CSU33031 Computer Networks Assignment 2 Report

Name: Steven Bondaruk
Student ID: 20333385
Date: 04/12/2022

Table of Contents

Introduction

Send a message across the protocol through the Java socket and packetizing utilities. Unique header names and values are created so that each node can identify the destination the packet needs to go to. Different nodes such as GW1, ISP, DServer1 etc. are on their own specific networks but they are all on the same port which in this protocol it's port 54321.

Theory of the Topic

To simplify it broadly, the system consists of 6 main types of components and 3 types of separate networks. The main components are the laptops, GW forwarders, ISP forwarder, CP forwarder, Server and Controller. The first type of network is the Home Network and it is connected between a specific laptop and its own specific GW forwarder. The second network is the ISP Network and it is connected between the GW forwarders and the ISP forwarder where it receives the packets from the GW forwarders. The third network is the Internet Network and it is connected between the ISP, CP and Server. The controller node is connected between every single network in this protocol. When a transmission happens, the laptop/forwarder/server/controller sends an acknowledgment from where it received the packet from. The laptop sends a message to the server by sending a packet to its GW forwarder. The packet contains the message and the laptop name separated by a comma with the header that identifies the destination it needs to go to. The forwarder initially doesn't know where to send the packet so that it reaches its destination server. So the forwarder temporarily saves the packet and sends a new packet to the controller containing a string of the node name it identifies as, and the destination node it needs to reach to. The controller receives the packet, checks its forwarding table, and then sends back the packet containing the IP address for the next node to the forwarder it received the packet from. The forwarder then receives the IP address, loads the saved packet and sends it to the next node. Other forwarders repeat the process until it reaches the server. Once the server receives the packet, it reads the message with the laptop name and sends back the response packet with the header of the laptop it received the message from. Just to note, once the forwarders receive the IP address to their next node, the forwarders save the address so they won't need to contact the controller every time for future messages.

Headers:

```
public abstract class PacketContent {

    public static final int ACKPACKET= 1;
    public static final int CLAPTOP= 10;
    public static final int PLAPTOP= 11;
    public static final int MLAPTOP= 12;
    public static final int GW1= 20;
    public static final int GW2= 21;
    public static final int GW3= 22;
    public static final int ISP= 30;
    public static final int CP= 40;
    public static final int DSERVER1= 50;
    public static final int DSERVER2= 51;
    public static final int CONTROLLER= 200;
```

Stop & Wait ARQ

In this protocol it uses the Stop and Wait ARQ for error detection and flow control. This means that if a client/laptop will send a message as a packet to the forwarder, before the forwarder will proceed its next actions with the packet, it will send an ACK(acknowledgement) to the node it received the packet from to let it know it got the packet. While the message is being sent, the client/laptop will wait until it receives a response that the server received the message and is ready to take new messages. This ensures that there won't be any lost messages or any information leaks due to the server being overloaded.

Advantages:

- Easy to implement in any system.
- Flow control involves by sending files and messages one frame at a time.
- Any lost frames can be resent.

Disadvantages:

- It is inefficient and a slow system.

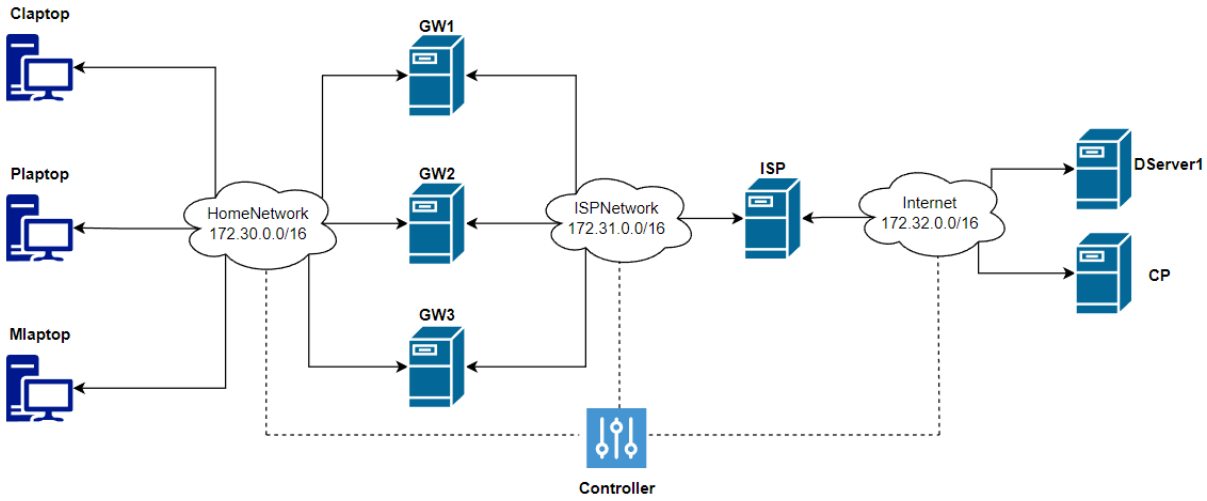
Example how the ACK is sent between the laptop/forwarder/server/controller:

```
DatagramPacket response;
response= new AckPacketContent(info: "OK - From Claptop to GW1").toDatagramPacket();
response.setSocketAddress(packet.getSocketAddress());
socket.send(response);
```

