图2-28着重显示了客户和服务器的主要与套接字相关的活动，两者通过UDP运输服务进行通信。

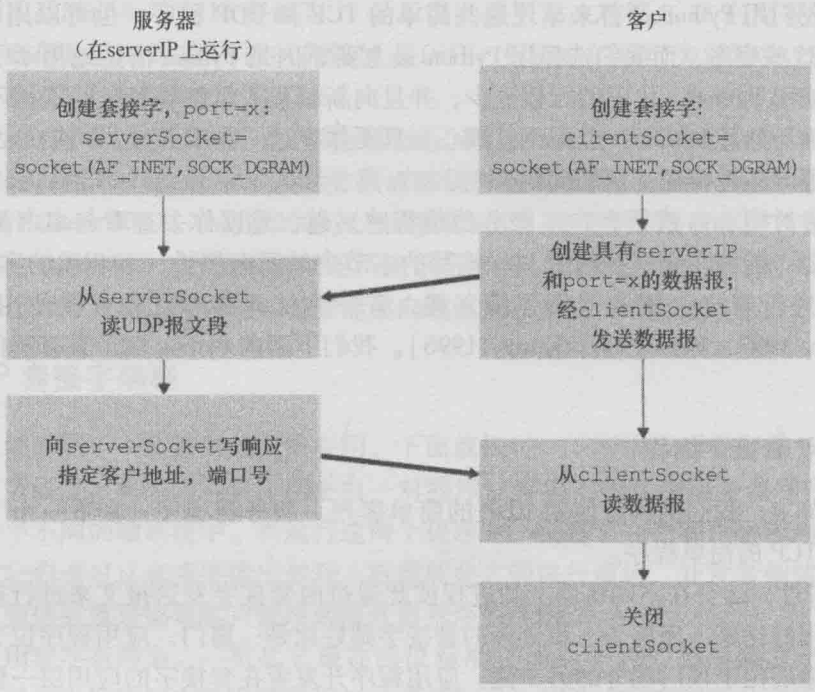

图2-28 使用UDP的客户－服务器应用程序

现在我们自己动手来查看这对客户－服务器程序，用UDP实现这个简单应用程序。我们在每个程序后也提供一个详细、逐行的分析。我们将以UDP客户开始，该程序将向服务器发送一个简单的应用级报文。服务器为了能够接收并回答该客户的报文，它必须准备好并已经在运行，这就是说，在客户发送其报文之前，服务器必须作为一个进程正在运行。

客户程序被称为UDPClient. py，服务器程序被称为UDPServer. py。为了强调关键问题，我们有意提供最少的代码。“好代码”无疑将具有一些更为辅助性的代码行，特别是用于处理出现差错的情况。对于本应用程序，我们任意选择了12000作为服务器的端口号。

1. UDPClient. py

下面是该应用程序客户端的代码：

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

现在我们看在UDPClient. py中的各行代码。

```
from socket import *
```

该socket模块形成了在Python中所有网络通信的基础。包括了这行，我们将能够在程序中创建套接字。