《网络连通测试程序》说明文档

姓名：杨天诺
学号：2120220603
时间：2022 年 11 月 12 日

目录

摘要

本程序是一个 Windows 平台上提供图形用户界面的网络连通程序。本程序的基本功能是发送回送 ICMP 请求，接收回送应答，并显示到目的结点的连通性。

用户界面上，支持自定义服务端地址、报文数量、时间间隔和报文字节大小。可显示的信息包括每一条请求的发送回送信息，即生存时间（Time To Live, TTL）和往返时间（Round Trip Time, RTT）；以及连通测试的整体统计信息，包括应答数量、包丢失率和延时情况。程序的基本界面如图 1。本程序的源代码和可执行文件公开在 https://github.com/tiannuo-yang/ping-my-network。

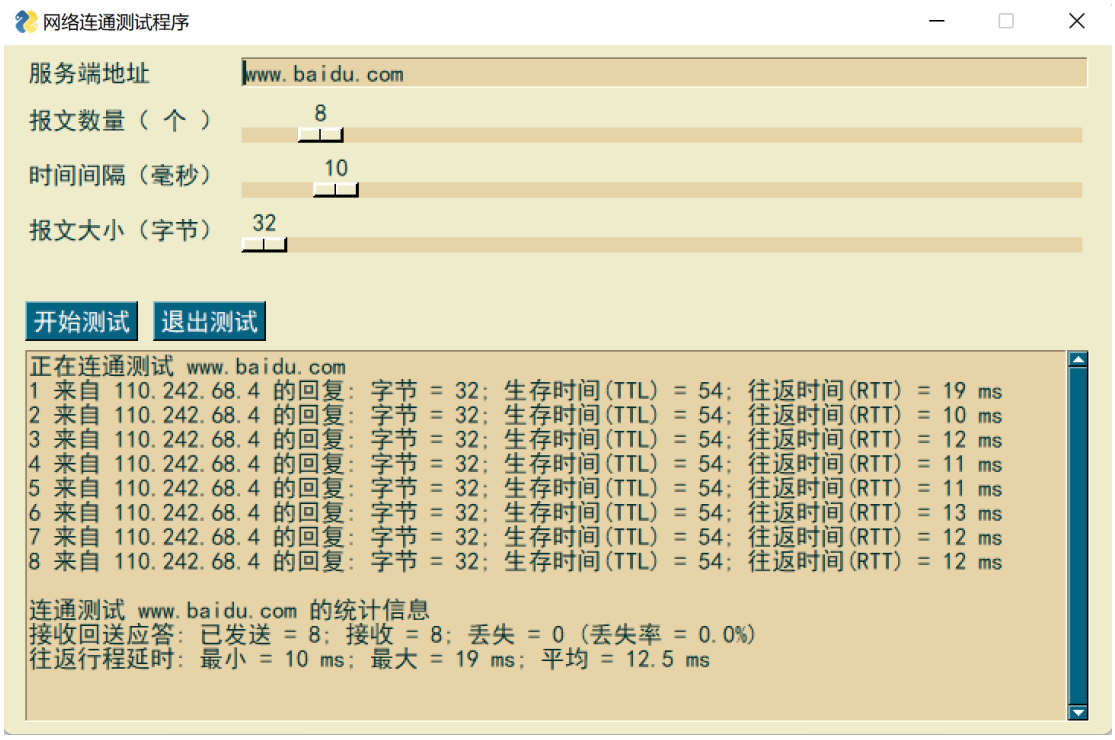

图 1. 网络连通测试程序的基本界面

1 编程环境

1.1 编程语言

本程序使用 Python 3.8.8 进行编写和测试。

1.2 依赖库

本程序调用库见表 1.

表 1. 程序依赖的 Python 库

socket	帮助在网络上的两个程序之间建立信息通道。本程序利用其实现 ICMP 报文的收发。
struct	执行 Python 值和以 Python 字节对象表示的 C 结构之间的转换。本程序利用其实现 ICMP 报文的打包。
time	提供各种与时间相关的函数。本程序利用其实现收发报文的时间统计。
array	定义了一种对象类型，可以紧凑地表示基本类型值的数组。本程序利用其帮助校验和的实现。
random	实现了各种分布的伪随机数生成器。本程序利用其实现给定字节大小下，报文内容的随机生成。
string	处理常见的字符操作。本程序利用其实现报文内容的 ascii 字符生成。
Thread	提供线程方面的操作，帮助并行编程。本程序利用其实现异步执行收发操作、多请求控制和可视化窗口控制。
math	提供算术运算方面的操作。本程序利用其帮助统计延时信息。
PySimpleGUI	Python 用于制作用户交互界面的库。这是本程序唯一使用的三方库，帮助实现可视化界面的生成和控制。

