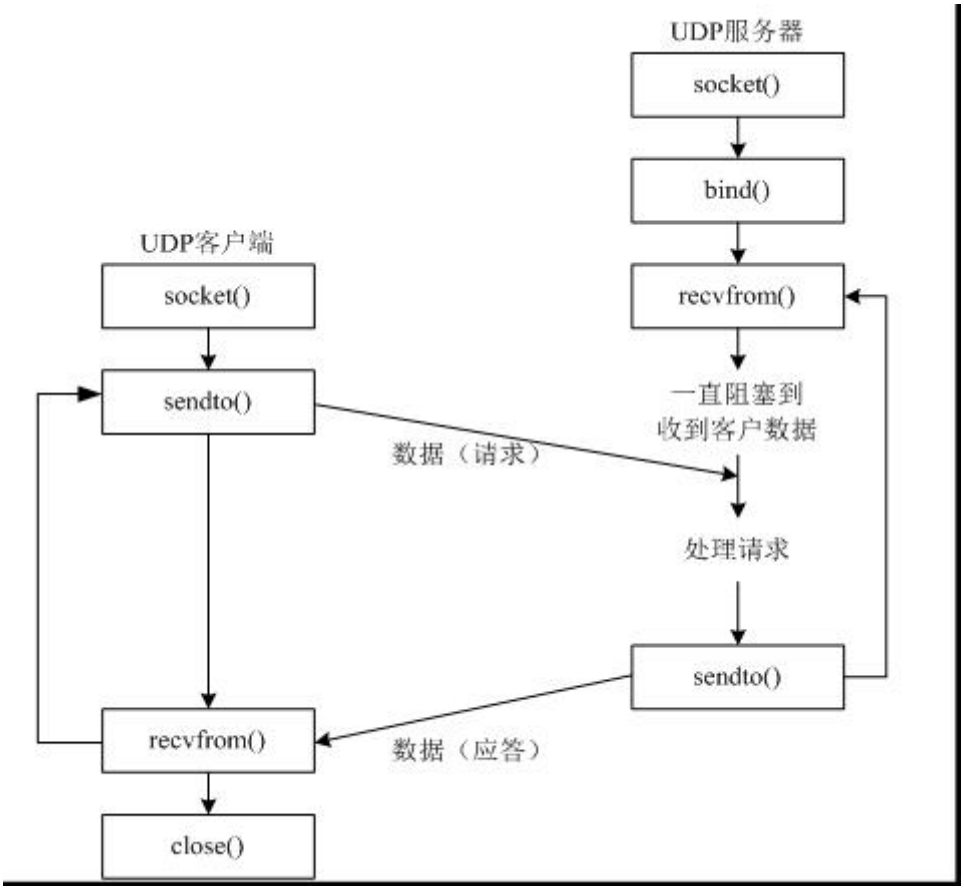
udp网络程序-发送、接收数据

> UDP是一种无连接的网络协议
>
> 1， 快速简单
>
> 2. 不保证交付
> 3. 无连接
> 4. 可能会出现乱序到达
>
> 非常适合对速度要求高，但可以容忍一定丢包的应用(视频，)

1. udp网络程序-发送数据

创建一个基于udp的网络程序流程很简单，具体步骤如下：

1. 创建客户端套接字
2. 发送/接收数据
3. 关闭套接字



代码如下：

```
# coding=utf-8

from socket import *

# 1. 创建udp套接字
udp_socket = socket(AF_INET, SOCK_DGRAM)

# 2. 准备接收方的地址
# '192.168.1.103'表示目的ip地址
# 8080表示目的端口
dest_addr = ('192.168.1.103', 8080)  # 注意是元组，ip是字符串，端口是数字

# 3. 从键盘获取数据
send_data = input("请输入要发送的数据:")
```

```
# 4. 发送数据到指定的电脑上的指定程序中
udp_socket.sendto(send_data.encode('utf-8'), dest_addr)

# 5. 关闭套接字
udp_socket.close()
```

运行现象：

在Ubuntu中运行脚本：

```
python@ubuntu:~/Desktop$ python3 01.py
请输入要发送的数据:nihao
```

在windows中运行“网络调试助手”：

2. udp网络程序-发、接收数据

```
#coding=utf-8

from socket import *

# 1. 创建udp套接字
udp_socket = socket(AF_INET, SOCK_DGRAM)

# 2. 准备接收方的地址
```

