CS305 2022 Fall Lab Assignment 2 - Raw Socket and ICMP

Introduction

The TCP/IP protocol family can be divided into five layers: application layer, transport layer, network layer, link layer, and physical layer.

For **standard sockets**, usually, the data is automatically encapsulated according to the selected transport layer protocol (such as TCP, or UDP).

The **raw socket** is a type of socket that allows access to the underlying transport provider, which means the raw socket allows the direct sending/receiving of IP protocol packets **without** any transport layer protocol format.

Internet Control Message Protocol(**ICMP**) is a ~~supporting~~ protocol in the TCP/IP protocol family. It is used by network devices, including routers, to send error messages and operational information indicating success or failure when communicating with another IP address. For details, you can refer to RFC 792: INTERNET CONTROL MESSAGE PROTOCOL.

In this assignment, you need to finish all the following tasks using the **raw socket**.

Tasks

Task 1 Checksum (20 points (10 points for generation and 10 points for check))

The checksum is used to verify the integrity of packets, and it is carried in the header of an IP packet. Please note that there are not only checksums in IP packets, but also checksums in ICMP, UDP, and TCP packets. They correspond to the verification information of different packets or packet headers, please pay attention to distinguish them.

In this task, we hope you can implement the checksum mechanism of the ICMP packets. For how to compute the checksum of a packet, you can refer to RFC 1071.

We have provided a simple ICMP class in `sockets.py`. You need to complete the code in method `_checksum(self, data)` and `_check_data(self, data, checksum)` of the file, the `_checksum(self, data)` method is to generate the checksum while the `_check_data(self, data, checksum)` method is to verify the integrity of the given data.

💡 Hint: Please pay attention to that, one importance to test in this task is the **correctness** of the calculation of the checksum.

❗ Apart from that, we believe that the smart students of SUSTech must have noticed that the basic unit for checksum is 2 bytes, so if you encounter a message with an odd length, you need to fill in the necessary padding (filling position) to meet the requirement of the calculation of the checksum.

Task 2 ICMP Request Message (20 points)

In this task, we hope that you can correctly construct an ICMP request message according to an id, a sequence number, and a payload. The type of the ICMP request message is 8 and the code is 0.

The only method you need to complete is `_create_packet(self, request: ICMPRequest)` in `sockets.py`.

⭐ Hint: Based on Task 1, the request message created should contain a correct checksum and the requested payload.

Task 3 ICMP Reply Message (20 points)

In this task, we hope you can parse an ICMP reply message correctly.

The **only** method you need to complete is `_parse_reply(self, packet, source, current_time)` in `sockets.py`.

🎯 Hint: The key point of this task lies in that, you should extract and parse the id, sequence, type, code, etc. of certain ICMP packets.

In normal cases, the ICMP Echo Message of Echo Reply Message is shown below. The Type is equal to 8 for an echo message and 0 for an echo reply message.

```
Echo or Echo Reply Message

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |     Code      |          Checksum             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Identifier          |        Sequence Number        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Data ...
   +-+-+-+-+-
```