2 关键问题

2.1 使用什么网络协议？

由于网络连通测试程序需要获取每条请求的“网络通不通”、“主机是否可达”、“路由是否可用”等网络本身的消息，所以本程序基于 ICMP 协议实现。实际上，Windows 和 Linux 操作系统上的 Ping 命令也是基于 ICMP 协议实现的。

ICMP 全称为网络控制报文协议（Internet Control Message Protocol）。当用户执行本程序时，本程序会发送一份 ICMP 回显请求报文给目标结点，并等待目标主机返回 ICMP 回显应答。ICMP 协议会要求目标主机在收到消息之后，必须返回 ICMP 应答消息给源主机。由此，本程序可以实现连通测试的目的。

ICMP 报文的结构如表 2。本程序使用的是 Type 8，即回送请求。在创建 header 信息后，按照用户指定的报文大小生成随机的数据，将二者一起进行校验和的计算。最终将所有信息全部组装，得到发送的 ICMP 报文。生成代码见代码 1。

表 2. ICMP 报文结构

Type	请求的类型，包括 0、3、4、5、8、9、10、11、17 和 18.
Code	占 1 字节，标识对应 ICMP 报文的代码。它与类型字段一起共同标识了 ICMP 报文的详细类型。
Checksum	对包括报文数据部分在内的整个 ICMP 数据报的校验和。
Identifier	进程号，回送响应消息与回送消息中 identifier 保持一致。
Sequence Number	序列号，由主机设定，一般设为由 0 递增的序列，回送响应消息与回送消息中 Sequence Number 保持。

代码 1. ICMP 报文生成

```
def create_packet(self):
    header = struct.pack('bbHHh', 8, 0, 0, self.__id, 0)
    packet = header + self.__data
    chkSum = self.inCksum(packet)
    header = struct.pack('bbHHh', 8, 0, chkSum, self.__id, 0)
    return header + self.__data
```

2.2 如何控制请求的发送和接收？

在网络连通测试程序中，最关键的一环就是按照指定的时间发送并接收请求。然而，按照传统的同步编程无法实现同时模拟请求的发送和接收。

因此本程序使用 Python Thread 实现多线程异步编程。**首先**按照用户输入创建一个 ICMP 报文；**其次**将 ICMP 报文发送给指定服务端；**同时**，不等待“发送”步骤结束，直接开启另一个线程用以接收服务端返回的回送请求。具体步骤可见 3.2 节。

2.3 如何判断接收回送应答情况？

网络联通测试程序需要告知用户，当前本机和对方网络结点是否连通。该判断需要人为设定一个时间阈值，即是否在 RTT 限制内接收到应答。

本程序通过以下两点判断每个 ICMP 请求是否得到应答：1）是否在给定超时时间（**Timeout**）内接收到回送请求；2）接收到的回送请求是否与发送的请求中 ID 对应无误（即表 2 中的 **Identifier**）。注意到 **Timeout** 并非一成不变，本程序的 IcmpRequest 类提供了对应的接口，开发者可以更换该阈值。

3 程序流程

3.1 顶层设计

程序的顶层设计包含三个部分：“单个请求处理”、“批量请求控制”，以及“图形界面和输出控制”。它们的关系见图 2。这三个子流程的详细流程在 3.2、3.3 和 3.4 节中。简而言之，程序首先为用户展示一个图形界面，然后不断捕获用户的输入，一旦用户点击“开始测试”，程序就调用“批量请求控制”，然后立即利用多线程构建多个请求的发送和收回。最终，所有信息被展示在图形界面上。

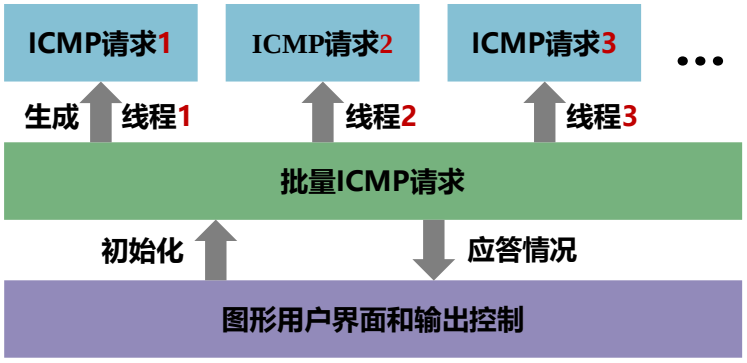

图 2. 程序基本结构

3.2 流程 1：单个请求处理

流程 1 见图 3 左侧。
输入：服务端地址、报文大小和超时时间。
处理：创建 ICMP 请求、发送 ICMP 请求和接收 ICMP 请求。
输出：是否成功收到回复、接收包的地址、请求的 TTL 和请求的 RTT。

