Traceroute

Software Documentation

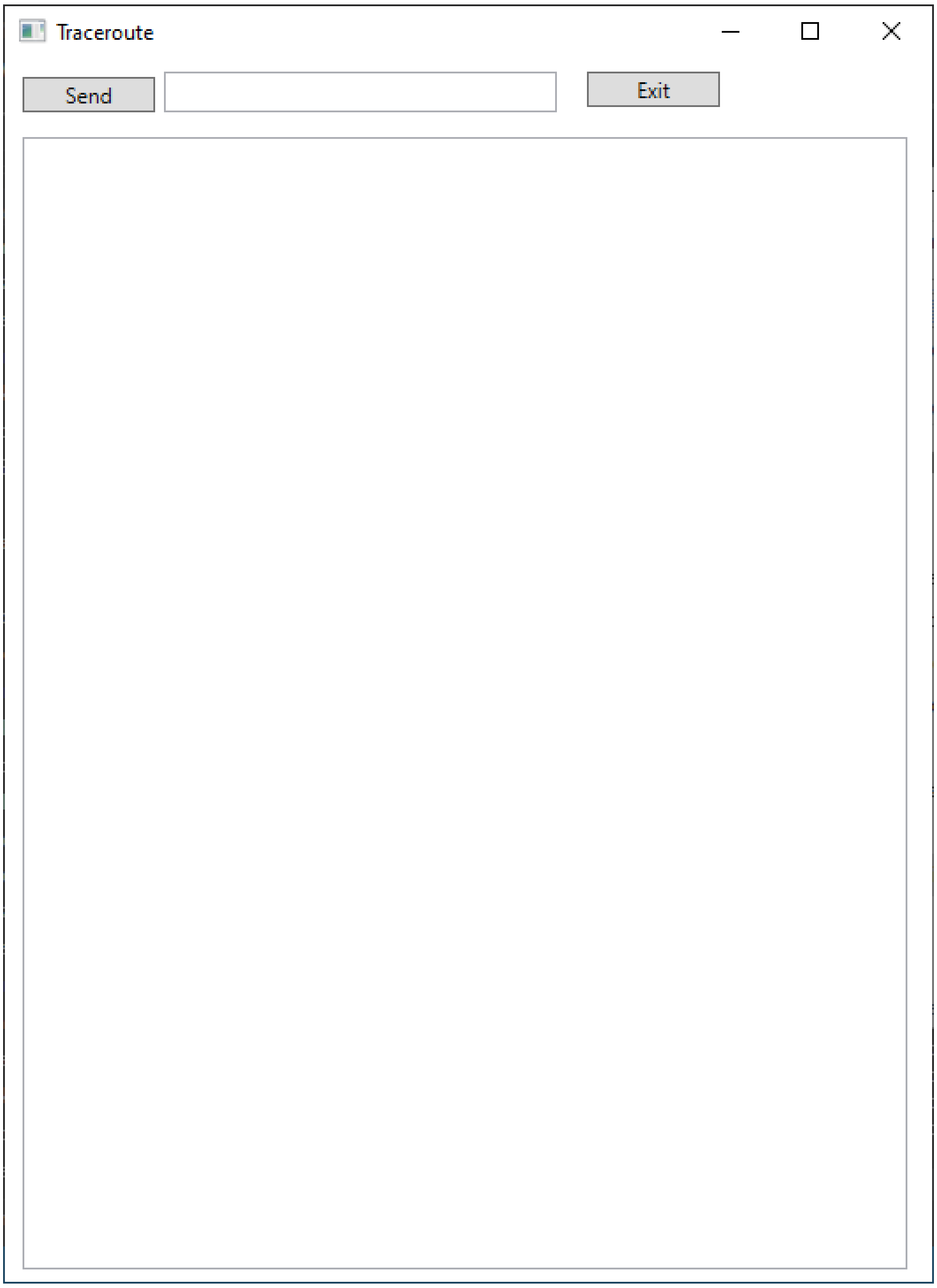
Author: matiwa

Table of contents

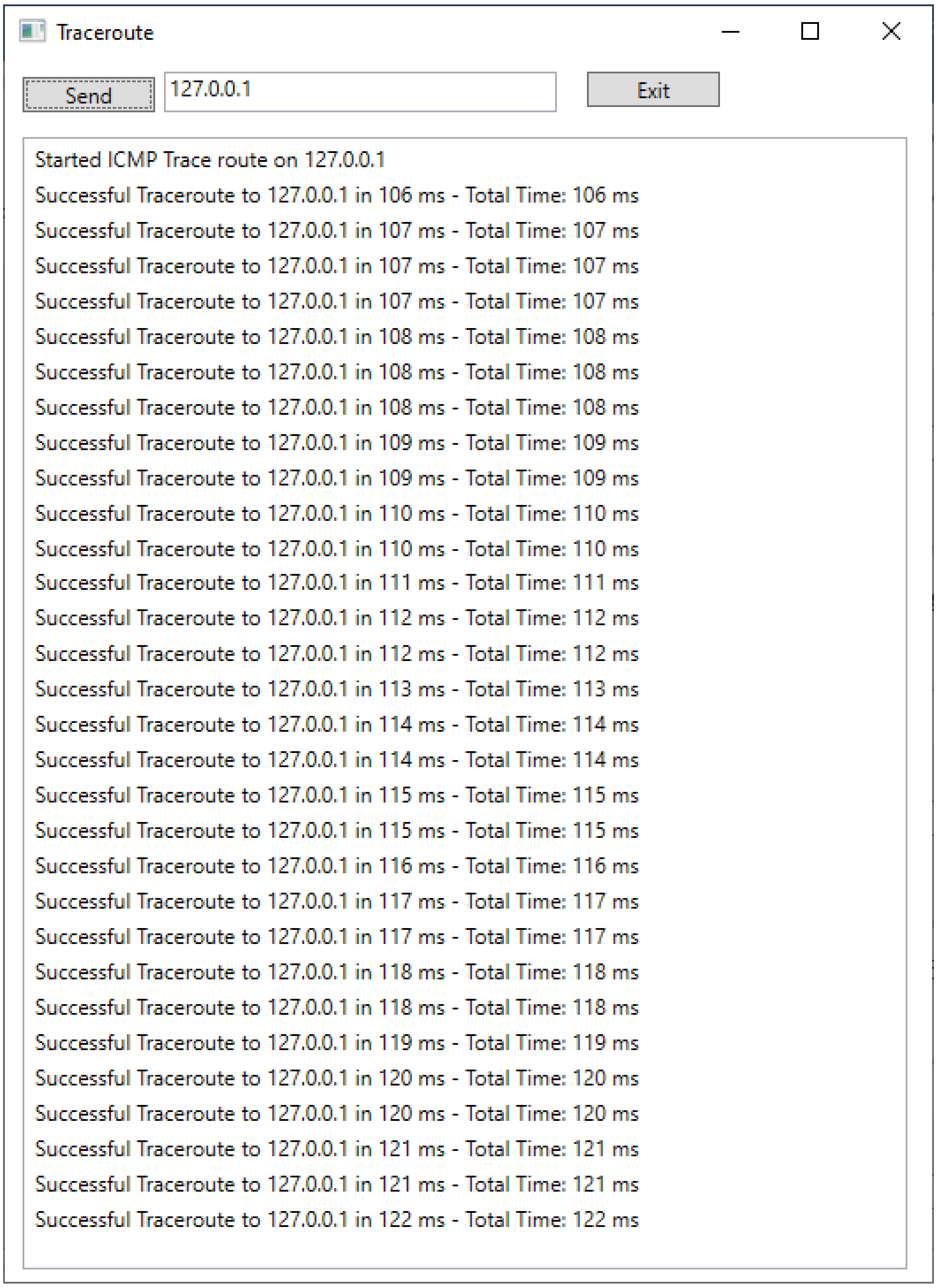
Introduction

This software documentation includes: description of the application's operation, what is needed for use, algorithms used, interface description and source code description. This application is used to study the route of packets in an IP network.

