第3章 传 输 层

本章首先介绍传输层的基本功能及提供的服务，然后依次介绍传输层的两个典型协议——用户数据报文协议 UDP 和传输控制协议 TCP，着重从报文数据格式、关键功能及机制方面进行介绍。

本章学习的目的是掌握传输层在 OSI 七层参考模型中的功能及作用，熟练了解 UDP 与 TCP 的主要功能及特征，并重点了解可靠数据传输的基本原理和关键机制。

3.1 传输层概述

从通信和信息处理两方面来看，传输层既是面向通信部分的最高层，与下面的三层一起共同构建进行网络通信所需的线路和数据传输通道；同时，传输层又是面向用户的最低层，因为无论何种网络应用，最终都需要把各种数据报传送到对方。来自应用层的用户数据必须依靠传输层协议在不同网络主机的应用进程间进行传输，仅靠网络层把数据传送到目的主机上还是不够的，还必须把它交给目的主机的应用进程。因此，可以说传输层的目的是在不同主机的应用进程间提供数据端到端传输的“逻辑通道”。

3.1.1 传输层基本功能

传输层位于应用层之下、网络层之上，实质上它是在网络体系结构中高低层之间衔接的一个接口层。传输层不仅是一个单独的结构层，它是整个分层体系协议的核心，如果没有传输层，那么整个分层协议就没有意义。传输层可以被划入高层，也可以被划入低层。如果从面向通信和面向信息处理的角度看，传输层属于面向通信的低层中的最高层，也就是它属于低层；如果从网络功能和用户功能的角度看，传输层则属于用户功能的高层中的最低层，也就是它属于高层。

传输层的主要功能是为应用进程间提供端到端的数据传输逻辑通道，如图 3-1 所示，通过传输层的服务增加服务功能，使通信子网对两端的用户都变成透明的。也就是说，对高层用户来说，传输层屏蔽了下面通信子网的细节，使高层用户看不见实现通信功能的物理链路是什么，看不见数据链路的规程是什么，看不见下层有多少个通信子网和通信子网是如何联接起来的。传输层使高层用户感觉到的就好像是在两个传输层实体之间有一条端到端的可靠的通信通路。对于应用层而言，应用层可灵活将传输层接口进行封装，而不用关注底层的网络协议，使得应用程序的开发更加灵活。

传输层协议采用端口号(Port Number)来识别不同的应用进程，在传输层数据单元中，需要将源应用的端口号和目的应用的端口号都进行封装，当数据到达传输层后，传输层会识别端口号，将数据发送给相应的应用进程。

传输层是一种独立于网络层的数据传输服务。传输层服务适用于各种网络，因而不必担心不同的通信子网所提供的不同服务及服务质量。从一定程度上而言，传输层是用于填

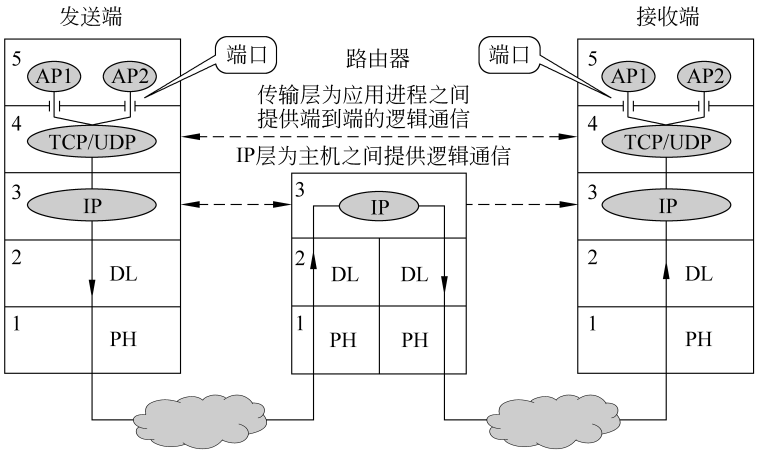


图 3-1　传输层为应用进程间提供逻辑通道

补通信子网提供的服务与用户要求之间的间隙的，其反映并扩展了网络层的服务功能。对传输层来说，通信子网提供的服务越多，传输层协议越简单；反之传输层协议越复杂。

3.1.2　传输层提供的服务

如前所述，传输层的主要功能是为应用进程间提供端到端的数据传输逻辑通道，其主要为应用层提供数据连接管理和数据传送服务，主要包括以下几项服务。

1. 连接管理

传输层能够实现端到端的数据传输，即在数据传输前，经过各种交换设备，在两端建立一条链路。链路建立后，发送端使用该链路发送数据，待数据发送完毕，接收端确认接收成功。连接管理是传输层在发送端和接收端之间建立和释放连接所必须遵循的协议，一般通过三次握手协议来完成两端点连接的建立：计算机 A 传送一个请求一次连接的 TPDU，它的序列号为 X；计算机 B 会发送一个确认该请求及其序列号的 TPDU，它的序列号为 Y；计算机 A 通过在第一个数据 TPDU 中包含序列号 X 和 Y，对计算机 B 的确认帧发回一个确认。