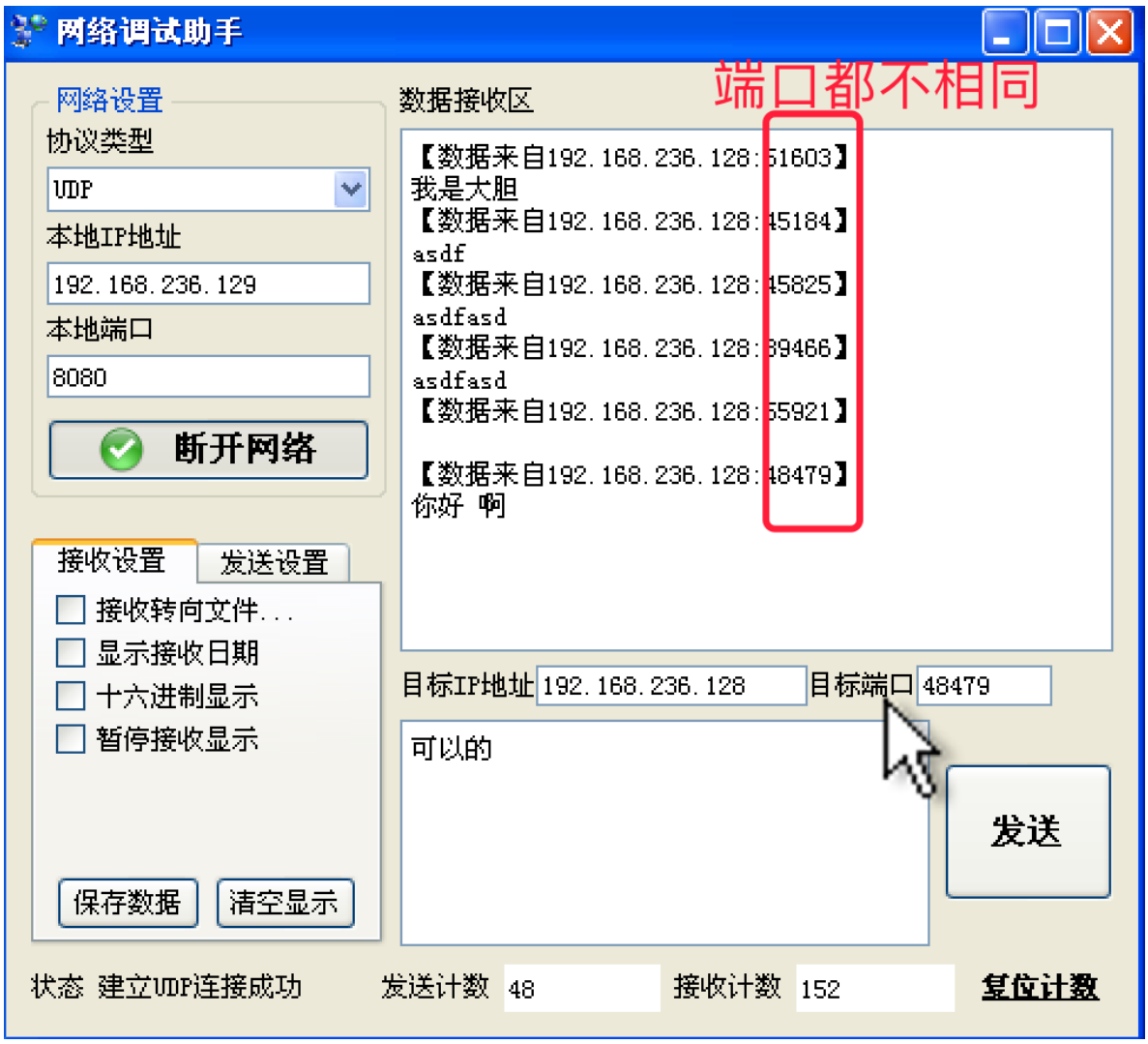
```
dest_addr = ('192.168.236.129', 8080)

# 3. 从键盘获取数据
send_data = input("请输入要发送的数据:")

# 4. 发送数据到指定的电脑上
udp_socket.sendto(send_data.encode('utf-
8'), dest_addr)

# 5. 等待接收对方发送的数据
recv_data = udp_socket.recvfrom(1024)  #
1024表示本次接收的最大字节数

# 6. 显示对方发送的数据
# 接收到的数据recv_data是一个元组
# 第1个元素是对方发送的数据
# 第2个元素是对方的ip和端口
print(recv_data[0].decode('gbk'))
print(recv_data[1])

# 7. 关闭套接字
udp_socket.close()
```