Describing of the application’s operation

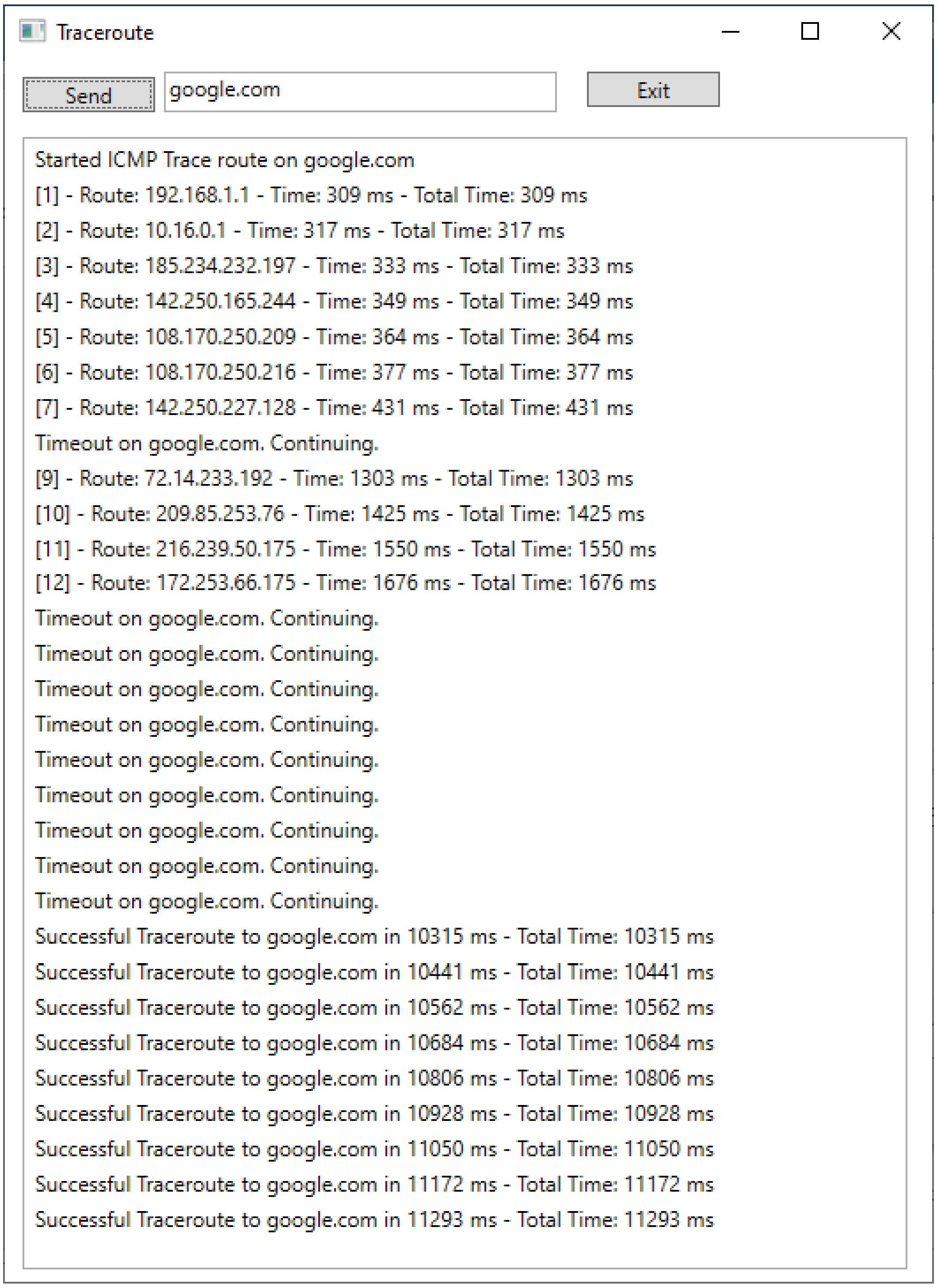


Drawing 1: The beginning of the application's operation [own study]