3.3 流程 2：批量请求控制

流程 2 见图 3 中间。
输入：服务端地址、报文数量、时间间隔、报文大小和超时时间。
处理：创建批量 ICMP 请求的多线程。
输出：包含所有 ICMP 请求输出的列表。

3.4 流程 3：图形用户界面和输出控制

流程 2 见图 3 右侧。
输入：服务端地址、报文数量、时间间隔和报文大小。
处理：开启图形用户界面和处理批量 ICMP 请求的结果信息。
输出：接收回送应答和目的结点的连通性信息。

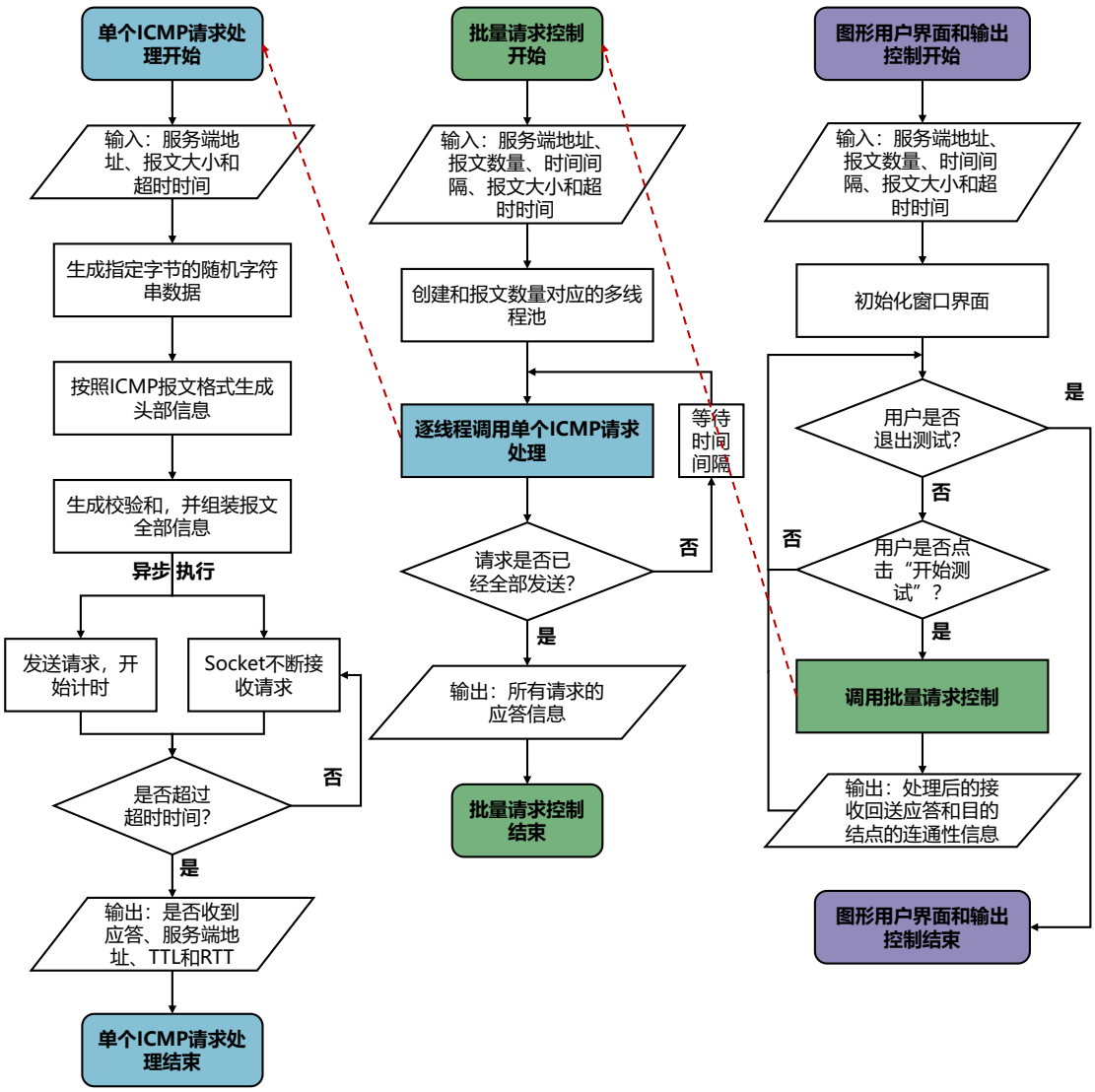

图 3. 程序的三个子流程

4 测试

依据以上程序流程，构建了 Windows 桌面程序，其桌面图标如图 4 所示。下面将分别对其进行基本测试、用户自定义测试和异常测试。本程序测试环境为：Windows 11 家庭中文版（21H2，64 位）的操作系统、AMD Ryzen 7 5800H、3.2GHz 频率的处理器和 16.0 GB 的内存。

