Distributed Systems and Cloud Computing: Election Distributed Algorithms

Di Battista Mattia

Università degli Studi di Roma "Tor Vergata"

October 3, 2022

Outline

Outline

Goal

The aim is to implement:

- **Chang and Roberts** algorithm
- **Bully** algorithm
- Several services (i.e., register, heartbeat, verbose, delay)
- Test suite

Goal

The aim is to implement:

- **Chang and Roberts** algorithm
- **Bully** algorithm
- Several services (i.e., register, heartbeat, verbose, delay)
- Test suite

Deployment on **AWS EC2** instance using **Docker** containers

Outline

Outline

Register Service (1)

Register service aims to:

- provide knowledge of the entire network to all nodes belonging to it
- generate a unique random ID for each member of the topology

Register Service (1)

Register service aims to:

- provide knowledge of the entire network to all nodes belonging to it
- generate a unique random ID for each member of the topology

Settings are:

- **TCP** port where the register node is listening
- listening window is kept open for a *SOCKET_TIMEOUT*

Register Service (1)

Register service aims to:

- provide knowledge of the entire network to all nodes belonging to it
- generate a unique random ID for each member of the topology

Settings are:

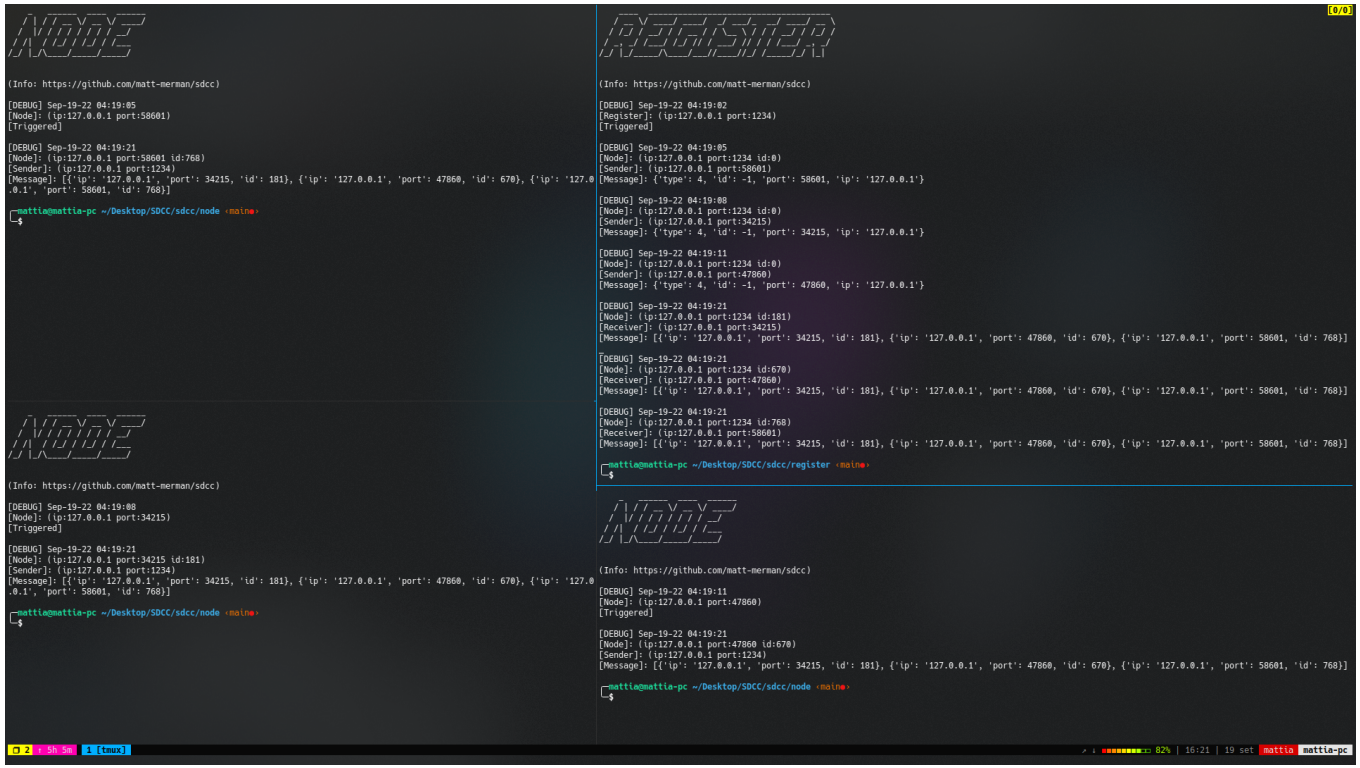
- **TCP** port where the register node is listening
- listening window is kept open for a *SOCKET_TIMEOUT*