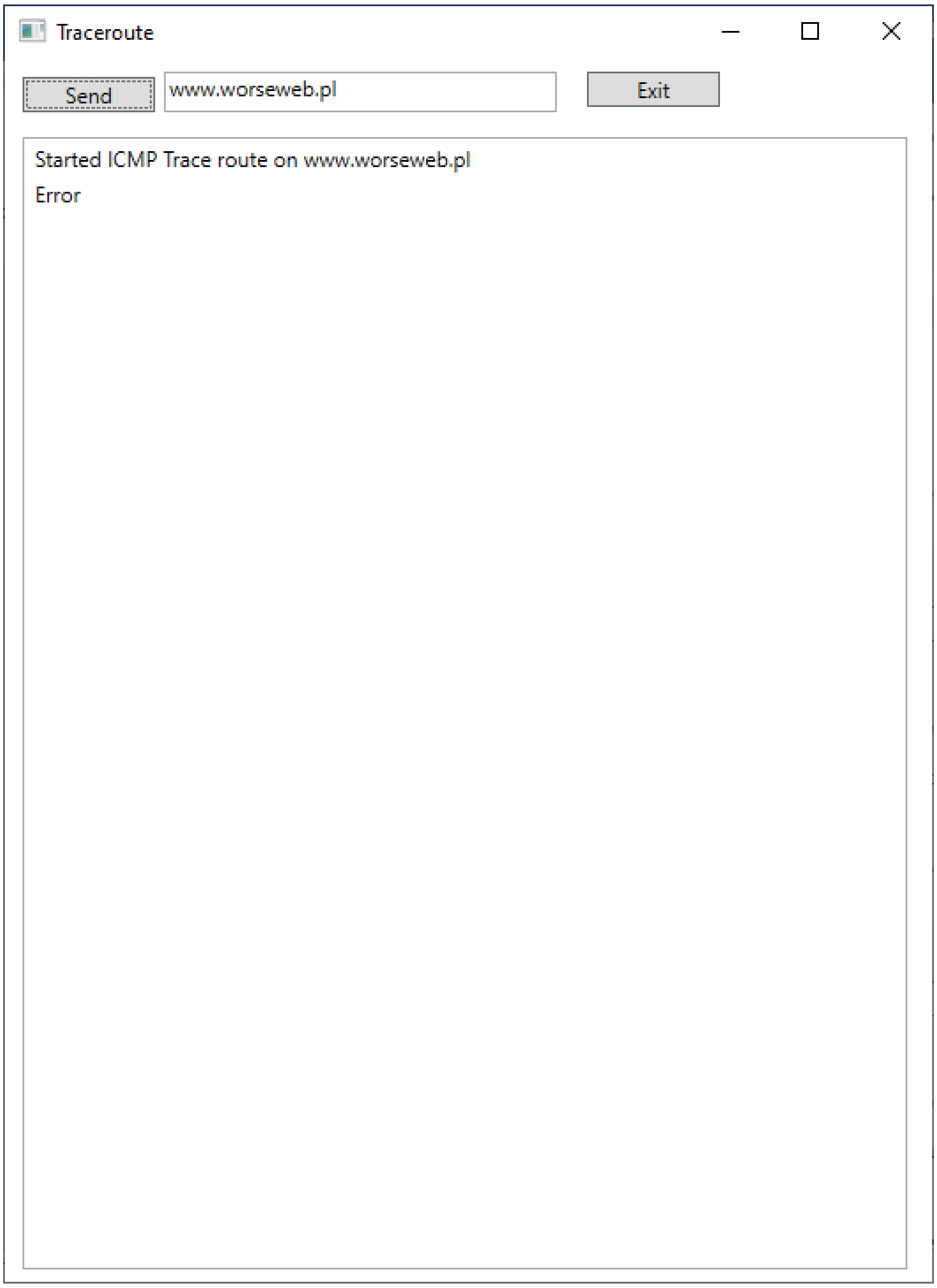
```
Started ICMP Trace route on 127.0.0.1
Successful Traceroute to 127.0.0.1 in 106 ms - Total Time: 106 ms
Successful Traceroute to 127.0.0.1 in 107 ms - Total Time: 107 ms
Successful Traceroute to 127.0.0.1 in 107 ms - Total Time: 107 ms
Successful Traceroute to 127.0.0.1 in 107 ms - Total Time: 107 ms
Successful Traceroute to 127.0.0.1 in 108 ms - Total Time: 108 ms
Successful Traceroute to 127.0.0.1 in 108 ms - Total Time: 108 ms
Successful Traceroute to 127.0.0.1 in 108 ms - Total Time: 108 ms
Successful Traceroute to 127.0.0.1 in 109 ms - Total Time: 109 ms
Successful Traceroute to 127.0.0.1 in 109 ms - Total Time: 109 ms
Successful Traceroute to 127.0.0.1 in 110 ms - Total Time: 110 ms
Successful Traceroute to 127.0.0.1 in 110 ms - Total Time: 110 ms
Successful Traceroute to 127.0.0.1 in 111 ms - Total Time: 111 ms
Successful Traceroute to 127.0.0.1 in 112 ms - Total Time: 112 ms
Successful Traceroute to 127.0.0.1 in 112 ms - Total Time: 112 ms
Successful Traceroute to 127.0.0.1 in 113 ms - Total Time: 113 ms
Successful Traceroute to 127.0.0.1 in 114 ms - Total Time: 114 ms
Successful Traceroute to 127.0.0.1 in 114 ms - Total Time: 114 ms
Successful Traceroute to 127.0.0.1 in 115 ms - Total Time: 115 ms
Successful Traceroute to 127.0.0.1 in 115 ms - Total Time: 115 ms
Successful Traceroute to 127.0.0.1 in 116 ms - Total Time: 116 ms
Successful Traceroute to 127.0.0.1 in 117 ms - Total Time: 117 ms
Successful Traceroute to 127.0.0.1 in 117 ms - Total Time: 117 ms
Successful Traceroute to 127.0.0.1 in 118 ms - Total Time: 118 ms
Successful Traceroute to 127.0.0.1 in 118 ms - Total Time: 118 ms
Successful Traceroute to 127.0.0.1 in 119 ms - Total Time: 119 ms
Successful Traceroute to 127.0.0.1 in 120 ms - Total Time: 120 ms
Successful Traceroute to 127.0.0.1 in 120 ms - Total Time: 120 ms
Successful Traceroute to 127.0.0.1 in 121 ms - Total Time: 121 ms
Successful Traceroute to 127.0.0.1 in 121 ms - Total Time: 121 ms
Successful Traceroute to 127.0.0.1 in 122 ms - Total Time: 122 ms
```

Drawing 2: The effect after entering the IPv4 address of the localhost [own study]



