Overlay Project

Introduction to Distributed System

Mpoki MWAISELA, Ahmad GHALAWINJI

April 2022

Contents

1 Introduction **2**

2 Main design decisions **2**

2.1 Learning Routes 3

2.2 Start the Ring 4

3 Implementation **5**

3.1 RabbitMQ 6

3.2 Node Class 6

3.3 Message Class 8

3.4 OverlayMonitor Class 9

4 Run and Compile **9**

5 Conclusion **10**

1 Introduction

A network overlay can be thought of as a network that is on top of another network as shown in figure 1 . All nodes in an overlay network are connected to each other by the means of Virtual(Logical) links. Virtual links refers to all those links that are created for the overlay network i.e connecting nodes in overlay. The purpose of this project is to create a ring network as an overlay over another network. This project assumes that each process represents a node in a network, each node knows its left and right neighbours in an overlay prior adjacency establishment between them and there is no network failure once the network has been established.

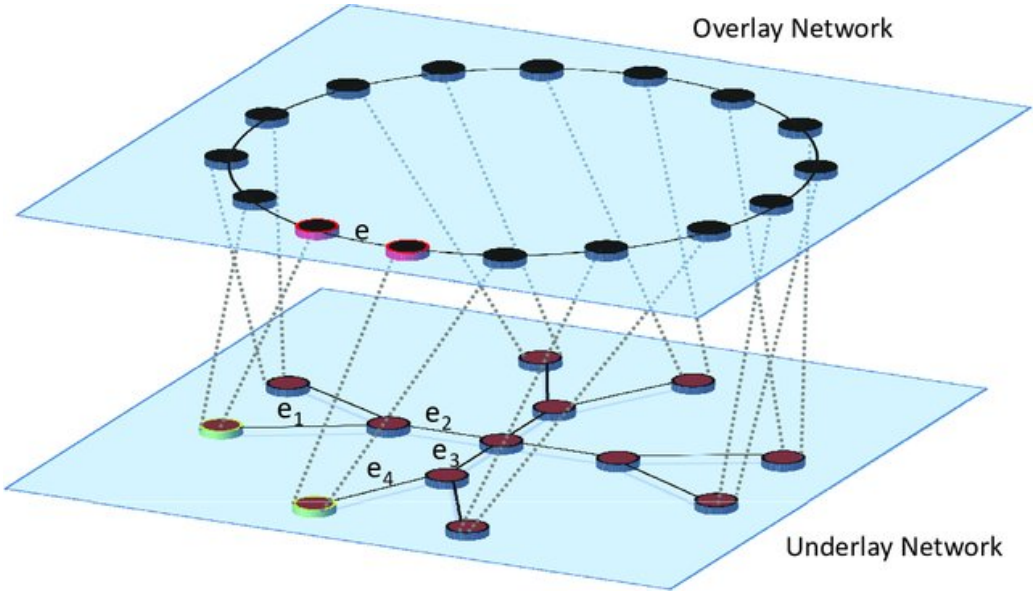

Figure 1: Ring Overlay Network

2 Main design decisions

Essentially our physical network does not know any node that is not directly to it. For example consider figure 2 below, the numbers in red circle represents the logical

addresses. Putting in focus physical node 3 which has a logical address 5 ,this node has knowledge on physical node 4 , 7 , 1 , 2 since they are direct connected to it. Therefore, this node has to learn the routes to other nodes in order to be connected to them. Example a route to physical node 6 with logical address 4 would be 4 , 5 , 6. Note the routes will always represents physical nodes and not logical.

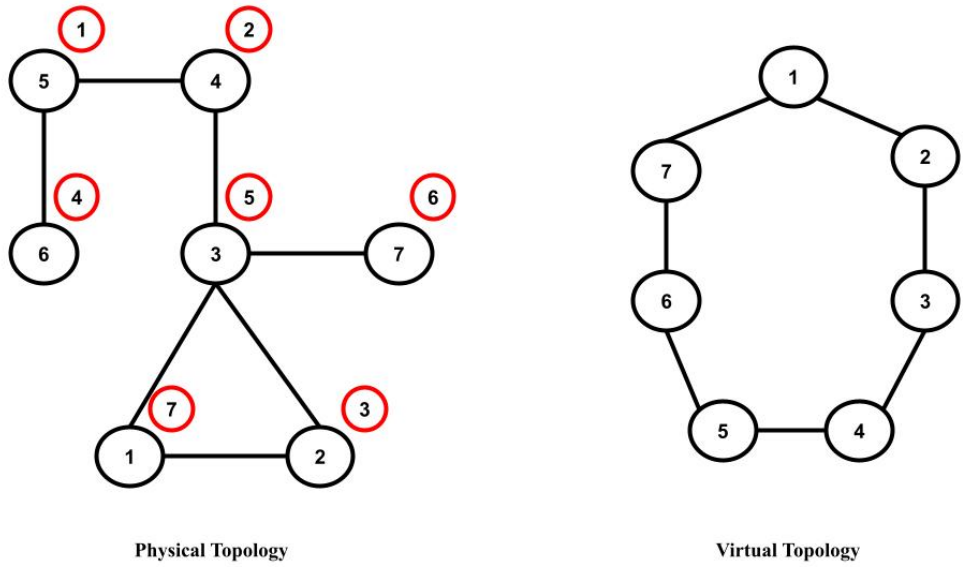

Figure 2: Networks Topology

2.1 Learning Routes

When the nodes are connected to the physical network, they immediately start flooding the network with request to discover it's left and right neighbour in the ring topology(overlay). If the message gets to the intended node i.e the right or left child