Two sockets are created:

1. to communicate with the register node
2. to listen to eventual packets from other nodes

Register Service (2)

E.g., of the **register phase** from three generic nodes and register node:



Outline

Heartbeat Service

Heartbeat service allows detaching crashes by the coordinator node

- a thread heartbeats the leader through a dedicated **TCP** socket

Heartbeat Service

Heartbeat service allows detaching crashes by the coordinator node

- a thread heartbeats the leader through a dedicated **TCP** socket

Settings are:

1. period waited to receive the ACK message
2. period waited to send the HEARTBEAT message

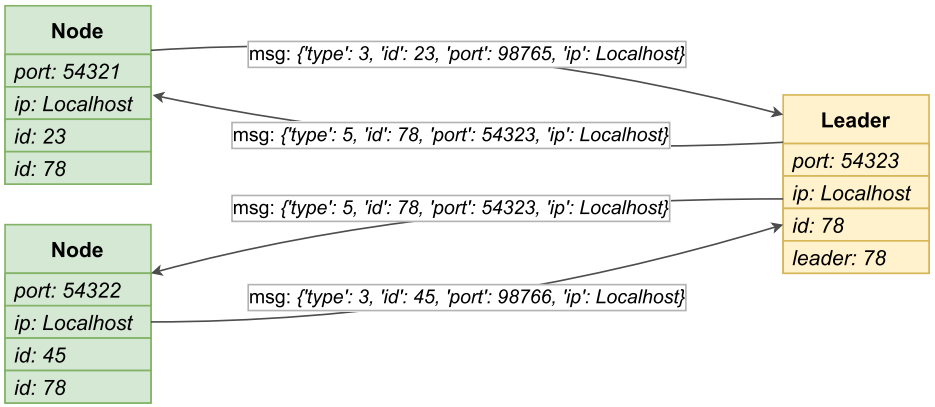


Figure 1: Heartbeat service invoked by two nodes

Outline

Verbose Service

Verbose service shows all messages exchanged (i.e., received, sent) throughout the node lifetime. Main info shown are:

- timestamp
- node info (i.e., IP address, port number, ID)
- receiver/sender info (i.e., IP address, port number, ID)
- message content

Verbose Service

Verbose service shows all messages exchanged (i.e., received, sent) throughout the node lifetime. Main info shown are:

- timestamp
- node info (i.e., IP address, port number, ID)
- receiver/sender info (i.e., IP address, port number, ID)
- message content

Logging library is used to define the message syntax

```
def set_logging() -> logging:
    logging.basicConfig(level=logging.DEBUG,
        format="[%(levelname)s]%(asctime)s\n%(message)s",
        datefmt='%b-%d-%y␣%I:%M:%S'
    )
return logging
```

Outline

Delay Service

Delay service generates a period waited by the sender to forward the current packet

- causing the receiver timeout to expire

```
def delay(flag: bool, ub: int):
    if flag:
        delay = randint(0, floor(ub*1.5))
        time.sleep(delay)
```

- activated by default when tests are performed

Outline

Outline

Implementation (1)

Implementation consists of:

- abstract class
- election distributed algorithms classes
  - as child class to extend the first one