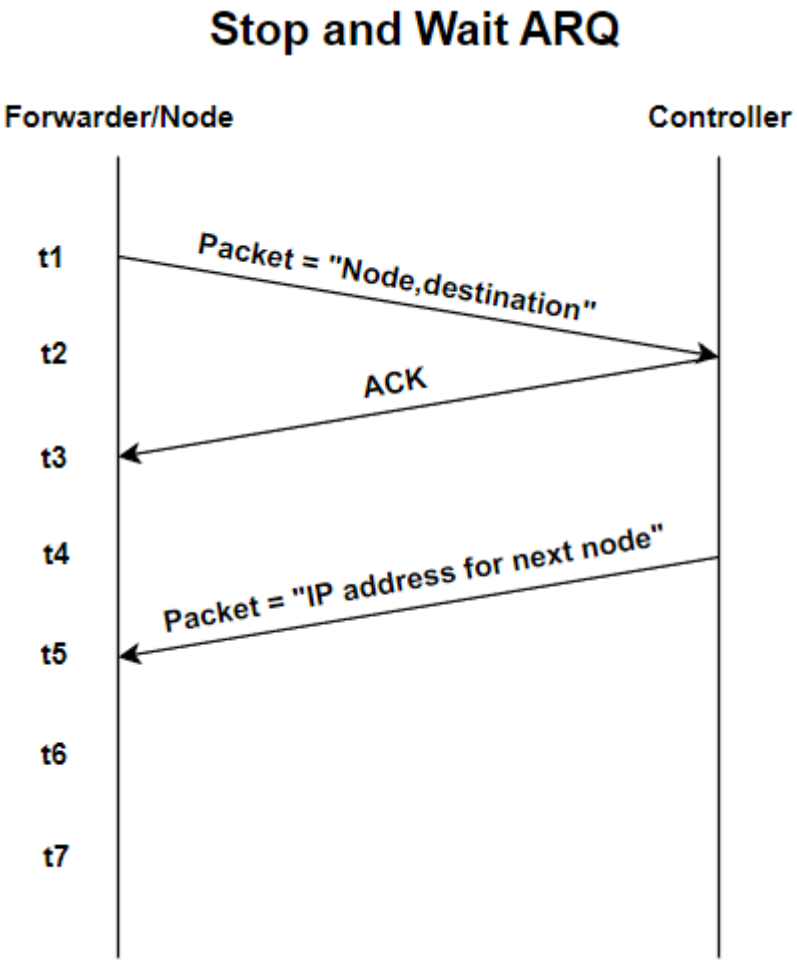
That way it ensures that the nodes only start to forward the message packet after it sends an ACK to where it received the transmission from.

Network structure and Flow Diagram

Topology



Flow Diagram



Components

Laptop

When the laptop launches, it prints a prompt in the terminal where the user can type in the message it wants to send the server. The user needs to write a message and the laptop name it is being sent from separated by a comma. Once the user writes the message, it is packetized with the header as destination.

GW

GW forwarder receives the packet from the laptop and forwards it to the ISP forwarder. If the destination is unknown it sends a request to the controller for the IP address.

ISP

ISP forwarder receives the packet from the GW and forwards it to the CP forwarder. If the destination is unknown it sends a request to the controller for the IP address.

CP

CP forwarder receives the packet from the ISP and forwards it to the DServer1. If the destination is unknown it sends a request to the controller for the IP address. It also receives a message from the server to let the CP know that it's running.