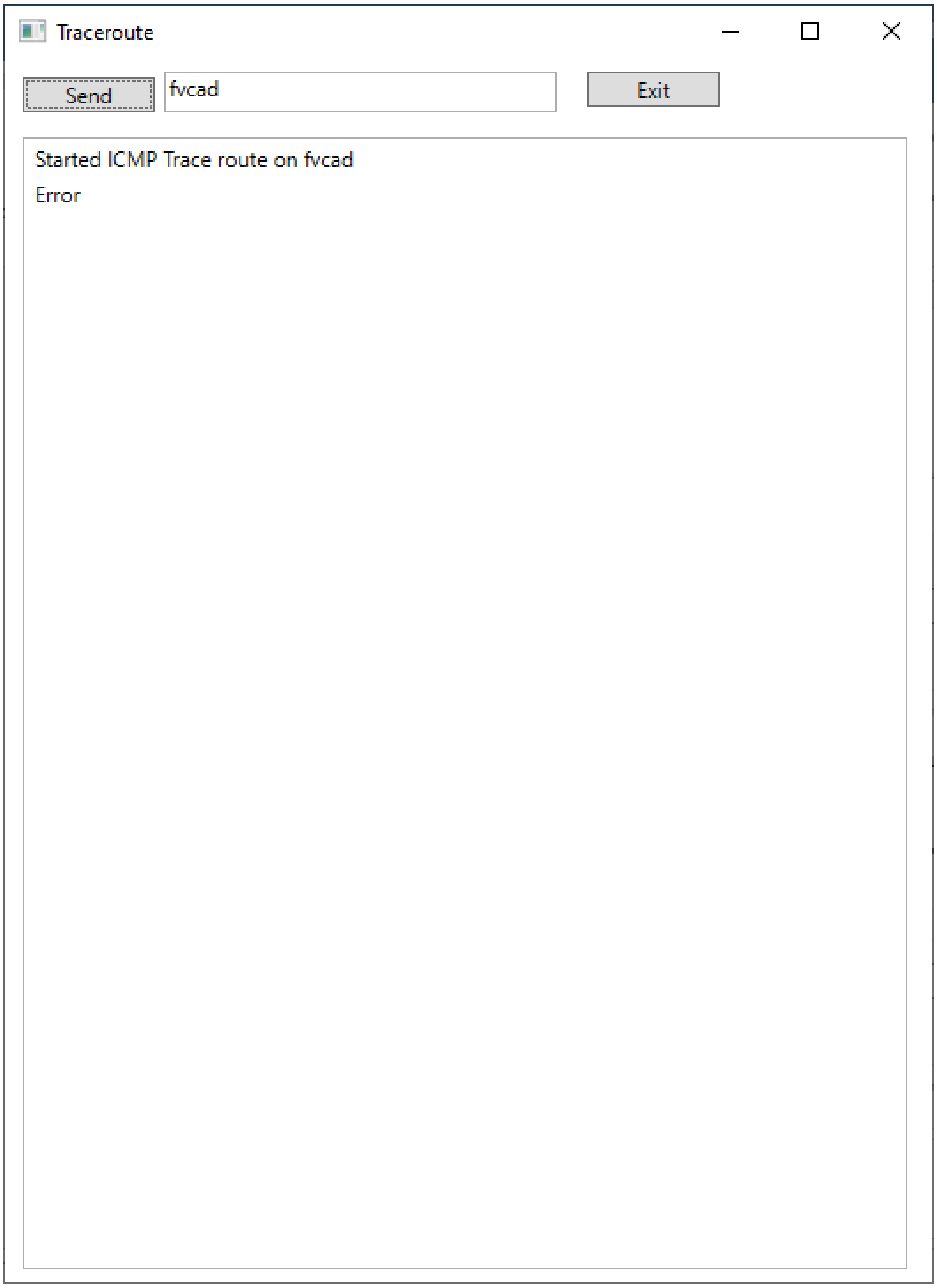
```
Started ICMP Trace route on google.com
[1] - Route: 192.168.1.1 - Time: 309 ms - Total Time: 309 ms
[2] - Route: 10.16.0.1 - Time: 317 ms - Total Time: 317 ms
[3] - Route: 185.234.232.197 - Time: 333 ms - Total Time: 333 ms
[4] - Route: 142.250.165.244 - Time: 349 ms - Total Time: 349 ms
[5] - Route: 108.170.250.209 - Time: 364 ms - Total Time: 364 ms
[6] - Route: 108.170.250.216 - Time: 377 ms - Total Time: 377 ms
[7] - Route: 142.250.227.128 - Time: 431 ms - Total Time: 431 ms
Timeout on google.com. Continuing.
[9] - Route: 72.14.233.192 - Time: 1303 ms - Total Time: 1303 ms
[10] - Route: 209.85.253.76 - Time: 1425 ms - Total Time: 1425 ms
[11] - Route: 216.239.50.175 - Time: 1550 ms - Total Time: 1550 ms
[12] - Route: 172.253.66.175 - Time: 1676 ms - Total Time: 1676 ms
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Timeout on google.com. Continuing.
Successful Traceroute to google.com in 10315 ms - Total Time: 10315 ms
Successful Traceroute to google.com in 10441 ms - Total Time: 10441 ms
Successful Traceroute to google.com in 10562 ms - Total Time: 10562 ms
Successful Traceroute to google.com in 10684 ms - Total Time: 10684 ms
Successful Traceroute to google.com in 10806 ms - Total Time: 10806 ms
Successful Traceroute to google.com in 10928 ms - Total Time: 10928 ms
Successful Traceroute to google.com in 11050 ms - Total Time: 11050 ms
Successful Traceroute to google.com in 11172 ms - Total Time: 11172 ms
Successful Traceroute to google.com in 11293 ms - Total Time: 11293 ms
```