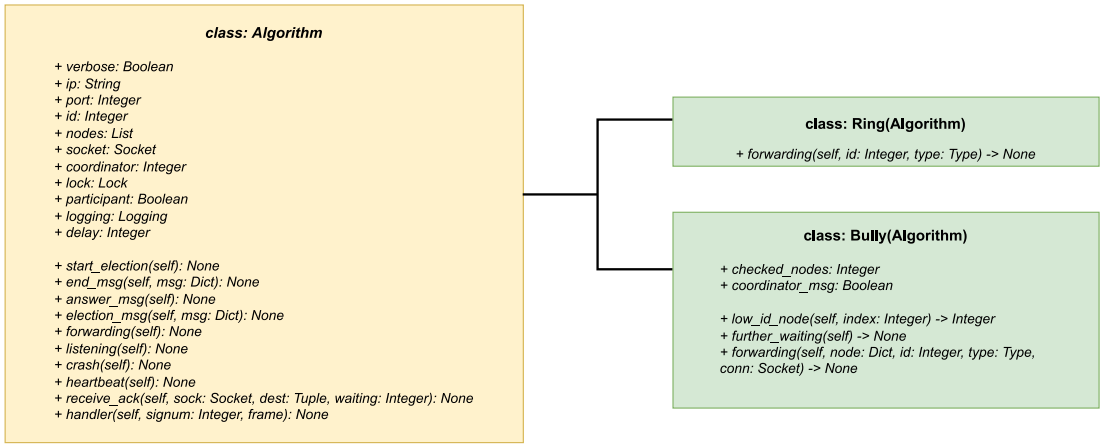


Figure 2: Logic implementation of the classes

Implementation (2)

Six **types of messages** can be exchanged between nodes:

```
class Type(Enum):
    ELECTION = 0
    END = 1
    ANSWER = 2
    HEARTBEAT = 3
    REGISTER = 4
    ACK = 5
```

- ANSWER type is used by the **Bully algorithm**
- REGISTER type is sent during register phase

Implementation (3)

Algorithms begin with run the **listening thread** after which they start an election

- after completing these two phases the heartbeat can start

```
def __init__(self, ...):
    ...
    self.lock = Lock()
    thread = Thread(target=self.listening)
    thread.daemon = True
    thread.start()
    self.start_election()
    Algorithm.heartbeat(self)
```

Implementation (4)

Lock is defined to manage shared resources

- many data are accessed simultaneously from multiple threads

```
self.lock.acquire()
if self.participant or (self.coordinator in
    [self.id, DEFAULT_ID]):
    self.lock.release()
    continue
```

Outline

Chang and Roberts Algorithm (1)

ID's node is used to define the **ring** network

- next node is the one with the greatest ID then current and the lowest among others

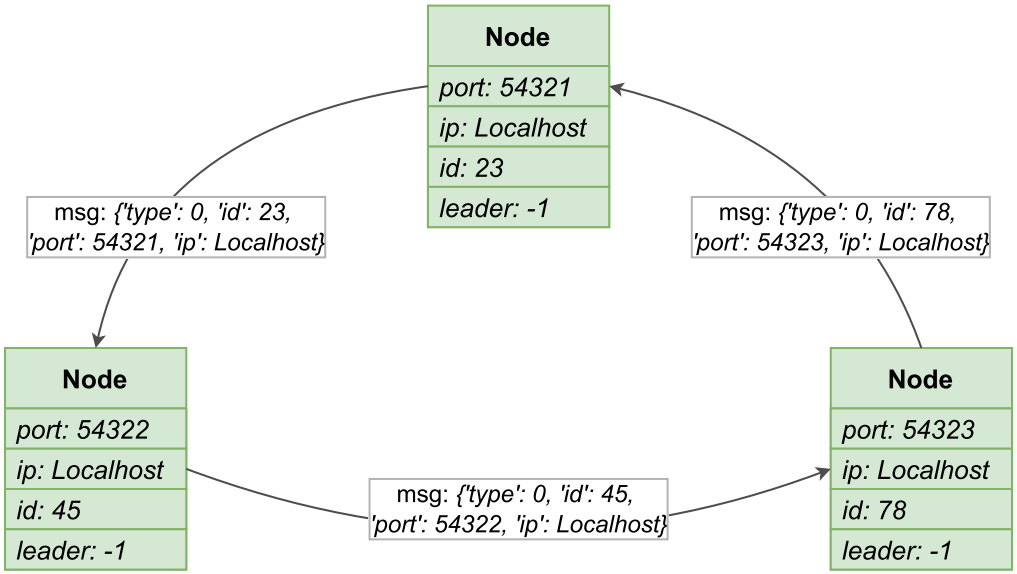


Figure 3: Election started by node with *id=23* in ring topology

Chang and Roberts Algorithm (2)

Leader is removed from the nodes list

- remaining nodes can not interact with it even if it is still active
  - when a leader crash occurs
  - node's timer associated with a heartbeat message elapses

Outline

Bully Algorithm (1)

Bully algorithm assumes that each process knows which processes have higher identifiers