DServer1

When the server launches it sends a message to the CP to let it know that it is running. Once it's running the server waits until it receives a message from the CP forwarder. When the message is received it will send a packet response to the laptop it received from by reading the laptop name within the message.

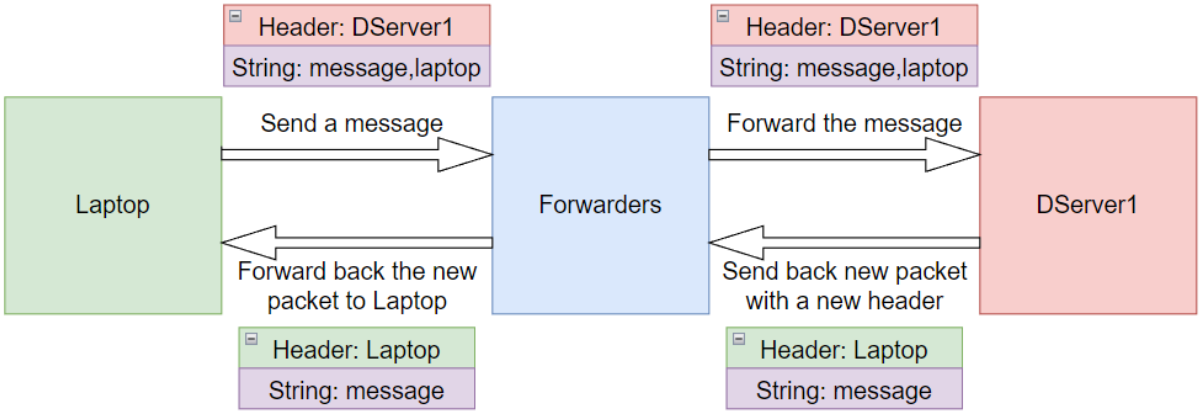
Controller

The controller is connected on all networks when it is running and waiting for requests. Once it receives the request from one of the forwarders, it looks up its forwarding table to figure out which node address the forwarder needs so the packet reaches the destination. Eventually the controller picks the address and sends it back to the forwarder.

