

Packet Route Tracer

Jose Gonzalez
Romeo Bellon
Vinicius Romani

Assignment Topic

Using your choice of programming language (this will probably require 2 languages) develop a graphical user interface (GUI) that allows the user to put in an IP address (or domain name) and then the application visually shows the route/path a data packet may take (i.e. using traceroute like application) using the internet control message protocol (ICMP).

The Choice

Challenging

Takeaways

Slow Start

Language Choices

- ❖ C with libpcap
- ❖ jpcap
- ❖ jNetPcap

GUI Implementation

- ❖ wxPython
- ❖ Jython Music
- ❖ Python GUI

(Really) Getting Started

Languages

- ❖ Java
- ❖ Python

GUI Implementation

- ❖ JavaFX

What is Packet Route Tracer?

A network tool that allows users to visualize a packet's possible route given a specific domain name.

Basic Flow

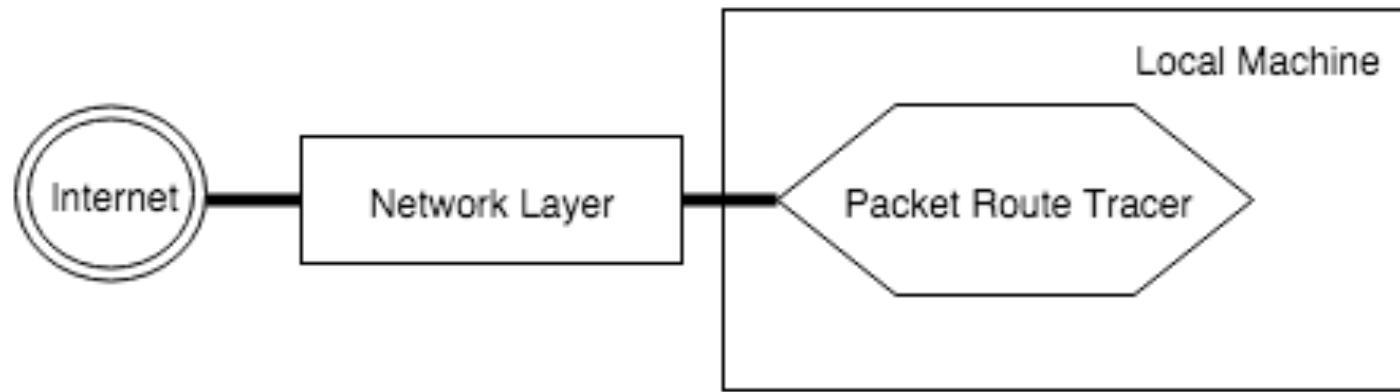


Figure 1: Packet Route Tracer uses the network layer to achieve it's goal.

Member Responsibilities

Romeo:

- ▶ 3.2.1 The GUI will be interactive, allowing for the user to input a desired domain name.
- ▶ 3.2.2 The GUI will display all available controls to the user.
- ▶ 3.2.3 The GUI will display a list of the clients traversed by the packet tracing algorithm.
- ▶ 3.2.4 The GUI will display the routes intuitively in a manner that can be understood easily by the user.
- ▶ 3.2.5 The GUI will display all of the network devices encountered and the relevant information for each device.
- ▶ 3.1.8 Provide a method for users to start and terminate the packet tracing algorithm.

Member Responsibilities Cont.

Vinicius:

- ▶ 3.1.1 Open and close sockets as needed throughout the algorithm.
- ▶ 3.1.2 Use the ICMP protocol to work with packets to determine routes.
- ▶ 3.1.3 Manipulate the Time to Live (TTL) field.
- ▶ 3.1.4 Detect the network devices along the packet path.
- ▶ 3.1.5 Send packets along the path.
- ▶ 3.1.9 Determine the geolocation of a network device along the path.
- ▶ 3.2.4 The GUI will display the routes intuitively in a manner that can be understood easily by the user.

Member Responsibilities Cont.

Jose:

- ▶ 3.1.7 Provide an ordered list of network devices encountered along the calculated path.
- ▶ 3.1.6 Repeat trace route algorithm until the destination is reached.
- ▶ 3.2.4 The GUI will display the routes intuitively in a manner that can be understood easily by the user.
- ▶ 3.1.10 (Optional): Map the network devices to a Google map using the geolocations of the devices.