python脚本:

```
python@ubuntu:~/Desktop$ python3 02.py
请输入要发送的数据:你好 啊
可以的
('192.168.236.129', 8080)
python@ubuntu:~/Desktop$
```

网络调试助手截图:

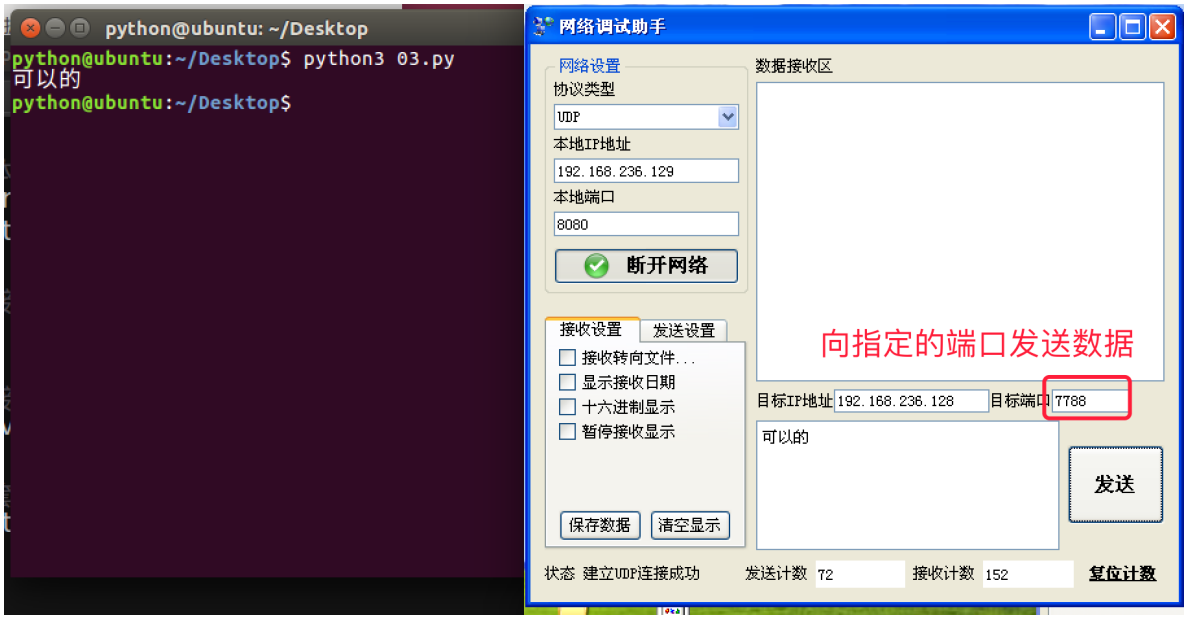
3.udp网络程序-绑定端口发送消息



绑定示例

```
#coding=utf-8

from socket import *

# 1. 创建套接字
udp_socket = socket(AF_INET, SOCK_DGRAM)

# 2. 绑定本地的相关信息，如果一个网络程序不绑定，则系统会随机分配
```

```python
local_addr = ('', 7788) #  ip地址和端口号，ip一般不用写，表示本机的任何一个ip
udp_socket.bind(local_addr)

# 3. 等待接收对方发送的数据
recv_data = udp_socket.recvfrom(1024) # 1024表示本次接收的最大字节数

# 4. 显示接收到的数据
print(recv_data[0].decode('gbk'))

# 5. 关闭套接字
udp_socket.close()
```



总结：
一个udp网络程序，可以不绑定，此时操作系统会随机进行分配一个端口，如果重新运行此程序端口可能会发生变化

一个udp网络程序，也可以绑定信息（ip地址，端口号），如果绑定成功，那么操作系统用这个端口号来进行区别收到的网络数据是否是此进程的