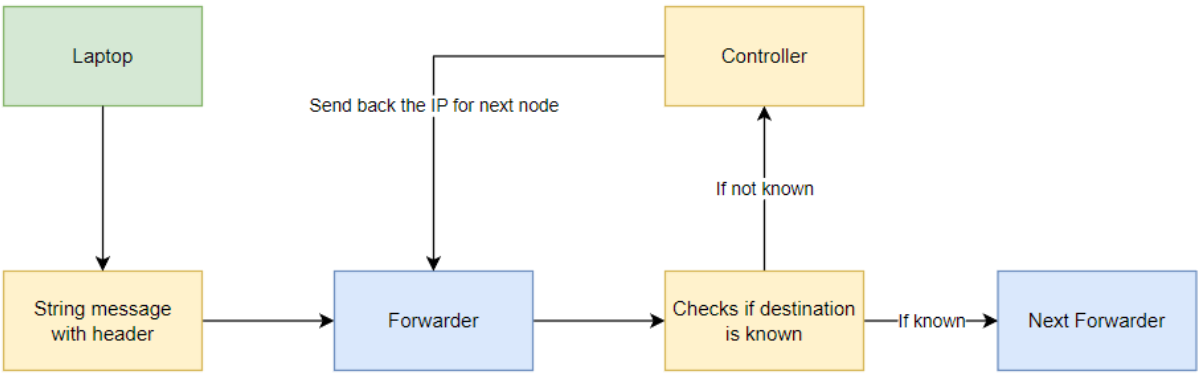
Communication and Packet Description

Outline

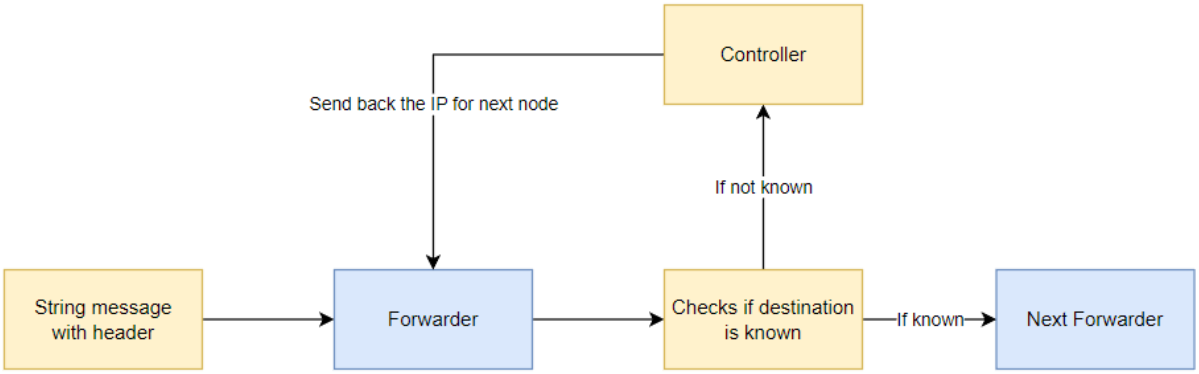
All the nodes within this protocol are connected to their own specific networks, but they are all connected to the port 54321. The laptop and the server communicate between each other through the forwarders using the headers to identify the destination. Every time the forwarder has an unknown destination to the next node, it will send a new packet to the controller with the header value for a controller as the destination to request IP address for its next node to pass on the packet. Once the forwarder receives the IP address, it will remember it and won't need to ask the controller for the address again.



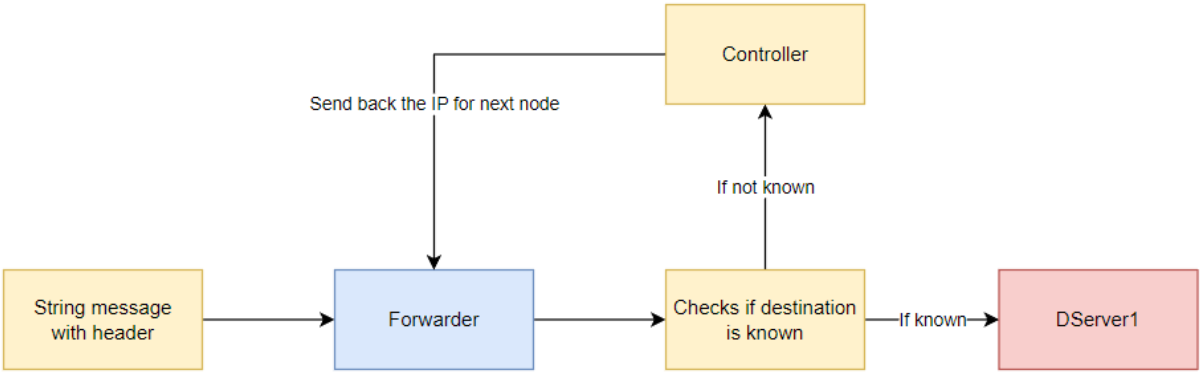
Laptop to Forwarder



Forwarder to Forwarder



Forwarder to DServer1



Implementation

Claptop

```
public class Claptop extends Node {
    static final int PORT = 54321;
    static final String GW1_IP = "172.30.0.3";
    InetSocketAddress dstAddress;

    /**
     * Constructor
     */
    Claptop(int srcPort) {
        try {
            socket= new DatagramSocket(srcPort);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

GW1

```
public class GW1 extends Node {
    static final int PORT = 54321;
    static String CONTROLLER_IP = "172.31.0.4";
    static String CLAPTOP_IP = "unknown";
    static String ISP_IP = "unknown";
    InetSocketAddress dstAddress;
    PacketContent tempContent;
    String tempName;

    boolean server1DestinationKnown = false;
    boolean claptopDestinationKnown = false;

    GW1(int srcPort) {
        try {
            socket= new DatagramSocket(srcPort);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

ISP

```
public class ISP extends Node {
    static final int PORT = 54321;
    static String CONTROLLER_IP = "172.31.0.4";
    static String GW1_IP = "unknown";
    static String GW2_IP = "unknown";
    static String GW3_IP = "unknown";
    static String CP_IP = "unknwon";
    InetSocketAddress dstAddress;
    PacketContent tempContent;
    String tempName;

    boolean server1DestinationKnown = false;
    boolean claptopDestinationKnown = false;
    boolean plaptopDestinationKnown = false;
    boolean mlaptopDestinationKnown = false;

    ISP(int srcPort) {
        try {
            socket= new DatagramSocket(srcPort);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

CP