Non-Functional Requirements

- ❖ The program will use Java and Python together to create a functional and reliable application.
- ❖ JavaFX will be used to efficiently create a simple and user friendly application.
- ❖ Python will be used to implement the route-tracing algorithm involved.
- ❖ Packet Route Tracer will be designed for OSX/Linux and will be launched by clicking on a .jar file.

Resources and Materials

- ▶ [JavaFX](#)
- ▶ [Java](#)
- ▶ [Python](#)
- ▶ [Python Requests](#)

Design Overview

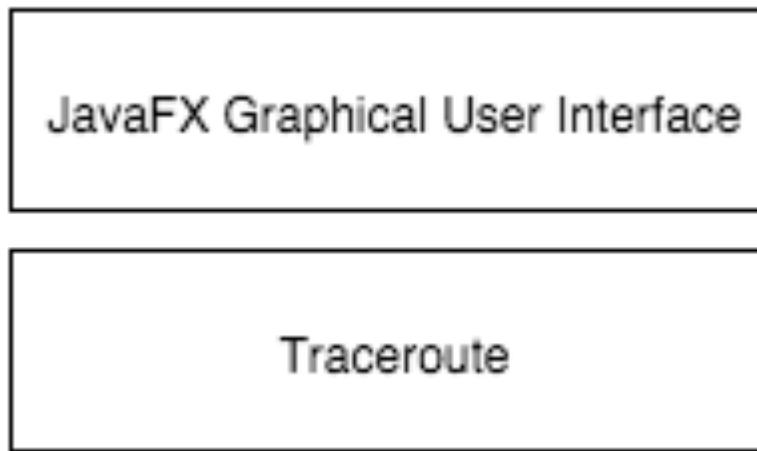


Figure 2: Packet Route Tracer layered architecture.

Design Overview Cont.

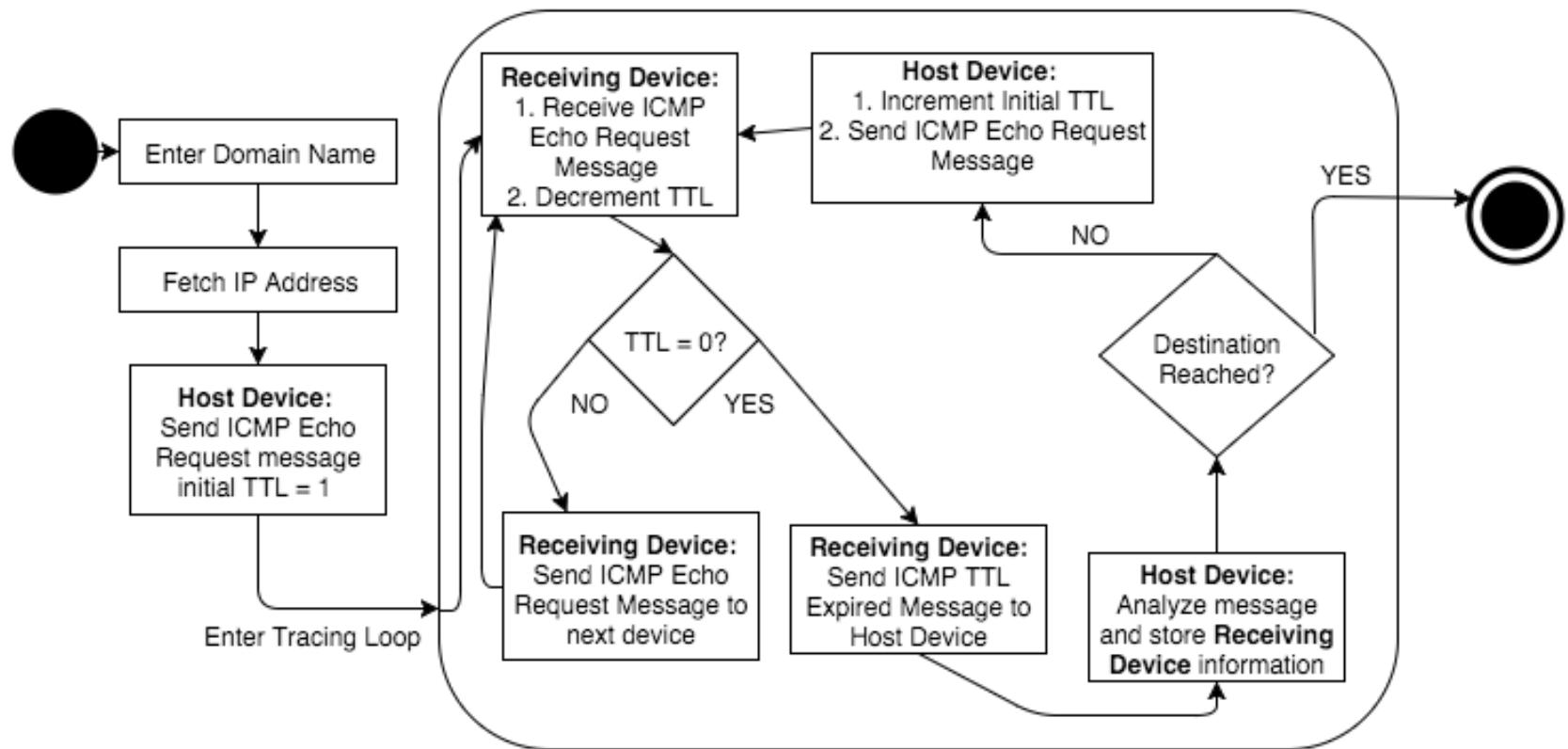
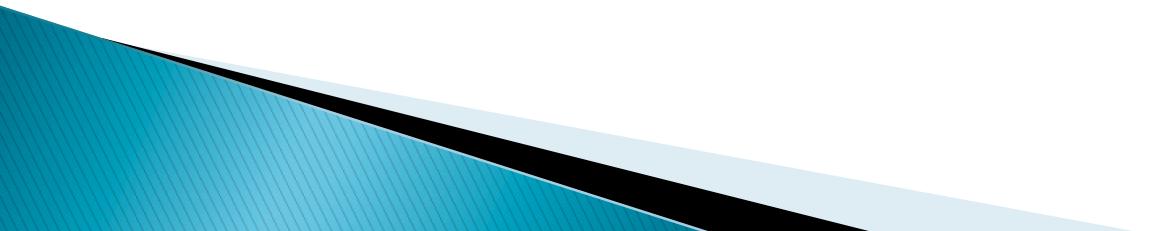