What's more, there will be some message that appears in some abnormal response, called the **ICMP error message**. Take `Time Exceeded Message` for an example, its structure is shown below.

```
Time Exceeded Message

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |     Code      |          Checksum             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             unused                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Internet Header + 64 bits of Original Data Datagram      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In the error message, the type is not equal to 0(In this Time Exceeded Message, the type is equal to 11), and the structure is different from the normal ICMP message, so you may treat them especially to parse their **type,** and **code**, and at the meantime, read the id and sequence of the original data datagram.

When receiving a message, you should first check the checksum, if the check fails, then you should report the error and terminate the analysis, if the check succeeds, congratulations, then you can start parsing it.

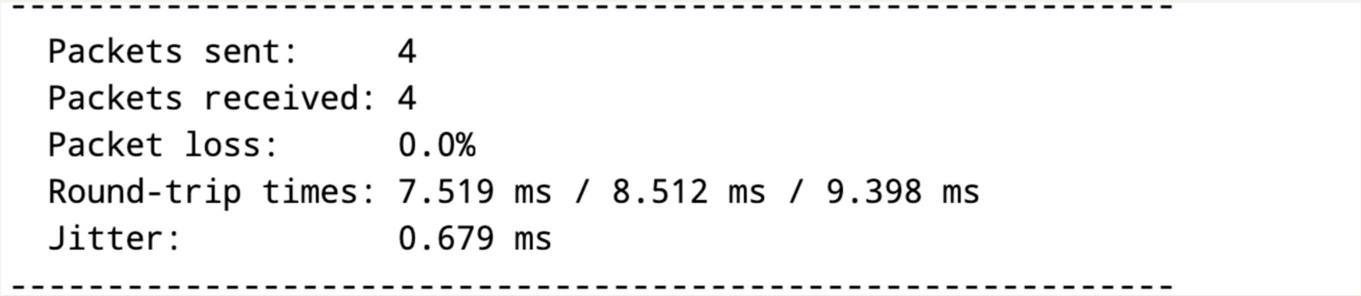
Task 4 Ping (20 points)

Ping is a utility used to test the reachability of a host on an IP network, and we have learned that `ping` operates using ICMP packets. Pinging involves sending an **ICMP echo request** to the target host and waiting for an **ICMP echo reply**.

In this task, we hope that you can implement the procedure of sending a testing message under the `ping` command, including waiting for the ICMP echo reply, parsing the reply message, and printing the statics. You only need to complete the `ping(address, n=4, payload=None,id=None)` in the `ping.py`.

☘️ Hint: During the ping process, you need to be able to customize the **number**, **payload**, and **id** of the sent packets. Since the purpose of ping is to test the reachability of a host on an IP network, accurate calculation of **RTT** and **packet loss rate** from received packets is also one of the factors considered in the test.

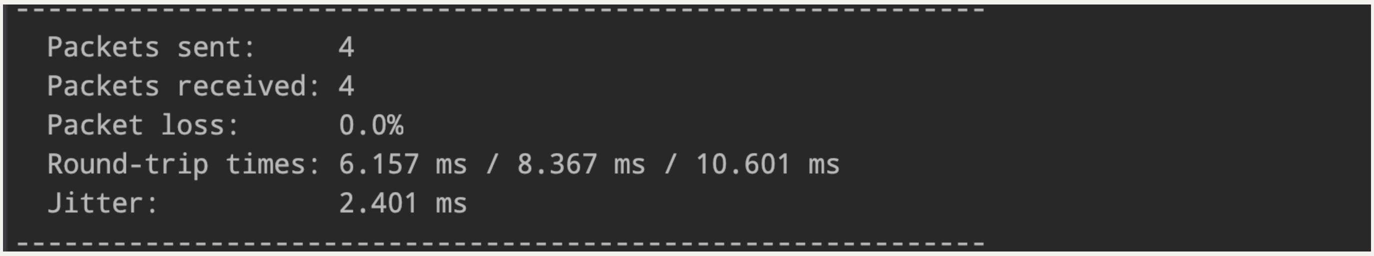
After successfully implementing the ping method, use `python ping.py www.baidu.com` to run ping.py, here is an example of testing the connectivity of the Baidu website, you will get an output similar to this, as shown below.



```
------------------------------------------------------------
  Packets sent:      4
  Packets received:  4
  Packet loss:       0.0%
  Round-trip times:  7.519 ms / 8.512 ms / 9.398 ms
  Jitter:            0.679 ms
------------------------------------------------------------
```

In the output message, the **sent packet** is the number of packets transmitted to the destination host. The **received packet** is the number of ICMP responses received from the remote host. **Packet loss** refers to the proportion of the package that cannot reach the endpoint of the sent packages. The **Round-trip times** here are the minimum, average, and maximum of the RTT calculated in ping. The **jitter** in milliseconds is defined as the variance of the latency of packets flowing through the network.

After you have completed this method, you can customize the payload by running `python ping.py www.baidu.com --p 'hello world'`, the result is as below.