为避免由于请求或确认信息丢失而导致的错误，在计算机 A 和计算机 B 中分别设置了定时器。如果计算机 A 的请求信息或计算机 B 的确认信息丢失，计算机 A 将在计时结束后重新发送请求。如果计算机 A 的确认信息丢失，计算机 B 将在计时结束后终止连接。

当计算机 A 与计算机 B 通信完毕后，需要两端点终止连接操作。而终止连接的操作为：首先计算机 A 请求终止连接，然后计算机 B 确认请求；如果计算机 A 接收到计算机 B 所发送的确认信息，则再发送一条确认信息，并终止连接；最后计算机 B 收到确认信息后，也终止连接。

2. 差错控制

在传输层的通信过程中，无论是面向连接还是面向无连接的传输，都需要对传输的内容进行差错控制编码、差错检测、差错处理三方面的处理。传输层的差错控制是在通信子网对差错控制的基础上采取的最后一道差错控制措施，其出错率相对较低。因此，传输层的差错控制编码一般采用比较简单的算法。例如，在传输层协议数据单元 TPDU 内留有专门的校

验和字段，用于存放校验码。

在对于差错的处理过程中，通常采用当即纠错、通知发送端重传和丢弃三种措施，具体采用什么措施与差错控制算法与传输层服务要求有关。

3. 流量控制

传输层的流量控制是对传输层协议数据单元发送速率的控制。其中包括两方面，分别在两端进行：在发送端控制传输层协议数据单元的发送速率；在接收端控制传输层协议数据单元的接收速率。也就是在同一对传输通信中，发送和接收的速率是各自独立的，这两端的速率可以不一样。传输层协议数据单元的发送与接收速率取决于两端计算机的发送和接收能力，以及通信子网的传输能力。

控制两端计算机收发信息数据单元速率的基本策略是采用缓存的方法，即在两端计算机设置用于缓存协议数据单元的缓存器。

缓存的设置策略主要是对于低速突发的数据传输，在发送端建立缓存；而对于高速平稳的数据传输，为了不增加传输负荷，最大限度利用传输带宽，则在接收端建立缓存。缓存的大小既可以是固定的，也可以是自适应的；可以为每一个传输连接建立一个缓存，也可以多个传输连接循环共用一个大的缓存。

4. 拥塞控制

拥塞现象指到达通信子网中某一部分的分组数量过多，使得该部分网络来不及处理，以致引起该部分乃至整个网络性能下降的现象。

拥塞控制是通过开环控制和闭环控制两种方法来实现的。开环控制是在设计网络时，力求在网络工作时不产生拥塞，但对于复杂的网络系统，使用这种控制方法代价很高，很难实现。所以采用比较容易实现的闭环控制，其实现方法如下：

（1）监测网络系统在何时何处发送了拥塞；

（2）将拥塞的信息传送到可以采取行动的地方；

（3）根据拥塞消息，调整网络系统的运行，解决拥塞。

端到端的拥塞控制就是由网络层将拥塞的信息传送到发送端，由发送端采取措施，控制发往网络的传输数据。

5. 复用与分用

协议各层的软件都要对相邻层的多个对象进行多路复用和多路分解操作。传输层接收多个应用程序送来的数据报，把它们送给网络层进行传输，同时接收从网络层送来的传输层数据，并把它们送给适当的应用程序。传输层模块与应用程序之间所有的多路复用和多路分解都要通过端口机制来实现。

多路复用和多路分解是一对匹配的过程，复用是在发送端将多个端口的数据复用到传输层报文中；分解，也称为解复用，是将从网络层收到的报文根据端口号分解到不同的端口队列中，等待应用进程进行读取。

实际上，每个应用程序在发送数据报之前必须与操作系统进行协商，以获得协议端口和相应的端口号。当指定了端口之后，凡是利用这个端口发送数据报的应用程序都要把端口号放入传输层协议报文的源端口字段中。在处理输入时，传输层从网络层模块接收传入的数据报，根据目的端口号进行多路分解操作。当传输层收到数据报时，先检查当前使用的端口是否就是该数据报的目的端口。如果不能匹配，则丢弃这个数据报；如果匹配，就把这个

