所有字节，同时假设它打算发送一个报文段给主机 B。主机 A 等待主机 B 的数据流中字节 536 及之后的所有字节。所以主机 A 就会在它发往主机 B 的报文段的确认号字段中填上 536。

再举一个例子，假设主机 A 已收到一个来自主机 B 的包含字节 0 ~ 535 的报文段，以及另一个包含字节 900 ~ 1000 的报文段。由于某种原因，主机 A 还没有收到字节 536 ~ 899 的报文段。在这个例子中，主机 A 为了重新构建主机 B 的数据流，仍在等待字节 536（和其后的字节）。因此，A 到 B 的下一个报文段将在确认号字段中包含 536。因为 TCP 只确认该流中至第一个丢失字节为止的字节，所以 TCP 被称为提供**累积确认**（cumulative acknowledgment）。

最后一个例子也会引发一个重要而微妙的问题。主机 A 在收到第二个报文段（字节 536 ~ 899）之前收到第三个报文段（字节 900 ~ 1000）。因此，第三个报文段失序到达。该微妙的问题是：当主机在一条 TCP 连接中收到失序报文段时该怎么办？有趣的是，TCP RFC 并没有为此明确规定任何规则，而是把这一问题留给实现 TCP 的编程人员去处理。他们有两个基本的选择：①接收方立即丢弃失序报文段（如前所述，这可以简化接收方的设计）；②接收方保留失序的字节，并等待缺少的字节以填补该间隔。显然，后一种选择对网络带宽而言更为有效，是实践中采用的方法。

在图 3-30 中，我们假设初始序号为 0。事实上，一条 TCP 连接的双方均可随机地选择初始序号。这样做可以减少将那些仍在网络中存在的来自两台主机之间先前已终止的连接的报文段，误认为是后来这两台主机之间新建连接所产生的有效报文段的可能性（它碰巧与旧连接使用了相同的端口号）[Sunshine 1978]。

2. Telnet：序号和确认号的一个学习案例

Telnet 由 RFC 854 定义，它现在是一个用于远程登录的流行应用层协议。它运行在 TCP 之上，被设计成可在任意一对主机之间工作。Telnet 与我们第 2 章讨论的批量数据传输应用不同，它是一个交互式应用。我们在此讨论一个 Telnet 例子，因为该例子很好地阐述 TCP 的序号与确认号。我们注意到许多用户现在更愿意采用 SSH 协议而不是 Telnet，因为在 Telnet 连接中发送的数据（包括口令!）是没有加密的，使得 Telnet 易于受到窃听攻击（如在 8.7 节中讨论的那样）。

假设主机 A 发起一个与主机 B 的 Telnet 会话。因为是主机 A 发起该会话，因此它被标记为客户，而主机 B 被标记为服务器。（在客户端的）用户键入的每个字符都会被发送至远程主机；远程主机将回送每个字符的副本给客户，并将这些字符显示在 Telnet 用户的屏幕上。这种“回显”（echo back）用于确保由 Telnet 用户发送的字符已经被远程主机收到并在远程站点上得到处理。因此，在从用户击键到字符被显示在用户屏幕上这段时间内，每个字符在网络中传输了两次。

现在假设用户输入了一个字符‘C’，然后喝起了咖啡。我们考察一下在客户与服务器之间发送的 TCP 报文段。如图 3-31 所示，假设客户和服务器的起始序号分别是 42 和 79。前面讲过，一个报文段的序号就是该报文段数据字段首字节的序号。因此，客户发送的第一个报文段的序号为 42，服务器发送的第一个报文段的序号为 79。前面讲过，确认号就是主机正在等待的数据的下一个字节序号。在 TCP 连接建立后但没有发送任何数据之前，该客户等待字节 79，而该服务器等待字节 42。