Overlay Project

Distributed System

20th of April 2024

—

Submitted by Groupe K

PHAN Manh Tung

Vadym Vyhovskyi

MOSIG M1

INTRODUCTION

The project aims to create a simulated distributed network where nodes can communicate using RabbitMQ messaging protocol. Each node is assigned a unique identifier and positioned within a virtual ring topology. However, the routing decision within this virtual ring is influenced by the underlying physical network topology. This means that messages may traverse through intermediary nodes before reaching their intended destination in the virtual ring. The primary goal of the project is to develop an efficient routing algorithm that takes into account both the virtual ring structure and the physical network topology, ensuring optimal message delivery between nodes.

Github:

RUN THE PROGRAM

Compile the code:

```
javac -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.7.36.jar Sender.java OverlayNode.java Start.java
```

Run RabbitMQ server:

```
sudo rabbitmq-server
```

Then, open 5 terminals (5 nodes): Start 1 to 5

```
java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.7.36.jar Start 1
```

Initiate a message with this format: Sender node_id R/L “Message”, for example I want to send from node 1 in the right direction message “1R”

```
java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.7.36.jar Sender 1 R "1R"
```

STRUCTURE

• **Topology**: The project features two types of topologies essential for establishing communication within the Overlay network. Firstly, the ***Virtual Ring*** delineates the route a message should take, represented as a circular order of node IDs. However, each node only considers its immediate left and right neighbors for communication purposes. Secondly, the ***Physical Topology*** outlines the actual path that a message traverses from one node to another, defined by an adjacency matrix.

• **Nodes**: Within the network, each node adheres to the Physical Topology, where RabbitMQ queues represent the edges. Every node possesses a unique identifier and maintains copies of both the virtual and physical topologies. Nodes await activation triggered by clients, and upon receiving a message, they relay it to the subsequent node via the associated RabbitMQ queue, following predefined rules.

• **Sender**: The sender serves as the initiator of communication within the network. By triggering a specific node, the sender prompts it to send a message either to its "left" or "right" neighbor. The content of the message is provided by the user through the sender interface.

DIAGRAM

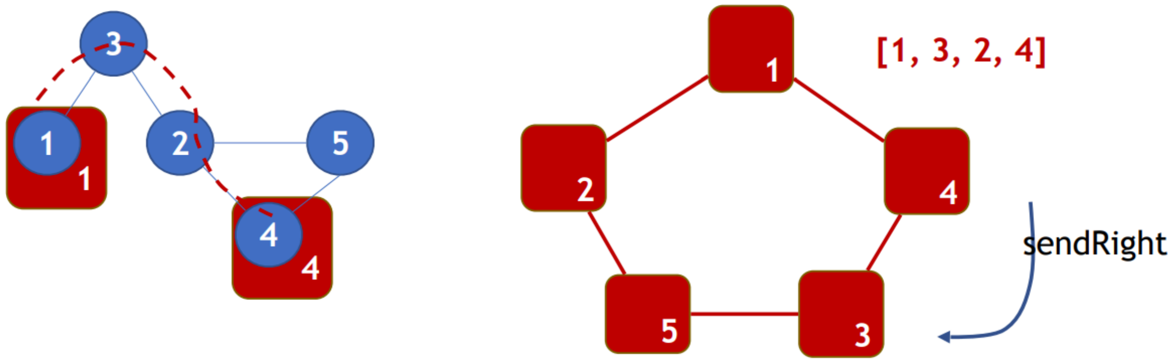
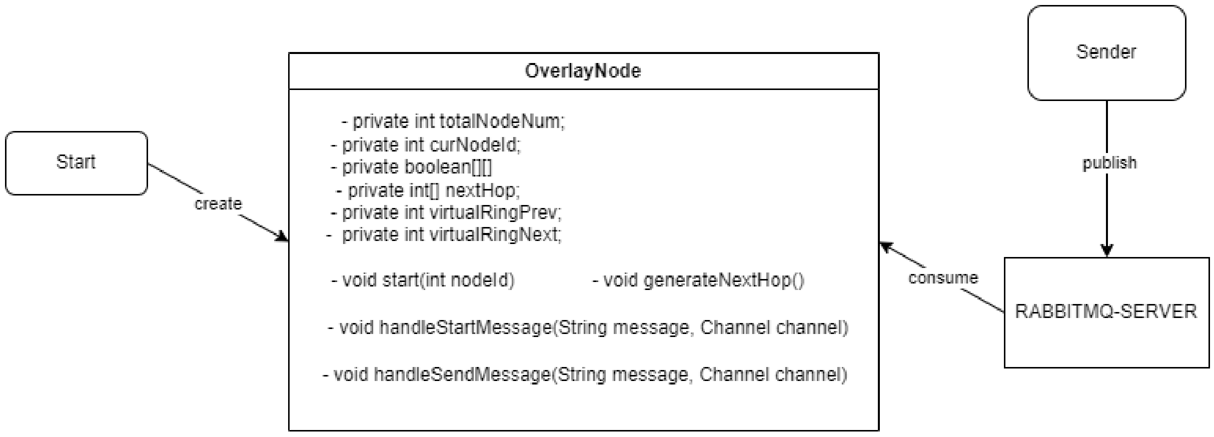


Figure 1: Physical Topology and Virtual Ring



```
OverlayNode

  - private int totalNodeNum;
 - private int curNodeId;
 - private boolean[][]
  - private int[] nextHop;
 - private int virtualRingPrev;
-  private int virtualRingNext;

 - void start(int nodeId)            - void generateNextHop()

 - void handleStartMessage(String message, Channel channel)

- void handleSendMessage(String message, Channel channel)
```

