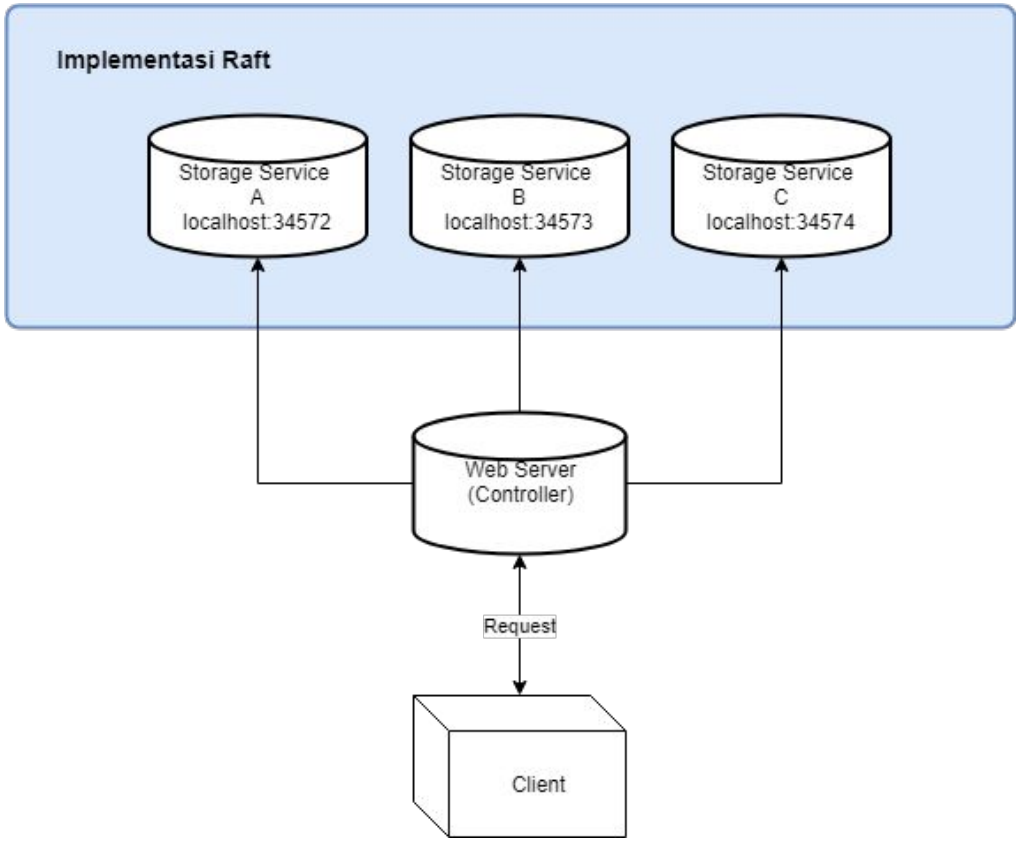
TUGAS 8
SISTEM TERDISTRIBUSI

Kelompok 5

- **Faiq** **05111540000007**
- **Naufal P F** **05111540000057**
- **Dicky Kaisar Utomo** **05111540000077**
- **Subhan Maulana** **05111540000149**
- **Wahyu Pujiono** **05111540000151**
- **Rakhma Rufaida Hanum** **05111540000161**

Pembagian Kerja

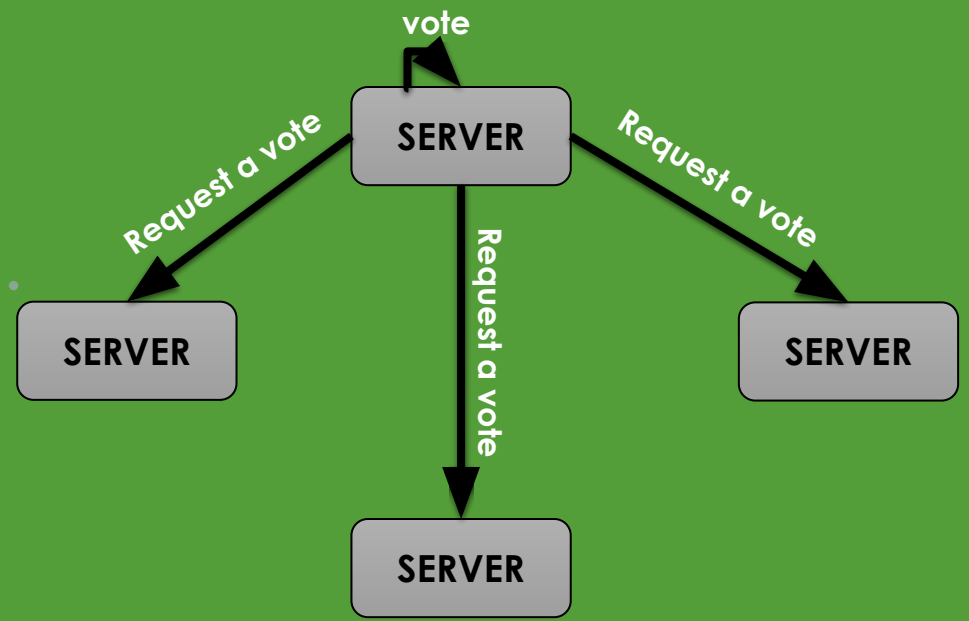
NRP	Nama	Pembagian Kerja
05111540000007	Faiq	Desain Model Flowchart
05111540000057	Naufal Pranasetyo	Implementasi Raft dan Konfigurasi Node
05111540000077	Dicky Kaisar Utomo	Testing Uji Parameter
05111540000149	Subhan Maulana	Desain Arsitektur Sistem
05111540000151	Wahyu Pujiono	Impelentasi Web Server untuk Client
05111540000161	Rakhma Rufaida H	Dokumentasi Hasil Uji Parameter