- it can communicate with all such processes

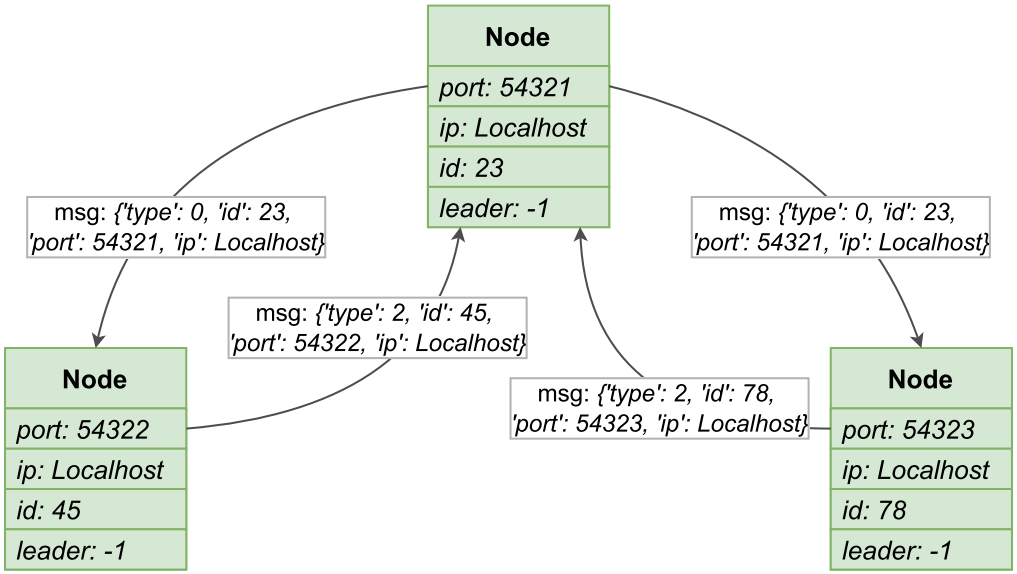


Figure 4: Election started by node with id=23

Bully Algorithm (2)

Possible **scenarios**:

1. leader delays sending the ACK packet
   - other nodes start a new election that will produce the next coordinator also if the previous one is running
2. leader is stopped
   - a new election will start
   - all messages sent to the sleeping node are queued
   - will receive those messages and begin the leader again

Outline

Test Suite (1)

Following **tests** are performed:

1. generic node fails
2. coordinator fails
3. both generic and coordinator nodes fail

Test Suite (1)

Following **tests** are performed:

1. generic node fails
2. coordinator fails
3. both generic and coordinator nodes fail

To interrupt a specific node **psutil** library is used

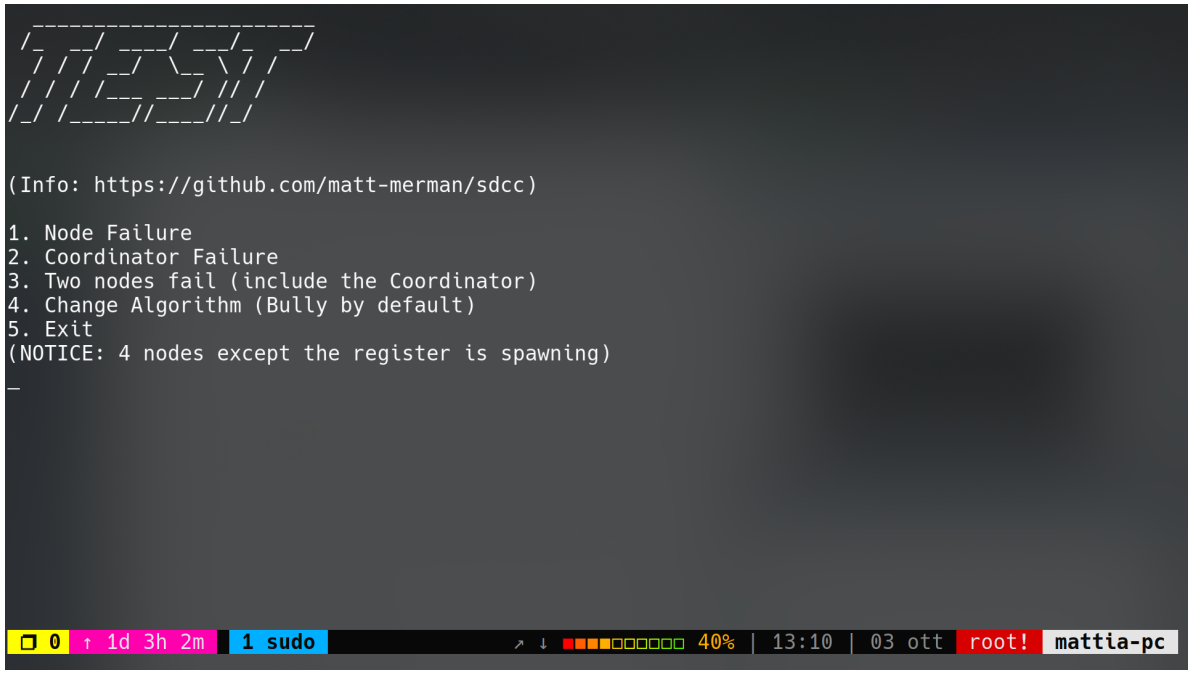
- to kill a process that is listening on a certain **TCP** port

```
def kill_node(self, port: int):
    for proc in process_iter():
        for conns in proc.connections(kind='inet'):
            if conns.laddr.port == port:
                proc.send_signal(signal.SIGINT)
```