Figure 3: Traceroute class implementation of the trace route algorithm.

Detailed Design



Packet Tracing Component

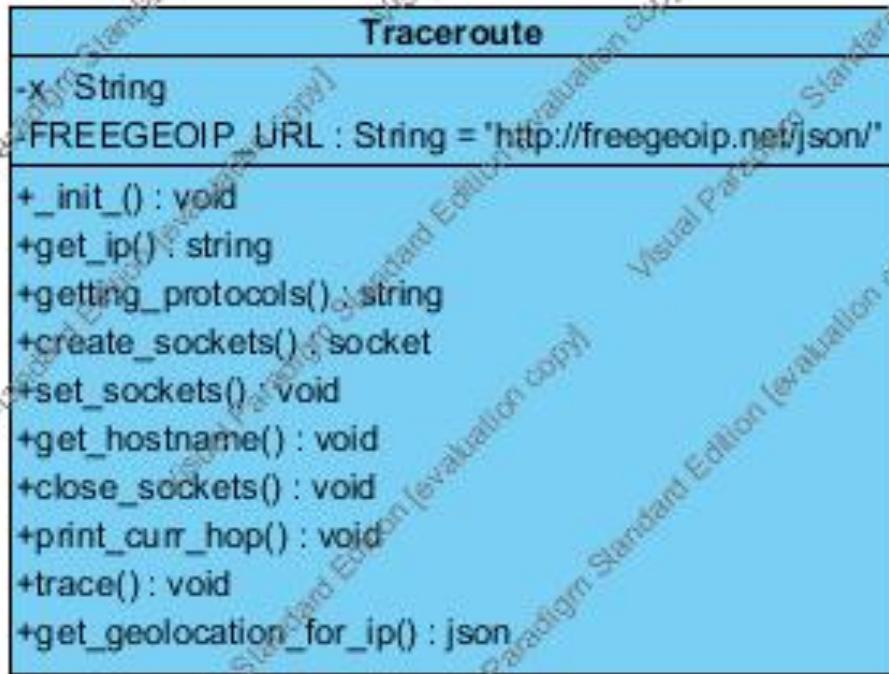


Figure 4: Traceroute class diagram

Packet Tracing Cont.

```
def trace(self):
    self.dest_addr = self.get_ip()

    self.icmp, self.udp = self.getting_protocols('icmp', 'udp')

    self.timeout = struct.pack("ll", 5, 0)

    while self.ttl < 20:

        self.recv_socket, self.send_socket = self.create_sockets()

        self.set_sockets()

        try:
            # getting data from receiving socket
            _, self.curr_addr = self.recv_socket.recvfrom(512)
            # _ is the data and curr_addr is a tuple with ip address and port, we care only for the first one
            self.curr_addr = self.curr_addr[0]

            self.get_hostname()

        except socket.error:
            pass

        finally:
            self.close_sockets()

        self.print_curr_hop()

        self.ttl += 1

        # when to stop
        if self.curr_addr == self.dest_addr or self.ttl > self.max_hops:
            break

x = Traceroute(sys.argv)
x.trace()
```

JavaFX FXML Application

Components:

- ❖ Main
- ❖ FXML style sheet
- ❖ Controller class

FXML Style Sheet

FXML Sample

```
1  ...
2  <HBox alignment="CENTER_RIGHT">
3      <children>
4          <LineChart fx:id="chart" prefHeight="410.0" prefWidth="551.0">
5              <xAxis>
6                  <NumberAxis fx:id="yAxis" forceZeroInRange="false" minorTickVisible="false"
7                      side="BOTTOM" tickLabelsVisible="false" tickMarkVisible="false"/>
8              </xAxis>
9              <yAxis>
10                 <NumberAxis forceZeroInRange="false" minorTickVisible="false" side="LEFT"
11                     tickLabelsVisible="false" tickMarkVisible="false" fx:id="xAxis"/>
12             </yAxis>
13         </LineChart>
14         <ListView fx:id="list_results" prefHeight="410.0" prefWidth="283.0" />
15     </children>
16 </HBox>
17 ...
```

GUI/JavaFX Controller

Visual Paradigm Standard Edition(College of Charleston)

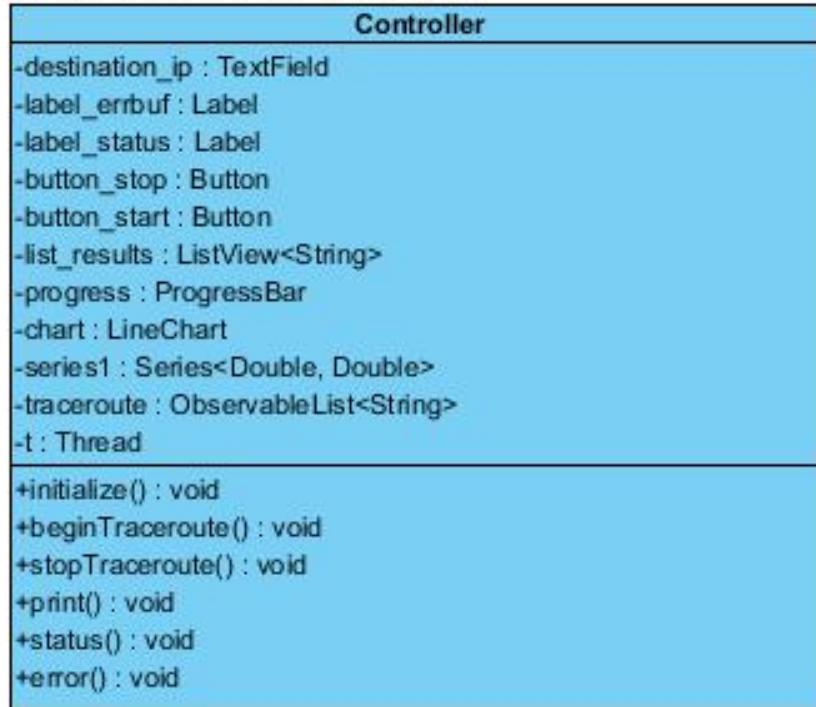


Figure 5: Class diagram for the JavaFX Controller.