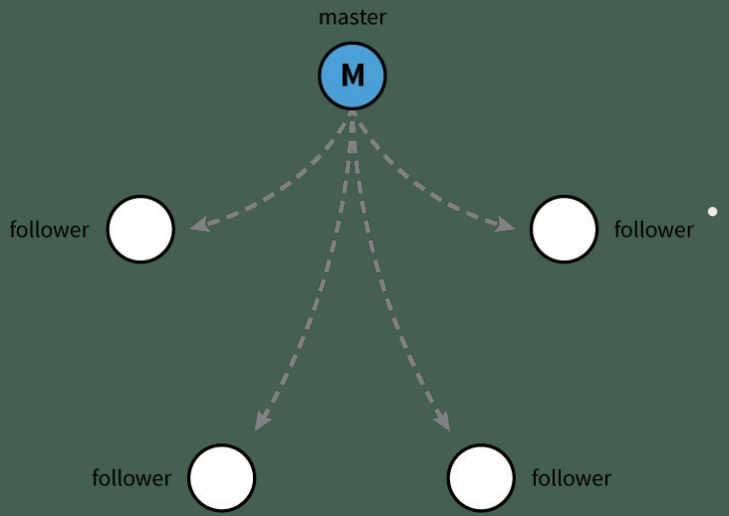
DESAIN ARSITEKTUR IMPLEMENTASI

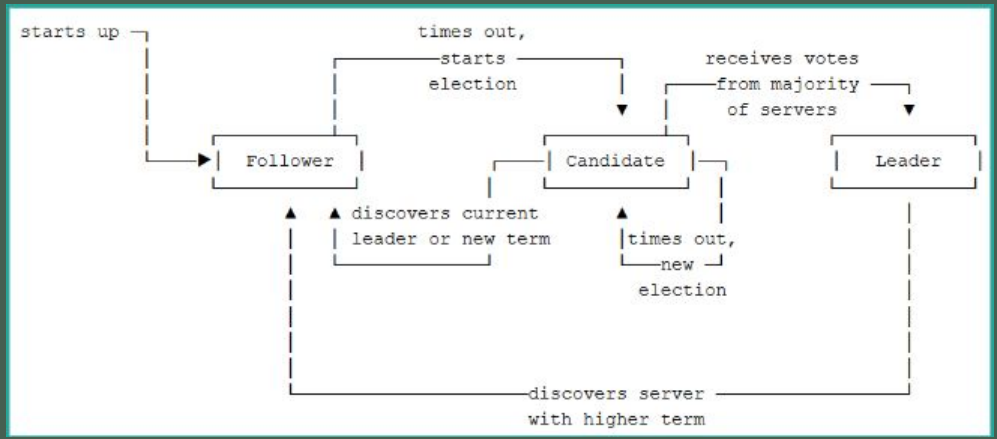
Pembagian State

Pada awalnya semua node atau web server menjadi follower state, masing-masing follower mencari apakah leader sudah ada atau belum. Jika belum, maka semua node menjadi candidate state, masing-masing candidate mengirimkan request dan menerima reply, node yang memiliki vote terbanyak dari semua follower akan menjadi leader.



Leader Election





Konfigurasi Node

- **KONFIGURASI SETIAP NODE**

A localhost:34572
B localhost:34573
C localhost:34574
D localhost:34575
E localhost:34576

```
def get_node_list():
    global node_list
    if node_list is None or len(node_list) == 0:
        node_list = read_nodes('../nodes.txt')
    return node_list

node_list = []
if __name__ == '__main__':
    HOST = environ.get('SERVER_HOST', 'localhost')
    get_node_list()
    try:
        PORT = int(environ.get('SERVER_PORT', '5555'))
    except ValueError:
        PORT = 5555
    app.run(HOST, PORT)
```