Figure 2: Object Diagram

ALGORITHM (Breadth-First Search)

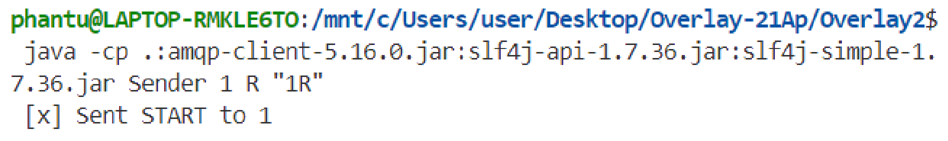
The *generateNextHop* method implements a breadth-first search algorithm to compute the shortest paths between nodes in the Overlay network. Initially, arrays are initialized to store the previous nodes and distances from the current node. Using a queue data structure, the algorithm traverses the network, starting from the current node. At each iteration, it explores neighboring nodes and updates the distance if a shorter path is found. The process continues until all reachable nodes are explored. Subsequently, the algorithm populates the next-hop routing table based on the computed shortest paths. By tracing back from each destination node, it identifies the next-hop node to reach that destination, effectively establishing efficient routing within the network.

MESSAGE HANDLING

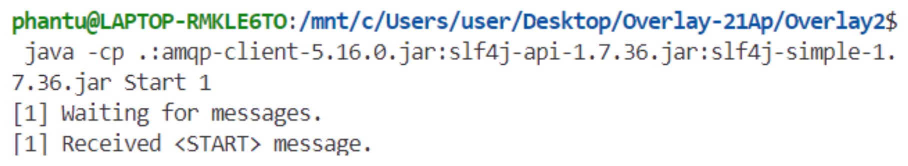
When a message is initiated, it first arrives at the initial node in the network. If the message is a "START" type, the node examines the request to determine whether it should be sent left or right within the virtual ring. Subsequently, the message is forwarded to the next node in the specified direction. This cycle repeats as each subsequent node along the path examines the message type and passes it onward until it reaches its intended destination. Conversely, if the message is of type "SEND," the node verifies if it represents the final destination. If not, the node calculates the next node in the physical ring and dispatches the message accordingly.

EXECUTION SCENARIO

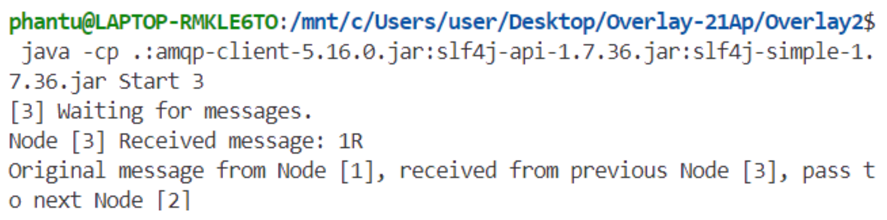
To illustrate our code in action, we send a message “1R” from node 1 to the right, it should traverse through node 3, then node 2 before reaching the destination at node 4.



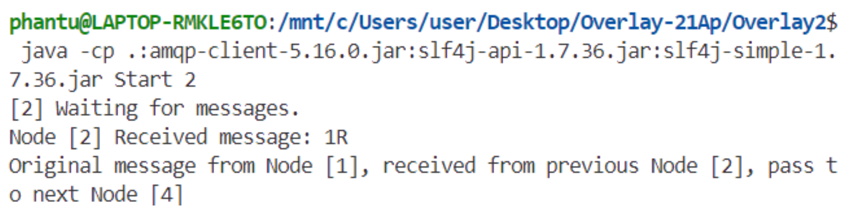
```
phantu@LAPTOP-RMKLE6TO:/mnt/c/Users/user/Desktop/Overlay-21Ap/Overlay2$
 java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.
7.36.jar Sender 1 R "1R"
 [x] Sent START to 1
```



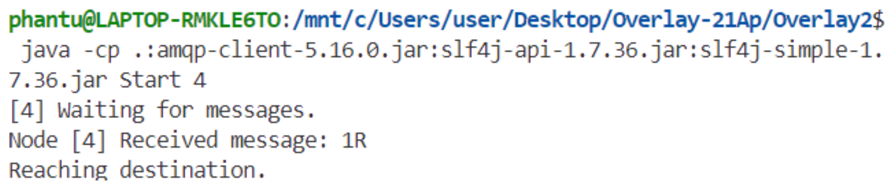
```
phantu@LAPTOP-RMKLE6TO:/mnt/c/Users/user/Desktop/Overlay-21Ap/Overlay2$
 java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.
7.36.jar Start 1
[1] Waiting for messages.
[1] Received <START> message.
```



```
phantu@LAPTOP-RMKLE6TO:/mnt/c/Users/user/Desktop/Overlay-21Ap/Overlay2$
 java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.
7.36.jar Start 3
[3] Waiting for messages.
Node [3] Received message: 1R
Original message from Node [1], received from previous Node [3], pass t
o next Node [2]
```



```
phantu@LAPTOP-RMKLE6TO:/mnt/c/Users/user/Desktop/Overlay-21Ap/Overlay2$
 java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.
7.36.jar Start 2
[2] Waiting for messages.
Node [2] Received message: 1R
Original message from Node [1], received from previous Node [2], pass t
o next Node [4]
```



```
phantu@LAPTOP-RMKLE6TO:/mnt/c/Users/user/Desktop/Overlay-21Ap/Overlay2$
 java -cp .:amqp-client-5.16.0.jar:slf4j-api-1.7.36.jar:slf4j-simple-1.
7.36.jar Start 4
[4] Waiting for messages.
Node [4] Received message: 1R
Reaching destination.
```