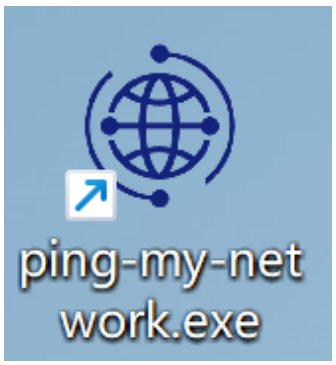

图 4. 网络连通测试程序的桌面图标

4.1 基本测试

基本测试中，程序使用窗口默认的用户输入进行网络连通测试。结果见图 5。可以发现，当给定服务端地址为 www.baidu.com；4 个报文；10 毫秒的时间间隔和 32 字节的报文大小时，程序如期执行了 4 个请求的连通测试。结果表示，丢包率为 0；往返延时平均约为 5.25 毫秒。

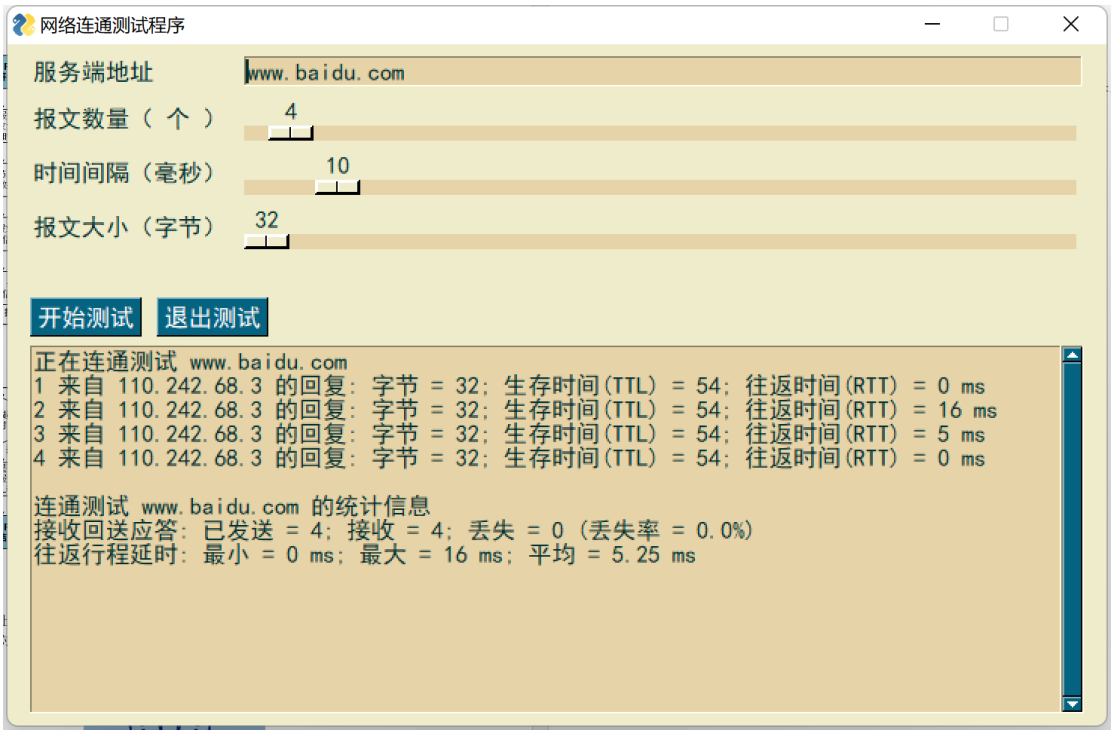

图 5. 程序的基本测试

4.2 用户自定义输入测试

用户自定义输入包括服务端地址、报文数量、时间间隔和报文大小。本节分别通过程序暴露的自定义输入端改变这四个值。结果见图 6、图 7、图 8 和图 9。

可以发现：1）在不同的服务端地址下（前提是正确且可达），本程序总能获取网络连通信息；2）增加报文数量时，本程序仍然能输出所有请求的应答和连通情况，一般而言，请求数量越多，测试结果越准确；3）将时间间隔增大为 100 毫秒时，程序仍能按照指定内容反馈信息；4）当增大报文大小时，往返行程延时可能会有小幅度波动。

