11.3 创建UDP服务器¶

问题¶

你想实现一个基于UDP协议的服务器来与客户端通信。

解决方案¶

跟TCP一样，UDP服务器也可以通过使用 `socketserver` 库很容易的被创建。 例如，下面是一个简单的时间服务器：

```
from socketserver import BaseRequestHandler, UDPServer
import time

class TimeHandler(BaseRequestHandler):
    def handle(self):
        print('Got connection from', self.client_address)
        # Get message and client socket
        msg, sock = self.request
        resp = time.ctime()
        sock.sendto(resp.encode('ascii'), self.client_address)

if __name__ == '__main__':
    serv = UDPServer(('', 20000), TimeHandler)
    serv.serve_forever()
```

跟之前一样，你先定义一个实现 `handle()` 特殊方法的类，为客户端连接服务。 这个类的 `request` 属性是一个包含了数据报和底层socket对象的元组。 `client_address` 包含了客户端地址。

我们来测试下这个服务器，首先运行它，然后打开另外一个Python进程向服务器发送消息：

```
>>> from socket import socket, AF_INET, SOCK_DGRAM
>>> s = socket(AF_INET, SOCK_DGRAM)
>>> s.sendto(b'', ('localhost', 20000))
0
>>> s.recvfrom(8192)
(b'Wed Aug 15 20:35:08 2012', ('127.0.0.1', 20000))
>>>
```

讨论¶

一个典型的UDP服务器接收到达的数据报(消息)和客户端地址。如果服务器需要做应答， 它要给客户端回发一个数据报。对于数据报的传送， 你应该使用socket的 `sendto()` 和 `recvfrom()` 方法。 尽管传统的 `send()` 和 `recv()` 也可以达到同样的效果， 但是前面的两个方法对于UDP连接而言更普遍。

由于没有底层的连接，UPD服务器相对于TCP服务器来讲实现起来更加简单。 不过，UDP天生是不可靠的（因为通信没有建立连接，消息可能丢失）。 因此需要由你自己来决定该怎样处理丢失消息的情况。这个已经不在本书讨论范围内了， 不过通常来说，如果可靠性对于你程序很重要，你需要借助于序列号、重试、超时以及一些其他方法来保证。 UDP通常被用在那些对于可靠传输要求不是很高的场合。例如，在实时应用如多媒体流以及游戏领域， 无需返回恢复丢失的数据包（程序只需简单的忽略它并继续向前运行）。

`UDPServer` 类是单线程的，也就是说一次只能为一个客户端连接服务。 实际使用中，这个无论是对于UDP还是TCP都不

是什么大问题。 如果你想要并发操作，可以实例化一个 `ForkingUDPServer` 或 `ThreadingUDPServer` 对象：

```
from socketserver import ThreadingUDPServer

  if __name__ == '__main__':
   serv = ThreadingUDPServer(('',20000), TimeHandler)
   serv.serve_forever()
```

直接使用 `socket` 来实现一个UDP服务器也不难，下面是一个例子：

```
from socket import socket, AF_INET, SOCK_DGRAM
import time

def time_server(address):
   sock = socket(AF_INET, SOCK_DGRAM)
   sock.bind(address)
   while True:
      msg, addr = sock.recvfrom(8192)
      print('Got message from', addr)
      resp = time.ctime()
      sock.sendto(resp.encode('ascii'), addr)

if __name__ == '__main__':
   time_server(('', 20000))
```