Drawing 3: The effect after entering the website [own study]



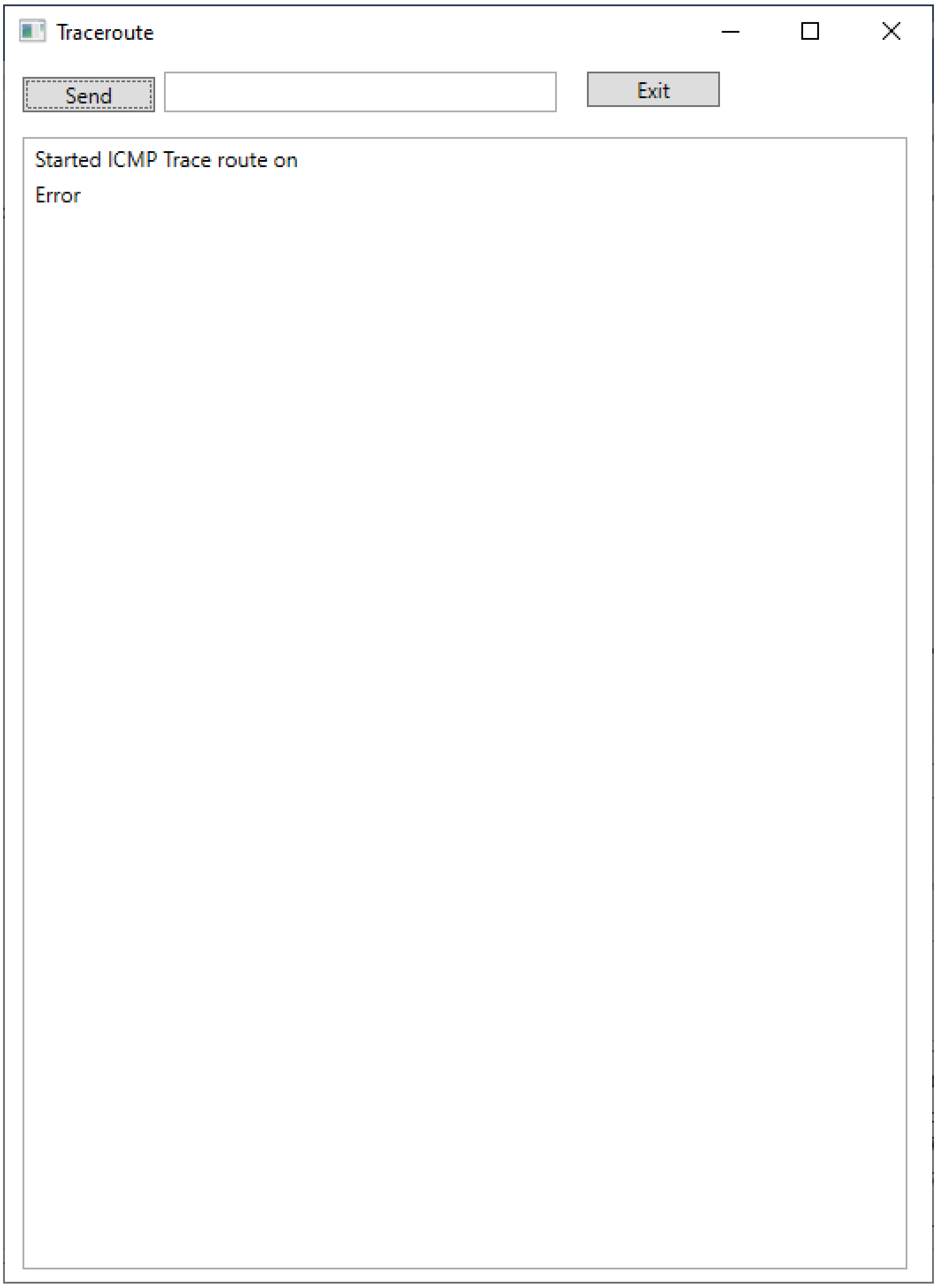
```
Started ICMP Trace route on www.worseweb.pl
Error
```