Test Suite (2)

Test execution is interactive

- user chooses which test executes
  - sees logging information on the terminal
  - sets which algorithm has to be performed



Outline

Deployment

Network is deployed on an **AWS EC2** instance using **Docker** container

- **Docker Compose** is used to automate the containers creation
  - creates a network where containers can communicate with each other
- to automate the deployment procedure **Ansible** is used
  - Docker installation
  - to forward the code application on an EC2 instance

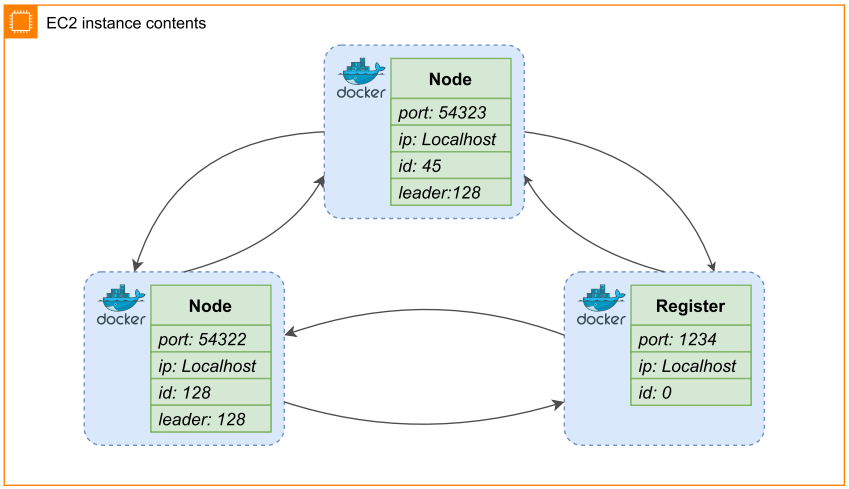


Figure 5: Heartbeat service invoked by two nodes

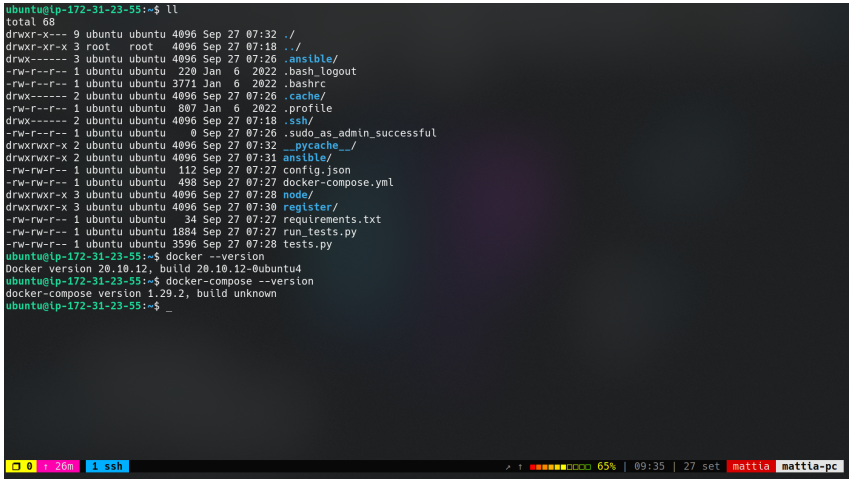


Figure 6: Result from Ansible execution on an EC2 instance