```
public class CP extends Node {
    static final int PORT = 54321;
    static String CONTROLLER_IP = "172.32.0.5";
    static String ISP_IP = "unknown";
    static String DSERVER1_IP = "172.32.0.4";
    InetSocketAddress dstAddress;
    PacketContent tempContent;
    String tempName;

    boolean server1DestinationKnown = false;
    boolean claptopDestinationKnown = false;
    boolean plaptopDestinationKnown = false;
    boolean mlaptopDestinationKnown = false;

    CP(int srcPort) {
        try {
            socket= new DatagramSocket(srcPort);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

DServer1

```
public class DServer1 extends Node {
    static final int PORT = 54321;
    static final String CP_IP = "172.32.0.3";
    static final String ISP_IP = "172.31.0.3";
    InetSocketAddress dstAddress;

    DServer1(int port) {
        try {
            socket= new DatagramSocket(port);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

Controller

```
public class Controller extends Node {
    static final int PORT = 54321;
    static final String CLAPTOP_IP = "172.30.0.2";
    static final String CLAPTOP_IP2 = "172.30.0.2";
    static final String PLAPTOP_IP = "172.30.0.4";
    static final String PLAPTOP_IP2 = "172.30.0.4";
    static final String MLAPTOP_IP = "172.30.0.6";
    static final String MLAPTOP_IP2 = "172.30.0.6";
    static final String GW1_IP = "172.31.0.2";
    static final String GW1_IP2 = "172.31.0.2";
    static final String GW2_IP = "172.31.0.5";
    static final String GW2_IP2 = "172.31.0.5";
    static final String GW3_IP = "172.31.0.6";
    static final String GW3_IP2 = "172.31.0.6";
    static final String ISP_IP = "172.31.0.3";
    static final String ISP_IP2 = "172.32.0.2";
    static final String CP_IP = "172.32.0.3";
    static final String DSERVER1_IP = "172.32.0.4";
    InetSocketAddress dstAddress;
    String ipName;
    String nodeName;
    String nodeString;
    String dest;

    ArrayList<String []> table = new ArrayList<String[]>();

    /**
     * Constructor
     * Attempts to create socket at given port and create an InetSocketAddress for the destinations
     */
    Controller(int srcPort) {
        try {
            socket= new DatagramSocket(srcPort);
            listener.go();
        }
        catch(java.lang.Exception e) {e.printStackTrace();}
    }
```

User Interaction

Once the laptop, forwarders, server and controller are running, the laptop prints the message to the terminal and asks the user the message it wants to send to the server along with the laptop name separated by comma such as “message1,claptop”. Once the message is sent the forwarders pass on the packet and communicate between other forwarders and controller. Once the server receives the message it sends back the response packet to the laptop it received the message from.

```
root@e99e429474c2:/compnets/assignment2# java -cp . Claptop
File name request: message1,claptop
Sending packet to GW1
File name: message1 Response
Filename: message1 Response
File name request: message11,claptop
Sending packet to GW1
File name: message11 Response
Filename: message11 Response
```

Packet Encoding

In this protocol, the packet consists of the header and the data that is transferred. The data consists of the message as a string and a header to identify the destination for other nodes. The data gets concatenated with the message and header, it then packetizes and is sent to the destination server. For instance a laptop will send a data of string message and laptop name concatenated with the header to identify the destination of the server. It converts into a packet and is sent to the GW forwarder. Then the forwarder then asks the controller for the address of the next node by sending a new packet containing the forwarder name and the required destination. Once the controller receives the request it looks at the forwarding table for the appropriate node to reach the destination. The controller eventually sends a packet containing the IP address as a string and a header destination to the forwarder. Once the forwarder receives the packet it takes the IP address and passes on to the next forwarder until it reaches the server. Once the server receives the packet it creates a new packet as a response and sends back to the laptop it received the packet from.

Summary

In conclusion, the system consists of 3 laptops, 1 Server, 3 GW forwarders, ISP forwarder, CP forwarder and a Controller, split into 3 networks. The communication between the laptop and the server is done through the forwarders using packets which consist of unique headers which identify the destination the packet needs to go to. Forwarders contact the controller to obtain the address for the next node/forwarder for the packet to reach the destination. Once the server receives the message it sends back the response to the laptop it received the message from.

Reflection

The drawback of this protocol is that once the nodes have to be set up in a certain way for them to connect and work since the controller is not dynamic and scalable. If I were to make this different I would program the controller where it would scan all the nodes on a certain network and automatically update the forwarding table appropriately where each of the nodes are able to communicate each other even if there is a new laptop or server is added.