```
--------------------------------------------------------------
  Packets sent:     4
  Packets received: 4
  Packet loss:      0.0%
  Round-trip times: 6.157 ms / 8.367 ms / 10.601 ms
  Jitter:           2.401 ms
--------------------------------------------------------------
```

Task 5 Tracert (20 points)

Apart from `ping`, the `tracert` will also make use of the ICMP packets. The main purpose of `tracert` is to detect the routers from the source to the destination, and the distance of these routers from the source

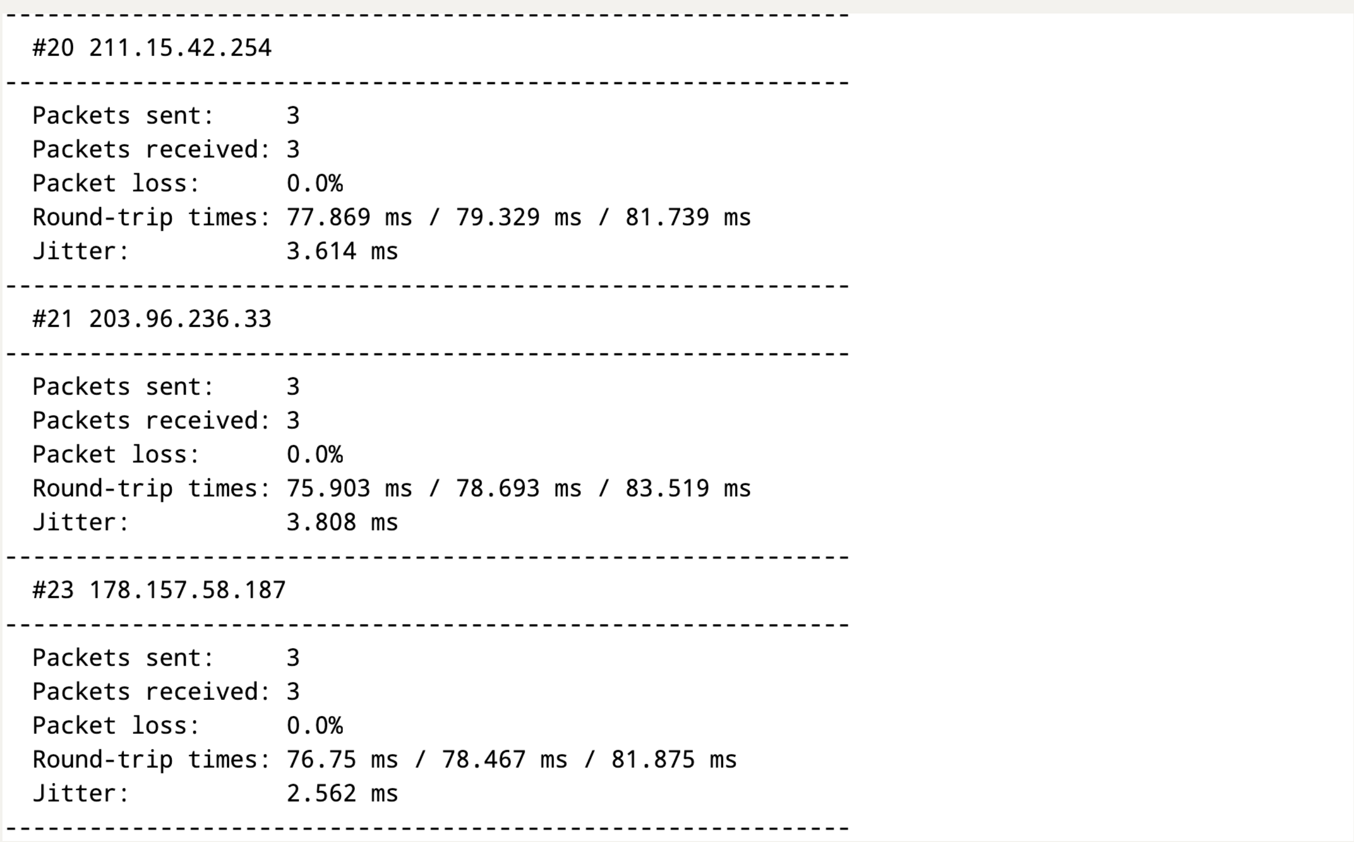
In this task, we hope that you can implement the procedure of sending a testing message under the `tracert` command. You only need to complete the `tracert(address, id=None)` in the `tracert.py`.

🌟 Hint: In the process of tracert, we hope that you can customize the id of the sent message, and stop the cycle of sending the message when the message has reached the destination host. In this process, you should also combine the principle of tracert to appropriately change the **value of TTL**. In addition, the most important thing is that you should analyze the address of each hop from the received message, and finally, find the corresponding quest information and accurately calculate the **RTT value**.

After successfully implementing the `tracert` method, use `python tracert.py jp.nuxjc.com` to run `tracert.py`, here is an example of testing the connectivity of the Baidu website, you will get an output similar to this, as shown below.

