
[*] Type your message:
    >> get ana
[*] Sending: get ana
[*] Received:

>> GET response:
4


------------------------------------------------
[*] Type your message:
    >> set d 8
[*] Sending: set d 8
[*] Received:

>> SET response:
d set to 8

------------------------------------------------
[*] Type your message:
    >> get d
[*] Sending: get d
[*] Received:

>> GET response:
8

------------------------------------------------
[*] Type your message:
    >>
```

```
crist@cristina MINGW64 ~/Documents/Projects/DA/raft/src (raft-dev-2)
$ python client.py 10000 2
[*] Connecting to localhost port 10000

------------------------------------------------
[*] Type your message:
    >> get d
[*] Sending: get d
[*] Received:

>> GET response:
8

------------------------------------------------
[*] Type your message:
    >>
```

CONCLUSION

- Main goal of the Raft Consensus algorithm is understandability which has been easily achieved during the implementation

THANK YOU