they reply the message with the uni-cast message to the node that requested them and this node adds the physical node it received the reply from to its table.

Moreover, if the request gets to the node that is neither its left nor right node , this intermediary node adds the physical and logical node where it received the request message from and the forwards it to all other nodes connected to it except to the node where it received the message from.

Additionally, it can be observed that network loops may rise and this is due to the presence of cycle in a graph as it can be seen physical node 1 3 and 2 in figure 3. To prevent loops each node has to check weather it had previously received the same massage, if yes then we don't process it i.e not forward it.

2.2 Start the Ring

Once a node has identified both its right and left node neighbours , it can start sending ring initialization messages . The node sends this messages after every 30 seconds if it hasn't received it within that period of 30 seconds. The reason to this is due to the fact that some nodes may not have identified their right and left neighbors yet hence can not pass the received message to their neighbours. Therefore, it is considered that a logical node is connected in a ring once it receives the messages it previously sent and once the ring is established messages can be sent to left or right of the ring.

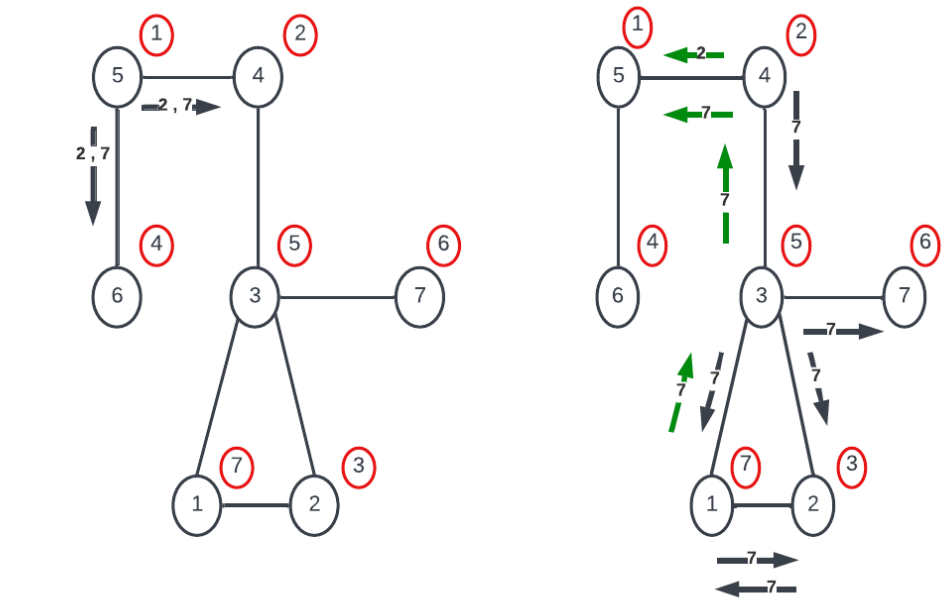

Figure 3: Learning Routes

3 Implementation

The implementation is done using RabbitMQ . The code structure includes three classes which are node, message and overlay Monitor which are further explained below.

3.1 RabbitMQ

In order to demonstrate a distributed environment for each node, this project uses direct exchanges which is an exchange that route messages to queues based on message routing key. The routing key is a message attribute in the message header added by the producer. Producer adds routing key in message header and sends it to direct exchange as shown in the figure below. Therefore, no node receives a message that was not intended for them providing us a complete distributed environment.

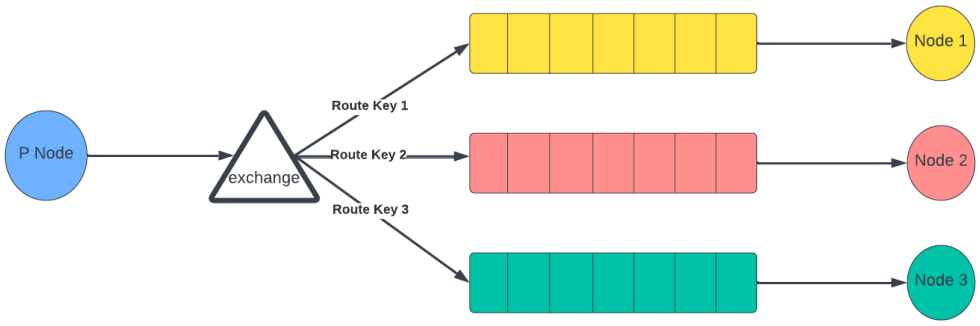

Figure 4: Direct Exchange

3.2 Node Class

The node class is used to implement the core functions for the nodes in a networks. It includes the following functions.