```
  #1  10.10.10.10
------------------------------------------------------------
  Packets sent:     3
  Packets received: 3
  Packet loss:      0.0%
  Round-trip times: 4.032 ms / 5.285 ms / 6.626 ms
  Jitter:           1.879 ms
------------------------------------------------------------
  #2  10.23.255.30
------------------------------------------------------------
  Packets sent:     3
  Packets received: 2
  Packet loss:      33.0%
  Round-trip times: 5.176 ms / 15.215 ms / 25.254 ms
  Jitter:           20.078 ms
------------------------------------------------------------
  #3  10.23.255.83
------------------------------------------------------------
  Packets sent:     3
  Packets received: 3
  Packet loss:      0.0%
  Round-trip times: 3.419 ms / 4.315 ms / 5.61 ms
  Jitter:           1.344 ms
------------------------------------------------------------
```

...

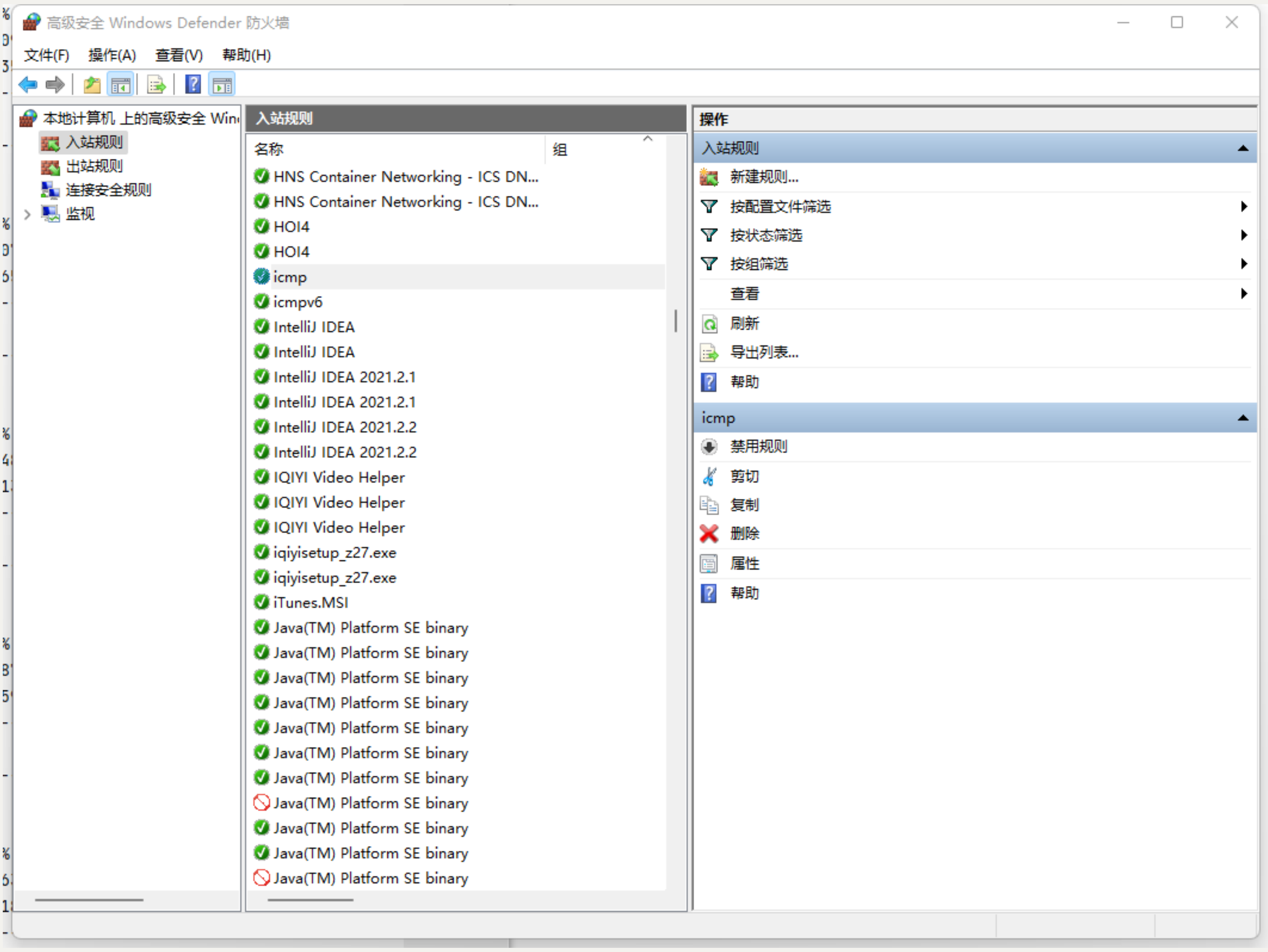


