2. 面向连接的多路复用与多路分解

为了理解 TCP 分解，我们必须更为仔细地研究 TCP 套接字和 TCP 连接创建。TCP 套接字和 UDP 套接字之间的一个细微差别是，TCP 套接字是由一个四元组（源 IP 地址，源端口号，目的 IP 地址，目的端口号）来标识的。这样，当一个 TCP 报文段从网络到达一台主机时，该主机使用全部 4 个值来将报文段定向（分解）到相应的套接字。特别与 UDP 不同的是，两个具有不同源 IP 地址或源端口号的到达 TCP 报文段将被定向到两个不同的套接字，除非 TCP 报文段携带了初始创建连接的请求。为了深入地理解这一点，我们再来重新考虑 2.7.2 节中的 TCP 客户 - 服务器编程的例子：

- TCP 服务器应用程序有一个“welcoming socket”，它在 12000 号端口上等待来自 TCP 客户（见图 2-29）的连接建立请求。
- TCP 客户使用下面的代码创建一个套接字并发送一个连接建立请求报文段：

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,12000))
```

- 一条连接建立请求只不过是一个目的端口号为 12000，TCP 首部的特定“连接建立位”置位的 TCP 报文段（在 3.5 节进行讨论）。这个报文段也包含一个由客户选择的源端口号。
- 当运行服务器进程的计算机的主机操作系统接收到具有目的端口 12000 的入连接请求报文段后，它就定位服务器进程，该进程正在端口号 12000 等待接受连接。该服务器进程则创建一个新的套接字：

```
connectionSocket, addr = serverSocket.accept()
```

- 该服务器的运输层还注意到连接请求报文段中的下列 4 个值：①该报文段中的源端口号；②源主机 IP 地址；③该报文段中的目的端口号；④自身的 IP 地址。新创建的连接套接字通过这 4 个值来标识。所有后续到达的报文段，如果它们的源端口号、源主机 IP 地址、目的端口号和目的 IP 地址都与这 4 个值匹配，则被分解到这个套接字。随着 TCP 连接完成，客户和服务器便可相互发送数据了。

服务器主机可以支持很多并行的 TCP 套接字，每个套接字与一个进程相联系，并由其四元组来标识每个套接字。当一个 TCP 报文段到达主机时，所有 4 个字段（源 IP 地址，源端口，目的 IP 地址，目的端口）被用来将报文段定向（分解）到相应的套接字。

关注安全性

端口扫描

我们已经看到一个服务器进程潜在地在一个打开的端口等待远程客户的接触。某些端口为周知应用程序（例如 Web、FTP、DNS 和 SMTP 服务器）所预留；依照惯例其他端口由流行应用程序（例如微软 2000 SQL 服务器在 UDP 1434 端口上监听请求）使用。因此，如果我们确定一台主机上打开了一个端口，也许就能够将该端口映射到在该主机运行的一个特定的应用程序上。这对于系统管理员非常有用，系统管理员通常希望知晓有什么样的网络应用程序正运行在他们的网络主机上。而攻击者为了“寻找突破口”，也要知道在目标主机上有哪些端口打开。如果发现一台主机正在运行具有已知安全缺陷的应用程序（例如，在端口 1434 上监听的一台 SQL 服务器会遭受缓存溢出，使得一个