4.3 异常测试

本节对程序可能产生的异常进行测试，以观察其是否在设计预期内。**注意到此处的异常指的不是程序错误，而是用户的不合理输入。**结果见图 10、图 11。

可以发现：1）当用户输入本机地址 127.0.0.1 时，往返时间几乎为 0，这是符合预期的，因为本机的网络通信时间可以忽略不计；2）当用户输入错误的 IP 地址时，程序给出的结果是所有请求均超时，即在 1.0 秒内无应答，与预期一致。

值得注意的是，本程序已经通过将部分自定义输入框改为“滑动选择”模块避免了部分不合理输入。这样做尽管可能会损失一点用户自主权（输入的上下界已被指定无法更改），但是能够大大提高用户的使用便捷性——拖一拖图标就能运行，且不容易输入非法值。

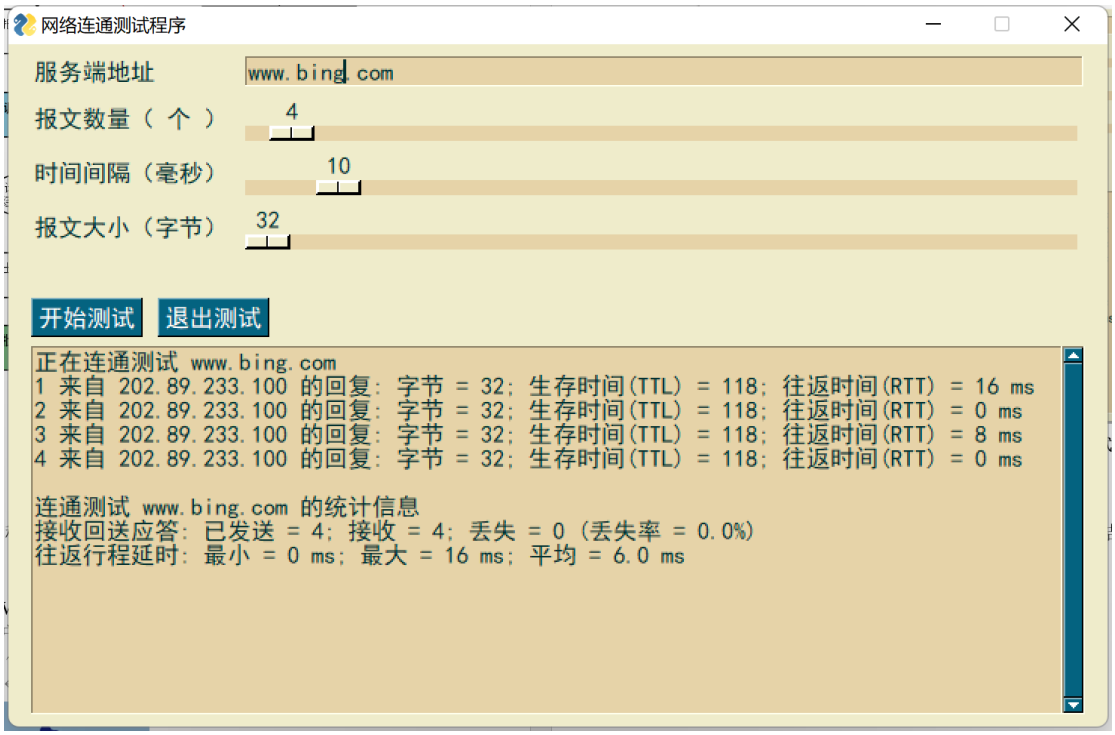

图 6. 改变服务端地址的测试

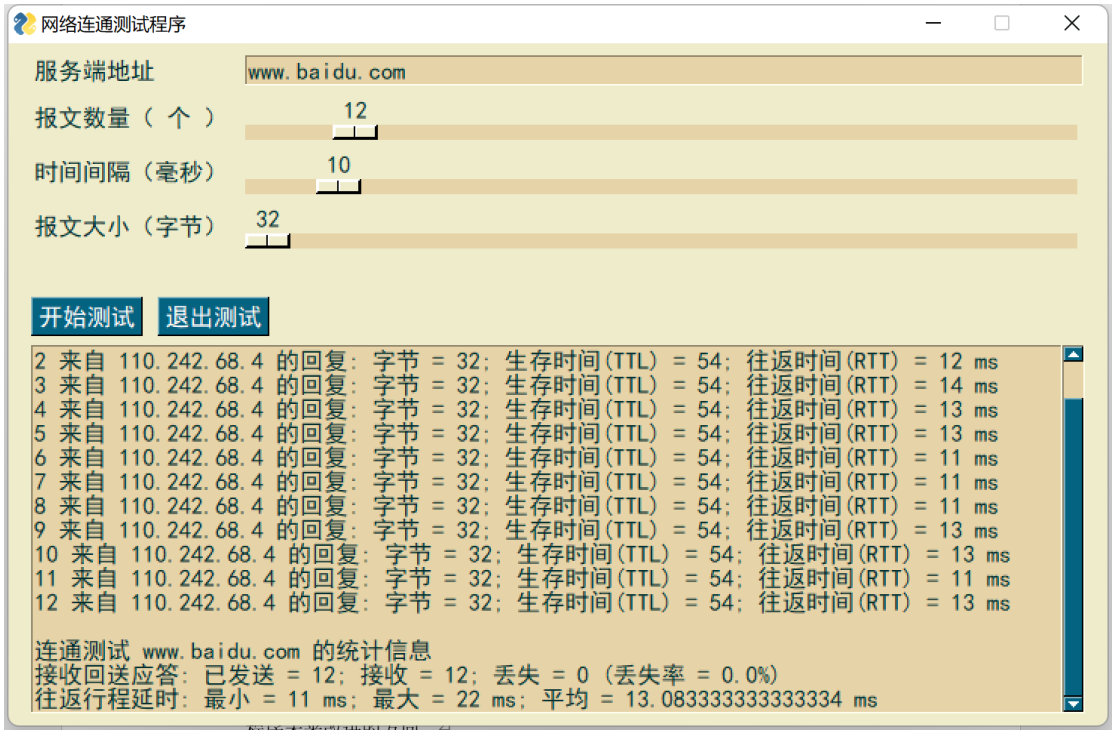

图 7. 改变报文数量的测试

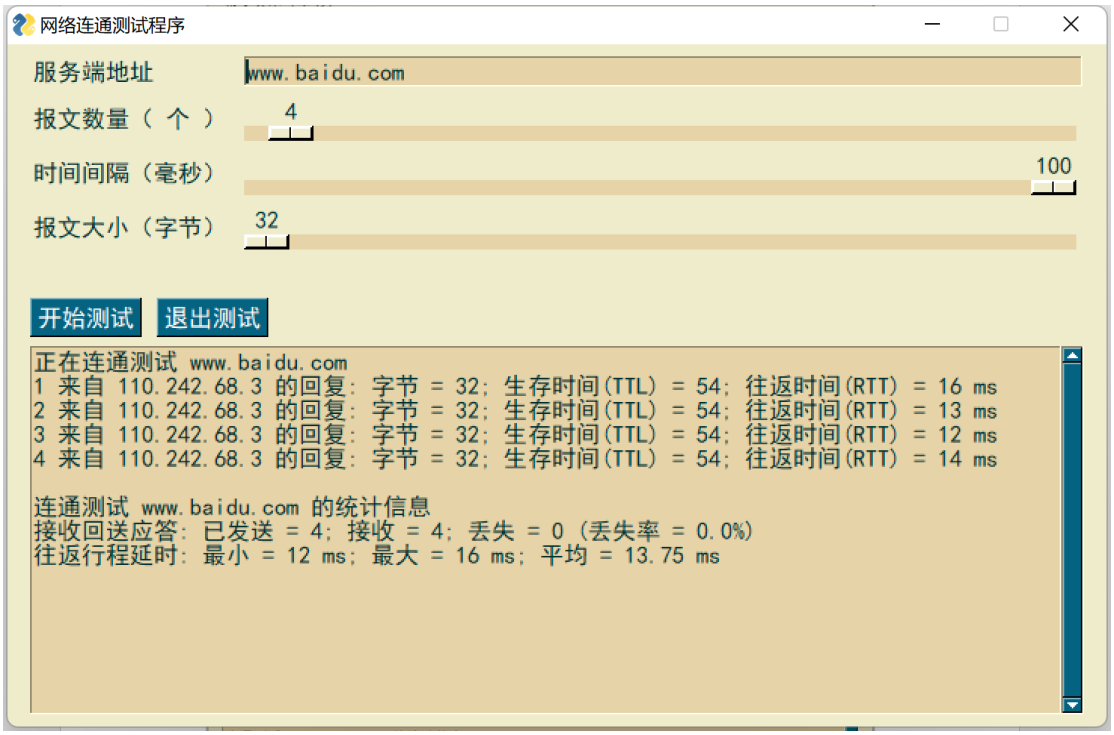