```
------------------------------------------------------------
  #20 211.15.42.254
------------------------------------------------------------
  Packets sent:     3
  Packets received: 3
  Packet loss:      0.0%
  Round-trip times: 77.869 ms / 79.329 ms / 81.739 ms
  Jitter:           3.614 ms
------------------------------------------------------------
  #21 203.96.236.33
------------------------------------------------------------
  Packets sent:     3
  Packets received: 3
  Packet loss:      0.0%
  Round-trip times: 75.903 ms / 78.693 ms / 83.519 ms
  Jitter:           3.808 ms
------------------------------------------------------------
  #23 178.157.58.187
------------------------------------------------------------
  Packets sent:     3
  Packets received: 3
  Packet loss:      0.0%
  Round-trip times: 76.75 ms / 78.467 ms / 81.875 ms
  Jitter:           2.562 ms
------------------------------------------------------------
```

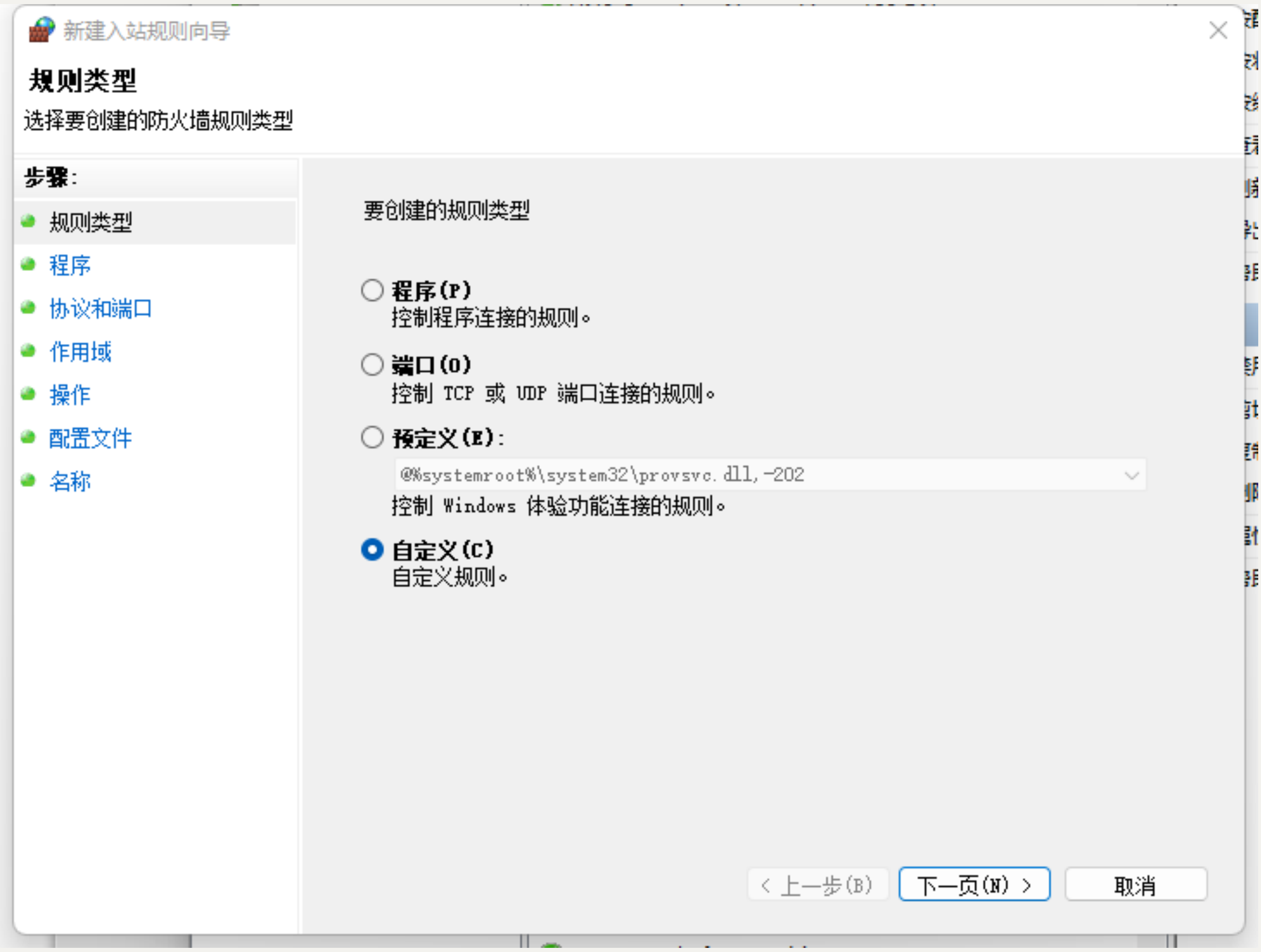