GUI/JavaFX Controller Cont.

```
public void beginTraceroute() throws IOException {
    traceroute.removeAll();
    list_results.getItems().clear();
    series1.getData().clear();
    error();

    status("Loading, this can take some time...");
    // Loader
    progress.setProgress(-1.0f);
    button_start.setDisable(true);
    button_stop.setDisable(false);
    try {
        ProcessBuilder pd = new ProcessBuilder().command("sudo", "python", "trace.py", destination_ip.getText());
        pd.redirectErrorStream(true);
        Process p = pd.start();

        Task task = new Task() {
            @Override
            protected Object call() throws Exception {
                try (BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()))) {
                    String line, message = null;

                    while ((line = in.readLine()) != null) {
                        if (!Objects.equals(line.substring(0, 1), "[")) {
                            if (Objects.equals(line.substring(0, 9), "Traceback")) {
                                message = "Please insert a valid domain name";
                            } else if (Objects.equals(line.substring(0, 4), "sudo")) {
                                message = "Please run this app with sudo privileges!";
                            } else {
                                message = "Unknown error occurred!";
                            }
                            break;
                        } else {
                            traceroute.add(line);
                        }
                    }
                }
            }
        };
    }
}
```

GUI/JavaFX Controller Cont.

```
        final String finalMessage = message;
        Platform.runLater(() -> {
            stopTraceroute();
            if (finalMessage != null) {
                error(finalMessage);
                series1.getData().clear();
            } else {
                double i = 5.0;
                int ii = 1;
                for (String s : traceroute) {
                    double height = Math.random() * 100;
                    final XYChart.Data<Double, Double> data = new XYChart.Data<>(i, height);
                    String[] output = s.split("", 5);
                    data.setNode(new HoveredThresholdNode(output[3], i));
                    series1.getData().add(data);
                    i += 5;
                    ii++;
                }
            }
        });
    } catch (IOException e) {
        System.out.print(p.getErrorStream());
    }
    return null;
}
};

t = new Thread(task);
t.setDaemon(true);
t.start();

} catch (Exception e) {
    System.out.println(e);
}
}
```

Interaction?

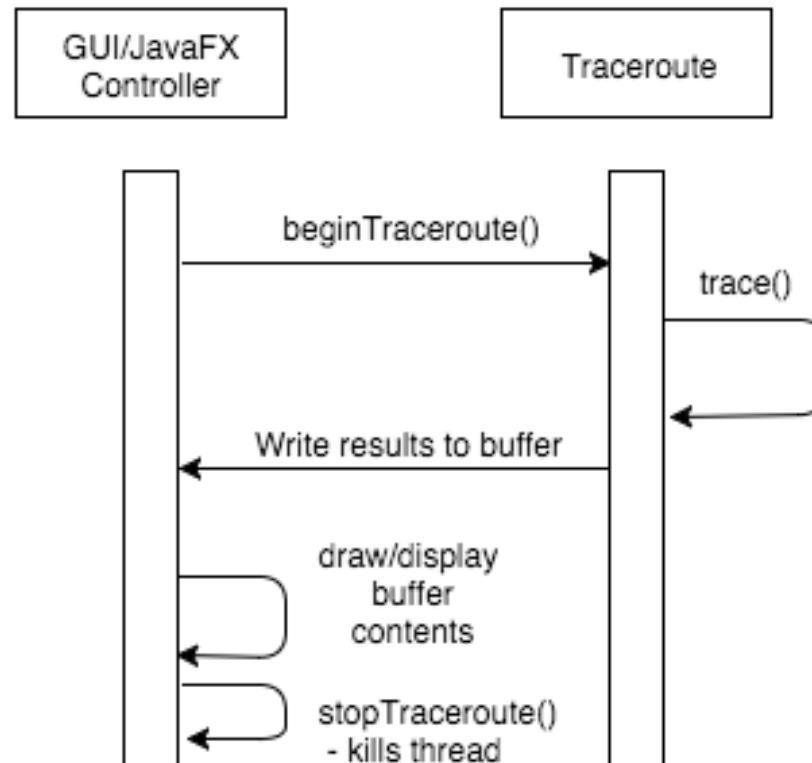


Figure 6: Sequence diagram for interaction
between the GUI and Traceroute.

Demo time!