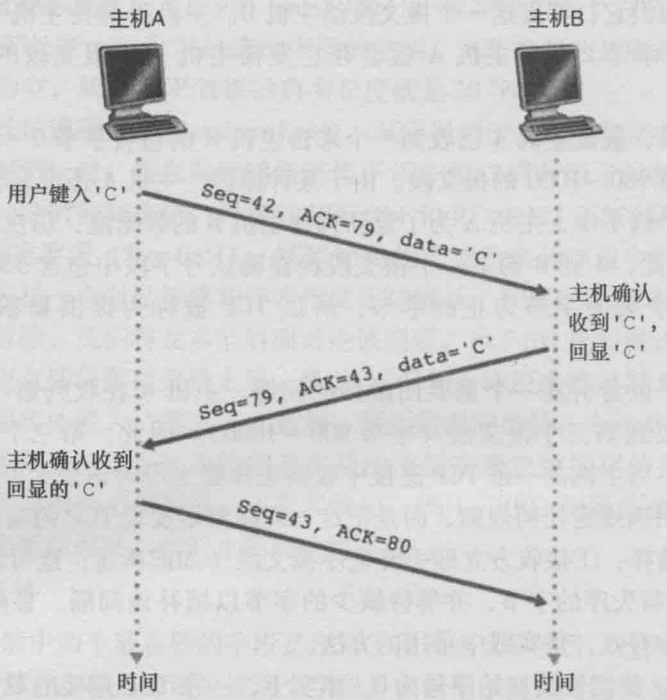

图 3-31 一个经 TCP 的简单 Telnet 应用的确认号和序号

如图 3-31 中所示，共发送 3 个报文段。第一个报文段是由客户发往服务器，在它的数据字段里包含一字节的字符‘C’的 ASCII 码。如我们刚讲到的那样，第一个报文段的序号字段里是 42。另外，由于客户还没有接收到来自服务器的任何数据，因此该第一个报文段中的确认号字段中是 79。

第二个报文段是由服务器发往客户。它有两个目的：首先它是为该服务器所收到数据提供一个确认。通过在确认号字段中填入 43，服务器告诉客户它已经成功地收到字节 42 及以前的所有字节，现在正等待着字节 43 的出现。该报文段的第二个目的是回显字符‘C’。因此，在第二个报文段的数据字段里填入的是字符‘C’的 ASCII 码。第二个报文段的序号为 79，它是该 TCP 连接上从服务器到客户的数据流的起始序号，这也正是服务器要发送的第一个字节的数据。值得注意的是，对客户到服务器的数据的确认被装载在一个承载服务器到客户的数据的报文段中；这种确认被称为是被**捎带**（piggybacked）在服务器到客户的数据报文段中的。

第三个报文段是从客户发往服务器的。它的唯一目的是确认已从服务器收到的数据。（前面讲过，第二个报文段中包含的数据是字符‘C’，是从服务器到客户的。）该报文段的数据字段为空（即确认信息没有被任何从客户到服务器的数据所捎带）。该报文段的确认号字段填入的是 80，因为客户已经收到了字节流中序号为 79 及以前的字节，它现在正等待着字节 80 的出现。你可能认为这有点奇怪，即使该报文段里没有数据还仍有序号。这是因为 TCP 存在序号字段，报文段需要填入某个序号。

3.5.3 往返时间的估计与超时

TCP 如同前面 3.4 节所讲的 rdt 协议一样，它采用超时/重传机制来处理报文段的丢失