Initialize Method This method initializes necessary variable for node network connectivity including Physical Address, Logical Address, Left child, Right child in a Logical Ring and all Physical Connection it has. Furthermore, this method initializes the RabbitMQ variable such as factory, connection, direct exchange, channel queue and binds the queue to the exchange.

Schedule Method This method schedules two broadcast messages for a node requesting for Physical Address where it's right and left children are located. The broadcasts are sent 10 seconds after the nodes have started and sends it repeatedly after every 30 seconds until it gets them.

Delivercallback Method This method sets the delivercallback for the RabbitMQ and consumes the messages from the queue.

BroadcastMessages Method This method sends a broadcast a message to all Physically connected nodes.

PropagateMessages Method This method forwards a message it received to its direct connected nodes except to the node it received the message. Before forwarding it updates its routing table with the source logical address of the message and the physical address it received the message from and updates the message's source physical address i.e(Store and forward).

UnicastMessages Method This method is responsible for sending uni-cast messages from one node to the other.

RequestReplymessages Method This method is responsible for responding to the request for the node's left and right child and reply messages from the children nodes.

ConnectingMessage Method This Method processes the connecting messages once a node has receive information on its neighbors. The connecting messages are

sent every 30 seconds until a node is connected. A node is considered connected once it receives the connecting message it sent earlier in the ring.

ringMessages This Method processes the messages sent in ring once it has been established. It forwards the message once it reaches to the intermediate nodes to the destined node in the ring sequence until the message reaches the initiator.

3.3 Message Class

This class simulates the message structure exchanged between nodes. The message has seven fields as depicted in figure 5.

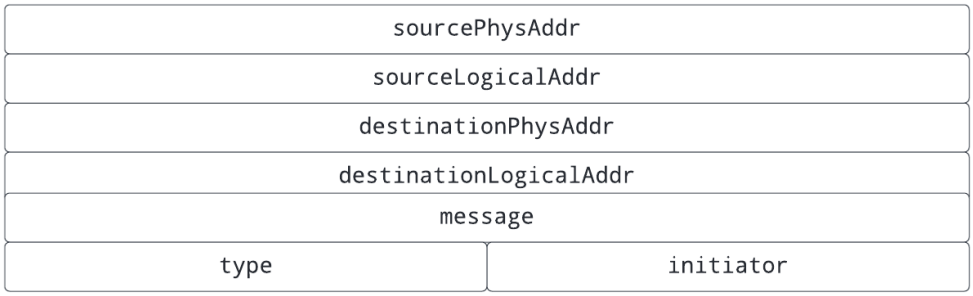

Figure 5: Message Structure

sourcePhysAddr: refers to the source physical address of the message ie source address in physical network

sourceLogicalAddr: refers to the source logical address of the message ie address in logical ring network.

destinationPhysAddrs: refers to the destination physical address of the message ie destination address in physical network.

destinationLogicalAddr: refers to the destination logical address of the message ie destination address in Logical network.

message: refers to the content of the message.

type refers to the type of message that is being exchanged

initiator: This field is only used once the ring has been started and it indicates who initiated the ring message.

3.4 OverlayMonitor Class

This class acts as a helper class to start multiple processes that simulate the physical network. The adjacency matrix is fed to this class and creates the topology for us.

4 Run and Compile

Compile : javac -cp ./bin/amqp-client-5.14.2.jar ./src/node.java ./src/message.java ./src/overlayMonitor.java -d ./bin

Run : "java -cp ./bin overlayMonitor" then fill in the GUI requesting for number of nodes and the adjacency matrix of the physical network.

Note: Running "java -cp ./bin overlayMonitor" starts multiple gnome terminal available in Linux Systems. If run in a different system example windows or mac it will fail hence will require opening all the terminals manually by using:

java -cp ./bin:./bin/amqp-client-5.14.2.jar:./bin/slf4j-api-1.7.36.jar:./bin/slf4j-simple-1.7.36.jar node physical_id logical_id left_neighbor right_neighbor conn1 conn2 ....

conn1 , conn2 ... represents all physical connection a node has.

Remember to be in the same directory as the files to run both java commands.

Read the ReadMe file for more explanation.

5 Conclusion

This projects simulates a distributed environment and aims to producing a virtual network over a physical network, a concept that is prominent at present time. Generally, this is how the internet works, having a big collection of individually managed connected routers, routing protocols definitely play a role in connecting them. In addition ,we also learned that an IP network serves as an overlay network in the internet and we are thankful for that.