Drawing 4: Effect after entering a wrong address [own study]



```
Started ICMP Trace route on fvcad
Error
```

Drawing 5: Effect after entering random characters [own study]



```
Started ICMP Trace route on
Error
```

Drawing 6: Effect after typing nothing [own study]

After starting the program, the user enters the IP address or website into the text field. By clicking the Send button, the program procedure for examining the packet route in the IP network is started, the operation of which is based on the UDP and ICMP protocols. If there is a network connection (when connecting to a host other than localhost) and the IP Address or WWW are correct, feedback is displayed. Otherwise, "Error" will be displayed in case of invalid address, random characters (not IP and WWW address) or no connection. It is possible that the user is not staring at anything and typing nothing, then the "Error" message will also appear. There are certainly bugs that the developer did not discover while working on the application.

What is needed for use?

The application does not require installation. It only needs the Windows operating system.

Algorithm used

The basic form of the algorithm can be deduced from the previous section. It only needs the Windows operating system. In summary, the application is the command line equivalent of Tracert on Windows systems, and Traceroute on Unix systems.

The traceroute operation relies on the UDP and ICMP communication protocols.

- The packet is sent first with the TTL (Time To Live) field set to 1. This value is decreased as it traverses subsequent routers in the route. If the TTL field reaches the value 0, the packet is dropped by the router which then sends back an ICMP "Time Exceeded" message. This is how the source computer obtains the IP address of the first router in the route.

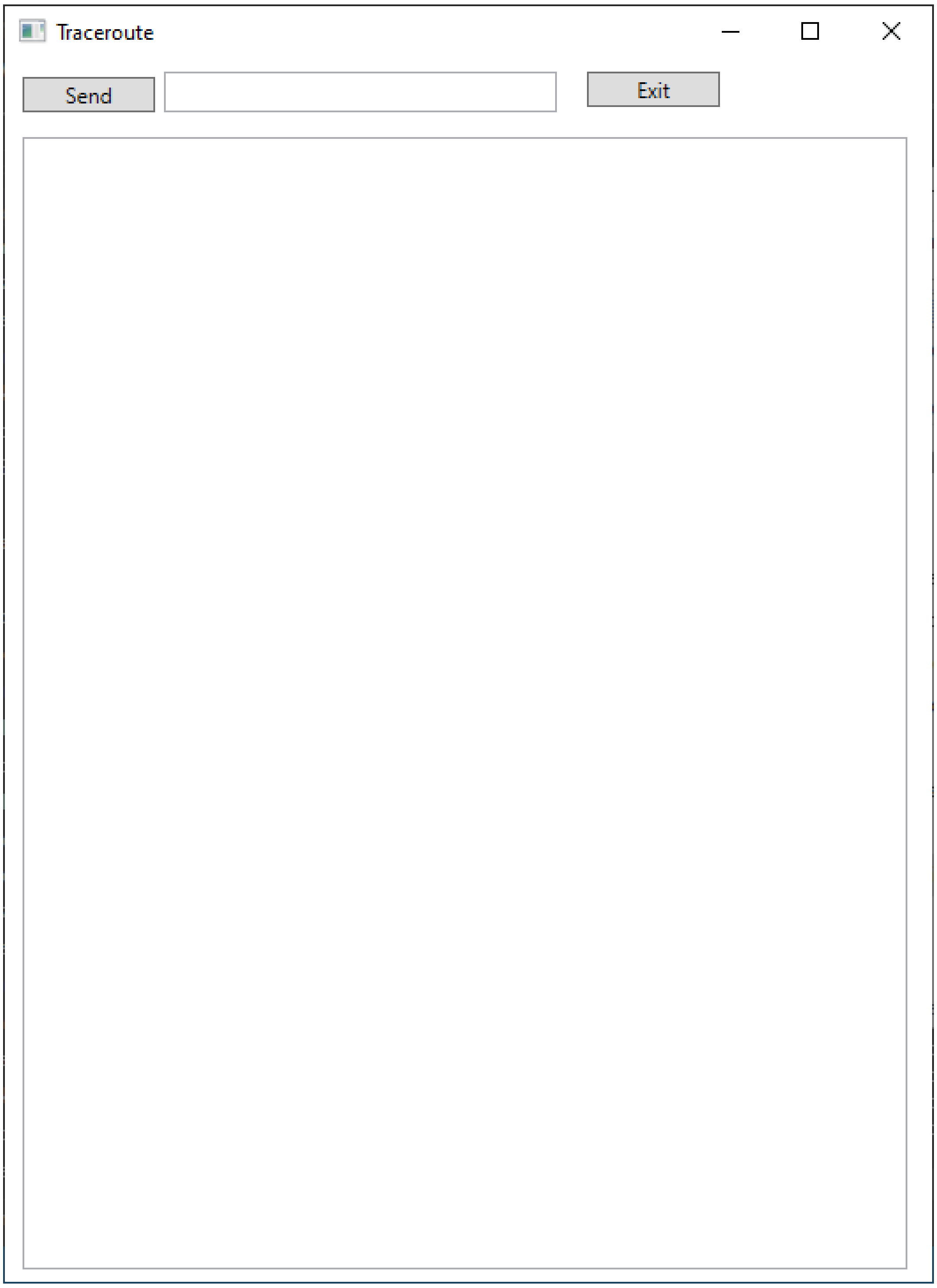
- In the next packet sent from the source computer, the TTL field is set to 2. The first router will reduce this value to 1 and pass it to the second router in the route. The second router will behave like the first one before, ie it will reduce TTL to 0 and discard the packet sending "Time Exceeded" message to the source computer.