The number following `#` represents the index of hop and the IP address following it is the router's IP and the last one is the destination's IP. Please notice that the IP is **not required** in this assignment, it is shown just for illustrating the `tracert` process. The meaning of the corresponding field is the same as `ping`.

🐔ps. If you are testing `tracert.py` separately under **windows**, if you encounter information that can only display **the last hop**, it may be caused by the firewall rules in windows, please follow the steps below to set it up separately.

1. Create a new inbound rule in the advanced firewall settings


高级安全 Windows Defender 防火墙
文件(F)
操作(A)
查看(V)
帮助(H)
本地计算机 上的高级安全 Win
入站规则
出站规则
连接安全规则
监视
入站规则
名称
组
HNS Container Networking - ICS DN...
HNS Container Networking - ICS DN...
HOI4
HOI4
icmp
icmpv6
IntelliJ IDEA
IntelliJ IDEA
IntelliJ IDEA 2021.2.1
IntelliJ IDEA 2021.2.1
IntelliJ IDEA 2021.2.2
IntelliJ IDEA 2021.2.2
IQIYI Video Helper
IQIYI Video Helper
IQIYI Video Helper
iqiyisetup_z27.exe
iqiyisetup_z27.exe
iTunes.MSI
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
Java(TM) Platform SE binary
操作
入站规则
新建规则...
按配置文件筛选
按状态筛选
按组筛选
查看
刷新
导出列表...
帮助
icmp
禁用规则
剪切
复制
删除
属性
帮助