Membaca list node storage

Mengaktifkan Web Server

Konfigurasi Leader Election

- **CANDIDATE**

```
def __become_candidate(self):
        self.__votes = 1
        self.__term += 1
        self.state =
NodeStates.CANDIDATE
        logging.info('{0} Became
CANDIDATE'.format(self.id))
        msg_data =
self.__compose_message(MessageT
ypes.VOTE_REQUEST, self.__term)
        self.__send_to_peers(msg_data)
        self.__reset_election_timer()
```

- **VOTE**

```
def __vote(self, candidate_id,
candidate_address,
candidate_term):
        self.__reset_election_timer()
        if self.__term < candidate_term:
            self.__term = candidate_term
            msg_data =
self.__compose_message(MessageT
ypes.VOTE_REPLY)

self.send(self.__peers[candidate_id],
msg_data)
            logging.info('{0} Voted for {1}
in term {2}'.format(self.id,
candidate_id, candidate_term))
```

- **LEADER**

```
def __receive_reply(self, voter_id,
voter_address):
        self.__votes += 1
        logging.info('{0} Current vote
count is {1}'.format(self.id,
self.__votes))
        if self.__votes >=
int(len(self.__peers) / 2 + 1) and
self.state != NodeStates.LEADER:
            self.state =
NodeStates.LEADER
            logging.info('{0} Became
LEADER'.format(self.id))
            self.__heartbeat()
            self.__cancel_election_timer()
```

Konfigurasi Halaman Web Server

- **HALAMAN INDEX**

```
@app.route('/')
def home():
    """Renders the home
page."""
    data = read_data()
    return render_template(
        'index.html',
        title='NoteRaft',
        content=data
    )
```

- **READ DATA**

```
@app.route('/read_data', methods=['GET'])
def read_data():
    nodes = get_node_list()
    value = ''
    has_read = False
    i = 0
    while not has_read:
        try:
            value = get(nodes[i][1])
            has_read = True
        except Exception as e:
            i += 1
            if i >= len(nodes):
                value = e
                break
    data = ''
    if value is not None:
        msg = pickle.loads(value)
        if msg is not None and msg[1] is not None:
            data = pickle.loads(msg[1])
    return data
```

- **WRITE DATA**

```
@app.route('/write_data',
methods=['POST'])
def write_data():
    nodes = get_node_list()
    data = request.form['textareaData']
    has_read = False
    i = 0
    value = {}
    while not has_read:
        try:
            value = set(nodes[i][1], data)
            has_read = True
        except Exception as e:
            i += 1
            if i >= len(nodes):
                value = e
                break
    return str(value)
```

Konfigurasi Message Format (1)

```
def receive(self, received_data, client_address):
        msg = pickle.loads(received_data)
        if msg[0] == MessageTypes.VOTE_REQUEST:
            self.__vote(msg[1], msg[2], msg[3])
        elif msg[0] == MessageTypes.VOTE_REPLY:
            self.__receive_reply(msg[1], msg[2])
        elif msg[0] == MessageTypes.HEARTBEAT:
            self.__respond_to_heartbeat(msg[1], msg[2], msg[3])
        elif msg[0] == MessageTypes.HEARTBEAT_RESPONSE:
            self.__process_heartbeat_response(msg[1], msg[2],
msg[3])
        elif msg[0] == MessageTypes.SET:
            self.set(msg[3])
        elif msg[0] == MessageTypes.GET:
            if msg[3] is not None:
                self.get(msg[3])
            else:
                self.get(client_address)
        elif msg[0] == MessageTypes.COMMIT:
            self.__commit()
```

```
def __send_to_peers(self, data):
        sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
        try:
            for key, value in self.__peers.items():
                self.send_to_sock(sock, value, data)
        finally:
            sock.close()
```

Konfigurasi Message Format (2)

```
def set(self, data):
        logging.info('{0} Trying to set value {1}'.format(self.id, data))
        if self.state == NodeStates.LEADER:
            if self.__data_state == DataStates.INCONSISTENT:
                logging.info('{0} Current state is
inconsistent'.format(self.id))
            else:
                self.__data_to_set = data
                self.__data_state = DataStates.INCONSISTENT
                logging.info('{0} Value was updated'.format(self.id))
        elif self.leader is not None:
            msg_data = self.__compose_message(MessageTypes.SET,
data)
            self.send(self.leader[1], msg_data)
            logging.info('{0} Redirecting to leader'.format(self.id))
        else:
            logging.info('{0} ...but leader is unknown.'.format(self.id))
```

```
 def get(self, client_address):
        logging.info('{0} Trying to read
value...'.format(self.id))
        if self.state == NodeStates.LEADER:
            self.send(client_address,
pickle.dumps((self.__data_state, self.__data)))
            logging.info('{0} Read value was sent to
{1}'.format(self.id, client_address))
        else:
            msg_data =
self.__compose_message(MessageTypes.GET,
client_address)
            self.send(self.leader[1], msg_data)
            logging.info('{0} Redirecting request to
{1}'.format(self.id, self.leader[0]))
```

TERIMA KASIH