- Information about the next routers on the route is obtained in the same way as in the above two points. You just incrementally increment the TTL field value by 1 in the outgoing packets.

- If the packet eventually reaches the target host, an ICMP "Port Unreachable" message will most likely be returned. This is because Unix traceroute implementations are deliberately sending UDP packets with a port number greater than 30000. Typically, no services are running on a port with this high number, so no process on the target computer will try to handle the incoming packet. In this case, the route study is terminated.

Similar tools as mtr and the tracert program in the Microsoft Windows family are implemented slightly differently. The main principle of operation, consisting in gradually increasing the TTL field in the sent packets, remains the same. The difference is that the packets sent are not UDP datagrams, but ICMP "Echo Request" messages. If such a message reaches its destination, an "Echo Reply" will always be returned. This saves you from having to rely on the assumption of high port numbers for UDP datagrams, and you can bypass some firewalls. [1]

Interface description



Drawing 7: Graphical interface [own study]

The interface is typical for a Windows Presentation Foundation. There are essential components: listbox, textbox and two buttons.

Source code description

The project was made in the C# programming language, in the Visual Studio Community 2017 programming environment. All work was done on the Windows 10 operating system. The application's source code looks like this.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Net.NetworkInformation;
using System.Threading.Tasks;
using System.Diagnostics;

namespace Traceroute
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void SendButton_Click(object sender, RoutedEventArgs e)
        {
            outlist.Items.Clear();
            string hostname = addresslist.Text;
            int timeout = 1000;
            int max_ttl = 30;
            int current_ttl = 0;
            const int bufferSize = 32;
            Stopwatch s1 = new Stopwatch();
            Stopwatch s2 = new Stopwatch();
            byte[] buffer = new byte[bufferSize];
            new Random().NextBytes(buffer);
            Ping pinger = new Ping();
            Task.Factory.StartNew(() =>
            {
                WriteListBox($"Started ICMP Trace route on {hostname}");
                for(int ttl = 1; ttl <= max_ttl; ttl++)
                {
                    current_ttl++;
                    s1.Start();
                    s2.Start();
                    PingOptions options = new PingOptions(ttl, true);
                    PingReply reply = null;
                    try
                    {
                        reply = pinger.Send(hostname, timeout, buffer, options);
                    }
                    catch
                    {
                        WriteListBox("Error");
                        break;
                    }
                    if (reply != null)
                    {
                        if (reply.Status == IPStatus.TtlExpired)
                        {
                            WriteListBox($"[{ttl}] - Route: {reply.Address} - Time: 
{s1.ElapsedMilliseconds} ms - " +
                                $"Total Time: {s2.ElapsedMilliseconds} ms");
                            continue;
                        }
                        if (reply.Status == IPStatus.TimedOut)
                        {
                            WriteListBox($"Timeout on {hostname}. Continuing.");
                            continue;
                        }
                        if (reply.Status == IPStatus.Success)
                        {
                            WriteListBox($"Successful Traceroute to {hostname} in 
{s1.ElapsedMilliseconds} ms - " +
                                $"Total Time: {s2.ElapsedMilliseconds} ms");
                            s1.Stop();
                            s2.Stop();
                        }
                    }
                }
            });
        }

        private void WriteListBox(string text)
        {
            Dispatcher.BeginInvoke(new Action(() =>
            {
                outlist.Items.Add(text);
            }));
        }

        private void BExit_Click(object sender, RoutedEventArgs e)
        {
            Environment.Exit(0);
        }
    }
}
```

Listing 1: Source code [own study]

List of drawings

List of listings

Bibliography

[1] https://pl.wikipedia.org/wiki/Traceroute