2. Next select "Custom"


新建入站规则向导
步骤:
规则类型
程序
协议和端口
作用域
操作
配置文件
名称
< 上一步(B)
下一页(N) >
取消

规则类型

选择要创建的防火墙规则类型

要创建的规则类型

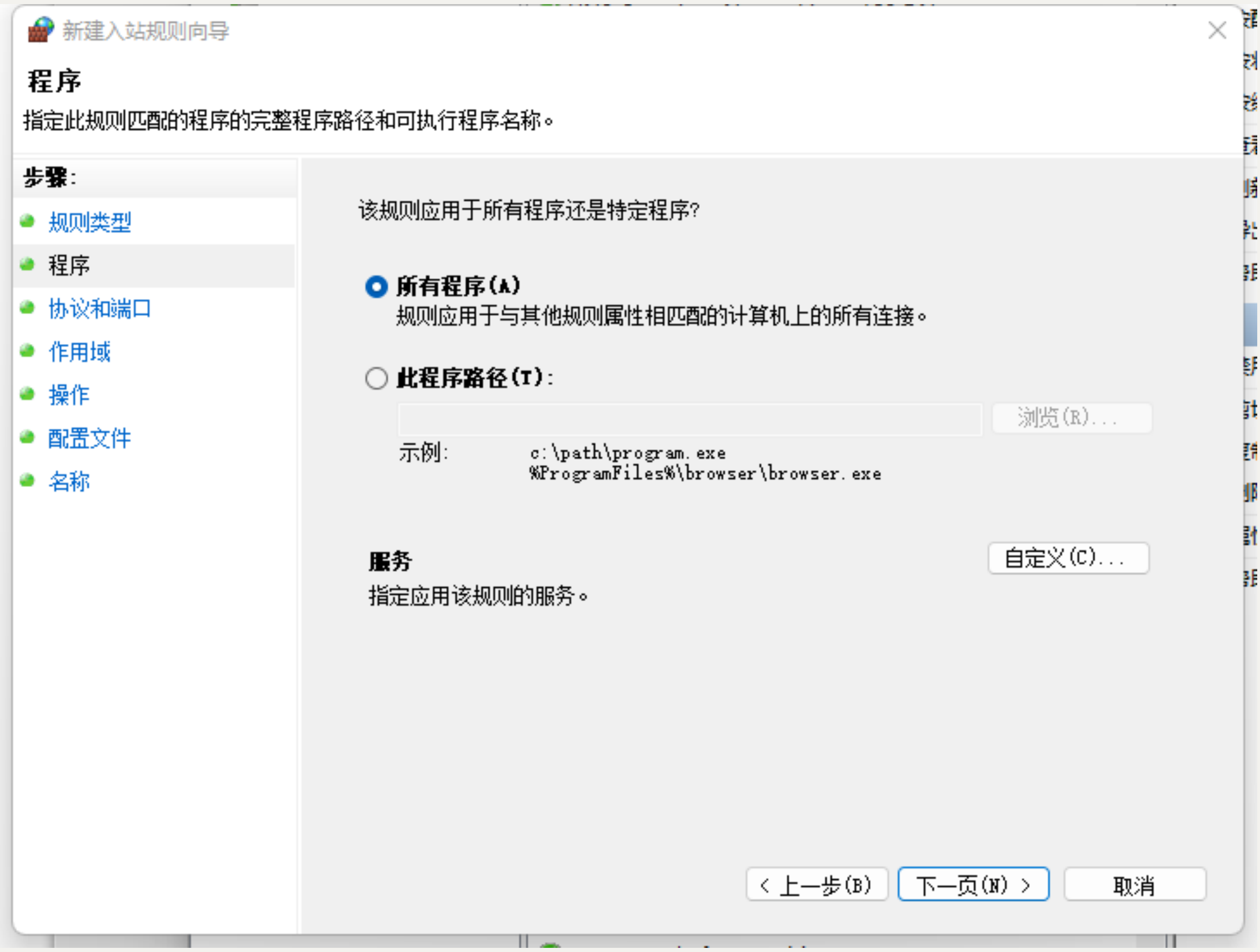
○ **程序(P)**
控制程序连接的规则。

○ **端口(O)**
控制 TCP 或 UDP 端口连接的规则。

○ **预定义(E):**
@%systemroot%\system32\provsvc.dll,-202
控制 Windows 体验功能连接的规则。

● **自定义(C)**
自定义规则。

3. select "all the programs"


新建入站规则向导
步骤:
规则类型
程序
协议和端口
作用域
操作
配置文件
名称
浏览(R)...
自定义(C)...
< 上一步(B)
下一页(N) >
取消

程序

指定此规则匹配的程序的完整程序路径和可执行程序名称。

该规则应用于所有程序还是特定程序?

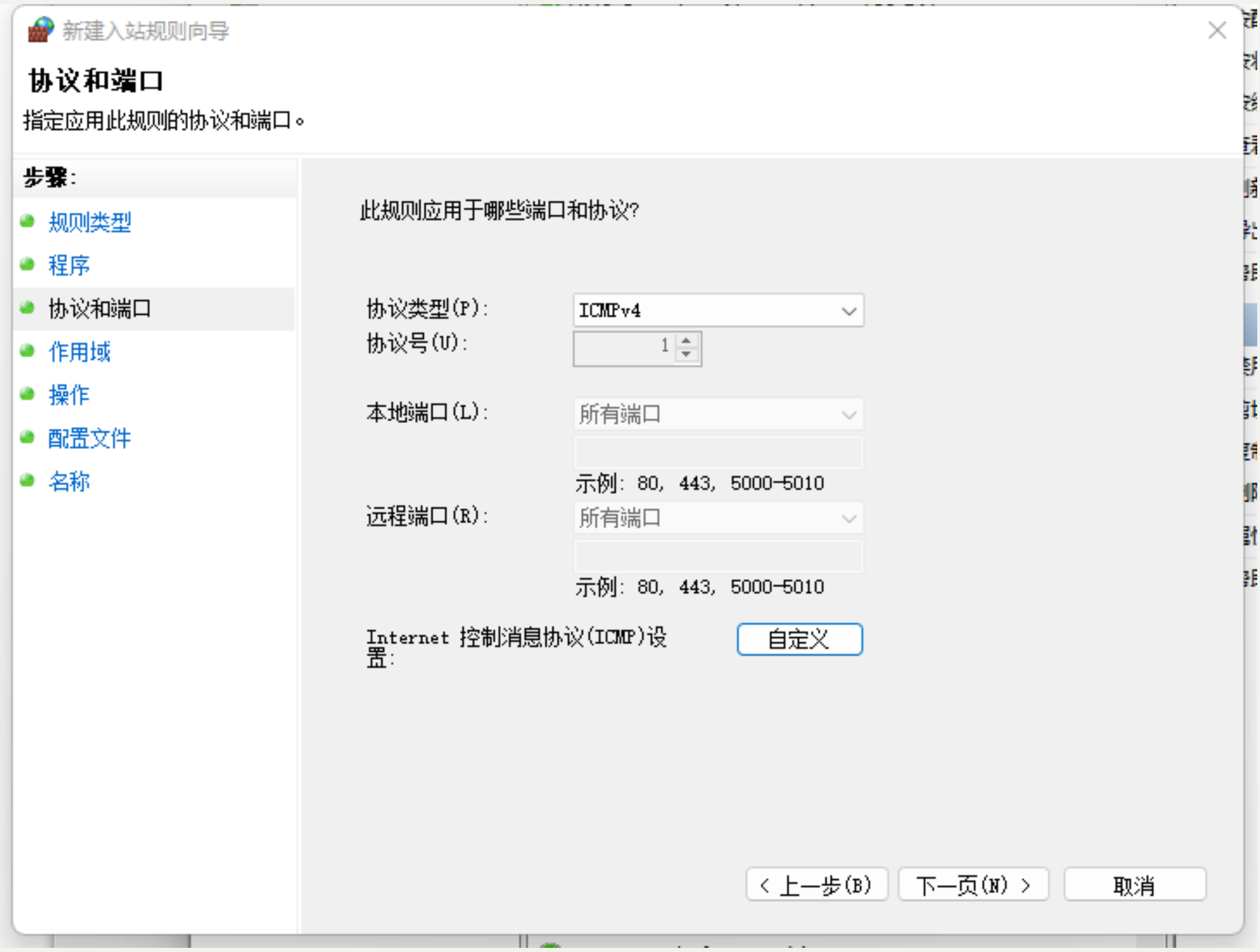
● 所有程序(A)
规则应用于与其他规则属性相匹配的计算机上的所有连接。

○ 此程序路径(T):

示例:
```
c:\path\program.exe
%ProgramFiles%\browser\browser.exe
```

服务
指定应用该规则的服务。

4. select the "icmpv4" protocol


新建入站规则向导
步骤:
规则类型
程序
协议和端口
作用域
操作
配置文件
名称
自定义
< 上一步(B)
下一页(N) >
取消

协议和端口

指定应用此规则的协议和端口。

此规则应用于哪些端口和协议?

协议类型(P): ICMPv4

协议号(U): 1

本地端口(L): 所有端口

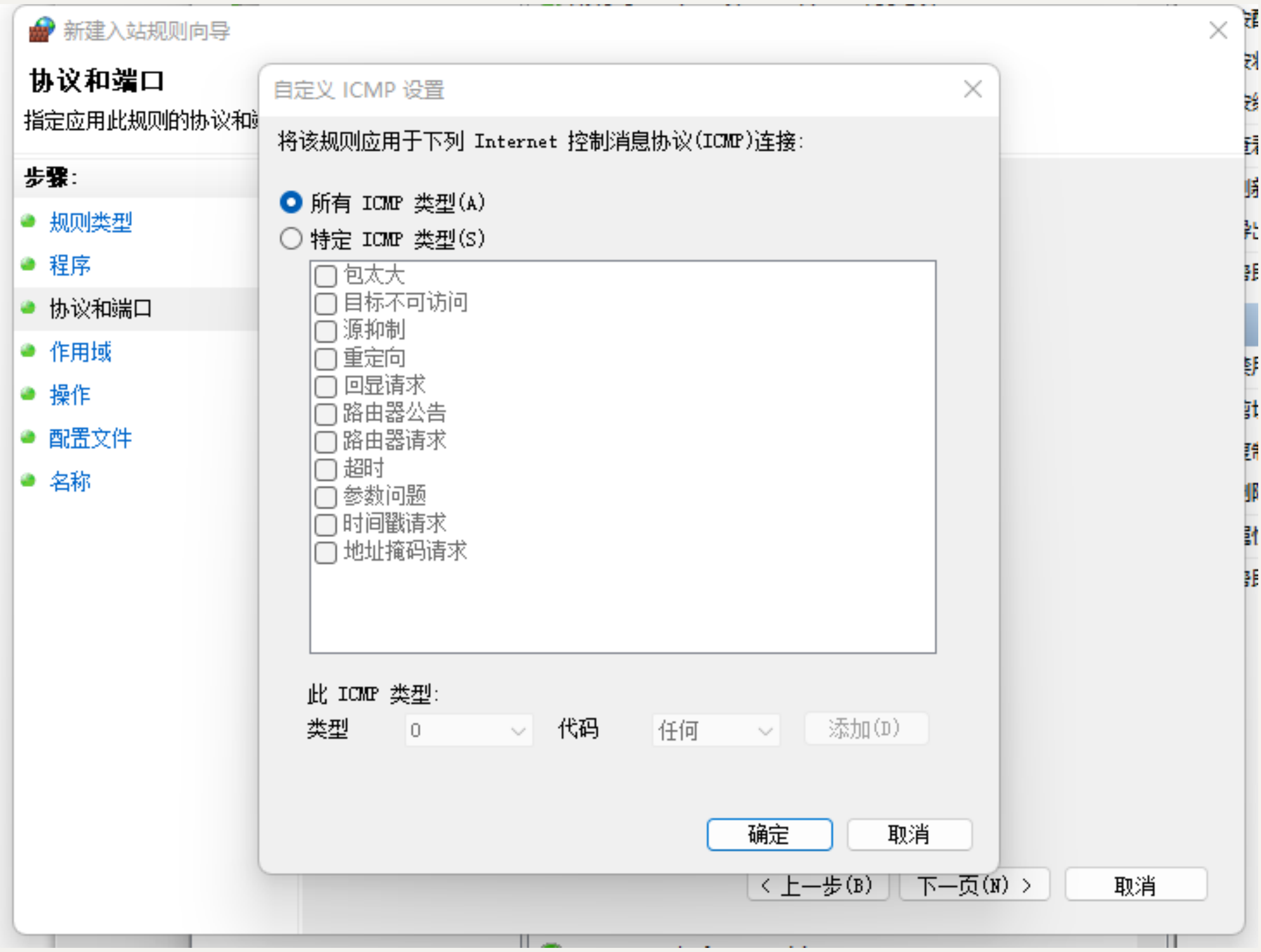
示例: 80, 443, 5000-5010

远程端口(R): 所有端口

示例: 80, 443, 5000-5010

Internet 控制消息协议(ICMP)设置:

5. Accept all the ICMP message types.



新建入站规则向导

协议和端口

指定应用此规则的协议和端

自定义 ICMP 设置

将该规则应用于下列 Internet 控制消息协议(ICMP)连接:

● 所有 ICMP 类型(A)

○ 特定 ICMP 类型(S)

□ 包太大
□ 目标不可访问
□ 源抑制
□ 重定向
□ 回显请求
□ 路由器公告
□ 路由器请求
□ 超时
□ 参数问题
□ 时间戳请求
□ 地址掩码请求

此 ICMP 类型:

类型 0 代码 任何

How to run the code

Pycharm

Move to the root directory of the assignment, and directly click the `run` button is OK.

VScode

1. Move to the root directory of the assignment
2. Run `python -m unittest tests.ping_test` to execute the `ping_test.py`, and the execution of `tracert.py` is similar.

Score Environment

Python 3.9

What to submit

You must provide **a zip file** of your implementation, which contains all the code you write, which includes `ping.py`, `sockets.py`, `tracert.py`, etc, and other files if needed.

And, the zip file should be named "SID1_SID2_SID3_labA2.zip"**, for example, "11810000_11910000_12010000_labA2.zip".

No need for the report, just show us your code is okay.

Q&A Link

If you have any questions about this assignment, please go to the link below to raise your question. And we will check and reply in time.
https://github.com/XJC-git/CS305-LabAssignment2

Pay attention that the deadline for this assignment is ⚠️**26, Dec 2022**

Enjoy yourselves 🏀