图 8. 改变时间间隔的测试

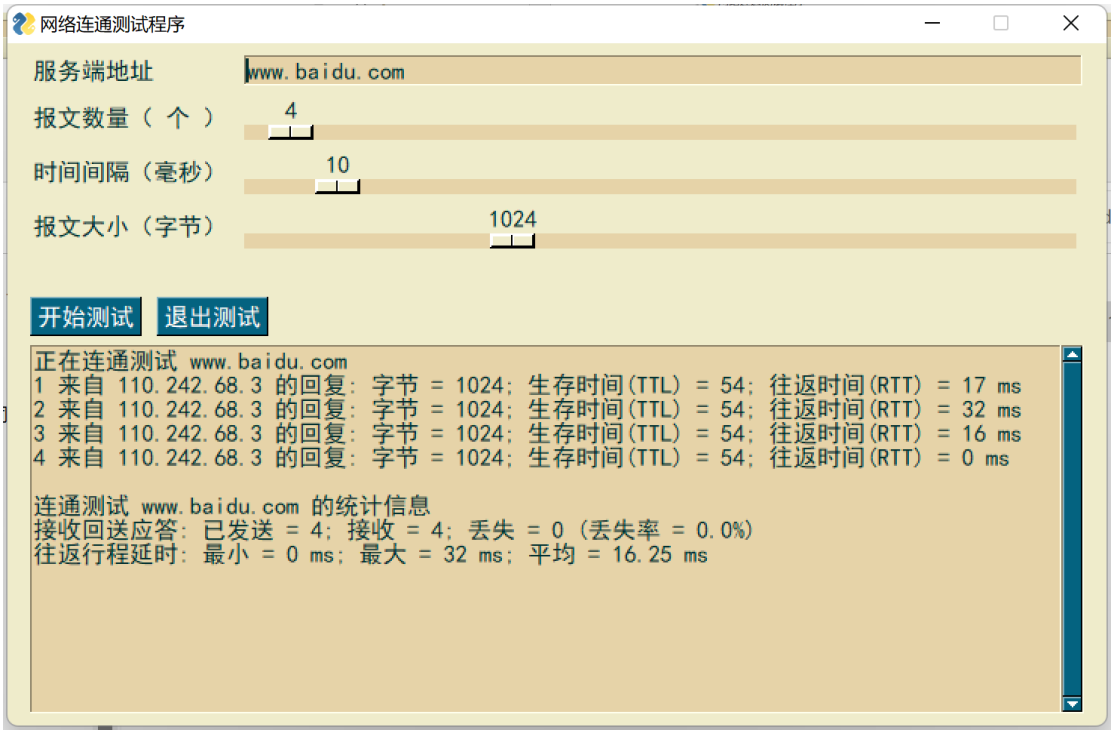

图 9. 改变报文大小的测试

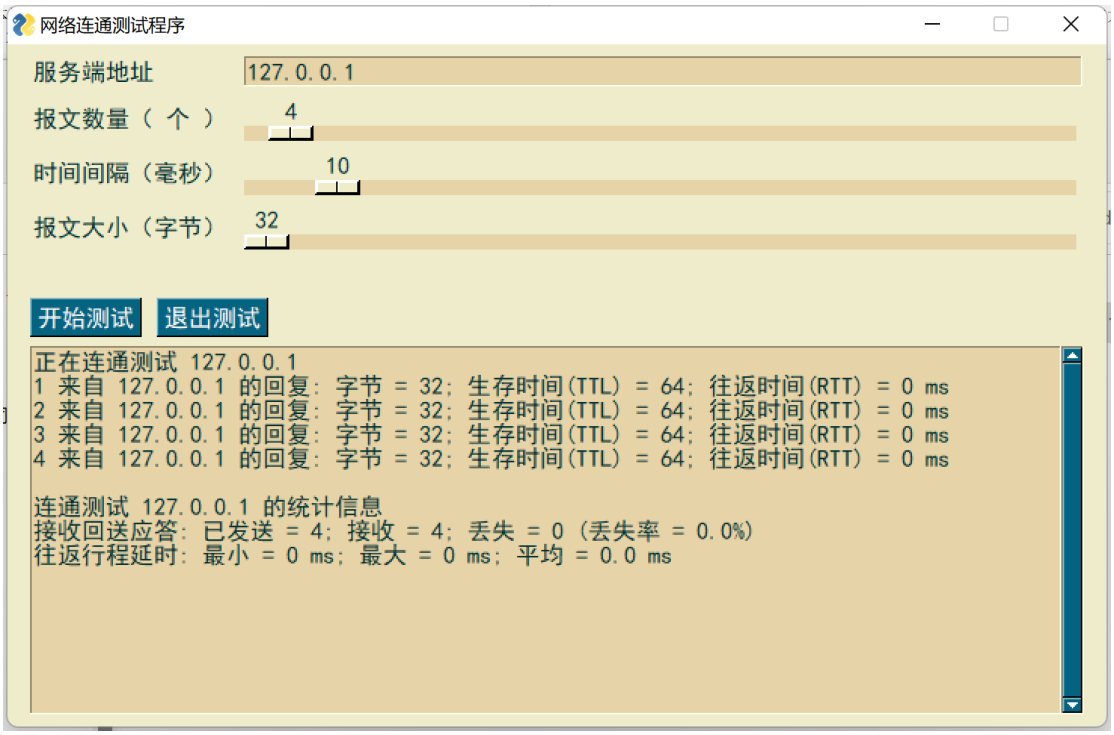

图 10. 用户输入本机地址的连通测试结果

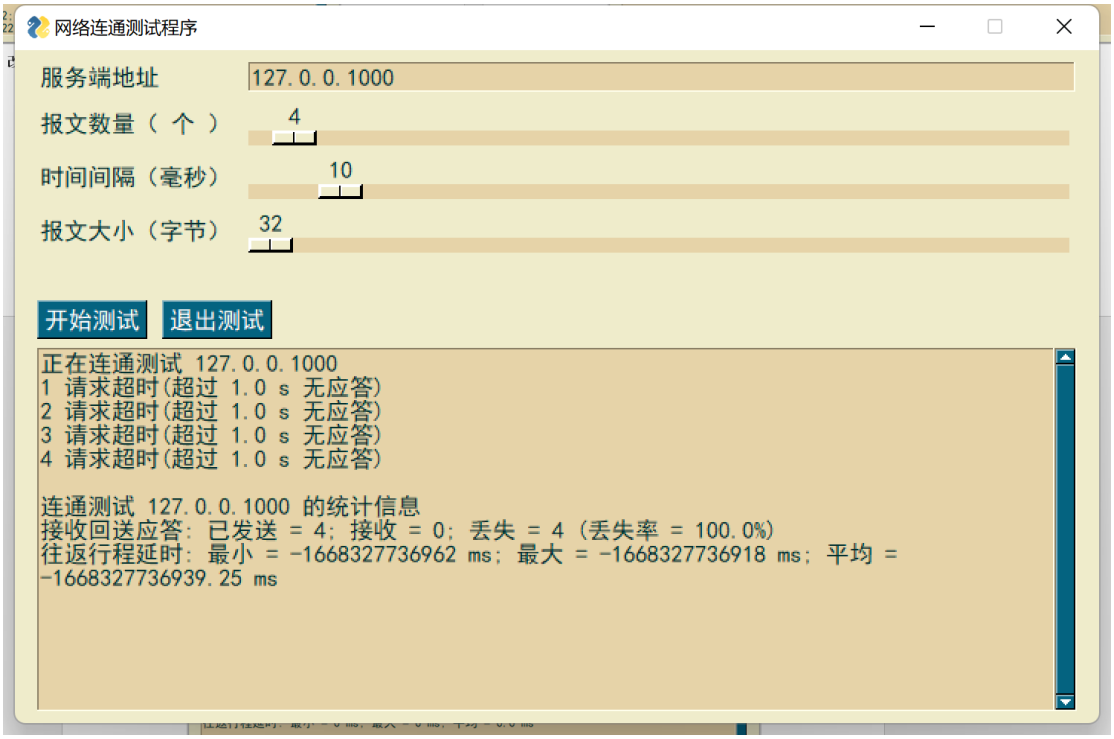

图 11. 用户输入错误服务端地址的连通测试结果

5 总结

通过本程序，用户可以实现简单、快捷的网络连通测试。笔者从中不仅学习到了 ICMP 协议和报文的实现细节，并且对用户界面编程有了更深刻的认识。

然而，相比于 Windows 和 Linux 平台自带的、较为成熟的 Ping 命令，本程序还有可提升空间，例如，本程序无法强制使用 IPv4 或 IPv6 协议。此外，本程序未来还可以进一步添加服务端地址合法性的本地校验，而不是将此校验交由 socket，从而能够区分“超时无应答”和“地址非法”分别造成的“请求超时”。