数据报送到相应的队列中，等待应用程序的访问。当然，如果端口已满也会出错，传输层也要丢弃传入的这个数据报。

6. 服务质量保证

传输层的主要功能也可以看作优化网络层服务质量。如果网络层提供的服务很完善，那么传输层的工作就很容易，否则传输层的工作就较复杂。对于面向连接的服务，传输服务在建立连接时，需要说明可接收的服务质量参数值。

传输层根据网络层提供的服务种类及自身增加的服务，检查用户提出的参数，如能满足要求则建立连接，否则拒绝连接。服务质量参数包括用户的一些要求，如连接建立延迟、连接失败概率、吞吐率、传输延迟、残余误码率、安全保护、优先级及恢复功能等。

(1) 连接建立延迟：从传输服务用户要求建立连接到收到连接确认之间所经历的时间，包括了远端传输实体的处理延迟，连接建立延迟越短，服务质量越好。

(2) 连接失败概率：在对很长的连接建立延迟时间之内，连接未能建立的可能性，例如，由网络拥塞、缺少缓冲区或其他原因造成的失败。

(3) 吞吐率：某个时间间隔内测得的每秒传输的用户数据的字节数。每个传输方向分别用各自的吞吐率来衡量。

(4) 传输延迟：从发送端计算机传输用户发送报文开始到接收端计算机传输用户接收到报文为止的时间，每个方向的传输延迟是不同的。

(5) 残余误码率：用于测量丢失或乱序的报文数占整个发送的报文数的百分比。理论上残余误码率应为零，实际上它可能是一个较小的值。

(6) 安全保护：为传输用户提供了传输层的保护，以防止未经授权的第三方读取或修改数据。

(7) 优先级：为传输用户提供用以表明哪些连接更为重要的方法，当发生拥塞事件时，确保高优先级的连接先获得服务。

(8) 恢复功能：当出现内部问题或拥塞情况时，传输层本身自发终止连接的可能性。

用户数据报文协议 UDP 和传输控制协议 TCP 是传输层的两个典型协议，传输层根据业务特征和特性选择不同的传输层协议。总体而言，UDP 是一种无连接的传输层协议，在数据传输前不需要在收发两端建立数据传输逻辑通道，并且 UDP 格式简洁、处理简单，但不提供可靠数据传输保障，适合对业务传输实时性要求高的业务；TCP 是一种面向连接的传输层协议，在数据传输前需要建立数据传输逻辑通道，在数据传输结束后需要释放通道资源；较之 UDP，TCP 报文结构和数据处理过程复杂，能够提供重传、流量控制和拥塞控制，适合对数据传输可靠度较高的业务。接下来的章节中将对 UDP 和 TCP 进行详细阐述。

3.2 用户数据报文协议 UDP

用户数据报文协议 UDP 的英文全称为 User Datagram Protocol。UDP 是在互联网中广泛应用的传输层协议之一。用户数据报文，也即应用层所传输来的一个完整数据单元，对于来自应用层的用户数据报文，UDP 不会对这个完整的数据进行处理和拆分，也不会进行合并传输等操作，因此，UDP 也称为面向报文的传输层协议。UDP 的数据长度主要是由应用层传输的数据长度所决定的，应用层传的数据越长，UDP 数据报文就越长。

本节将重点讲述 UDP 的特征、功能及服务，并对 UDP 的数据报文格式和差错检测机制进行阐述。

3.2.1 UDP 功能特征概述

用户数据报文协议(UDP)是一种无连接的传输层协议，提供面向应用层用户报文的简单不可靠的信息传送服务。IETF RFC 768 是 UDP 的正式规范，于 1980 年发布。尽管 UDP 是一个“老”协议，但是 UDP 仍然继续在主流应用中发挥着作用，在当前互联网或通信应用中被广泛使用。

UDP 协议的特点主要如下。

(1) 无连接。UDP 是无连接的传输层协议，在传输应用层数据之前发送端和接收端不需要事先建立连接。当应用层有数据报文需要进行传输时，简单调用 UDP 即可，UDP 简单地抓取来自应用进程的数据报文，做简单处理后交给网络层协议进行处理。在发送端，UDP 传送数据的速度仅受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP 把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

(2) 面向报文。UDP 不对用户报文进行分拆、合并等处理。当收到应用层报文时，仅简单加上 UDP 报文头部，就将封装后的报文发给网络层进行处理。因此，UDP 简单的数据处理方式适合对实时性要求较高的业务应用。

(3) 不可靠。UDP 使用“尽力而为”的数据传输方式，不保证数据的可靠交付。对接收到的数据报不发送确认信号，发送端不知道数据是否被正确接收，也不会重发数据。

(4) 无序性。UDP 不对收到的数据进行排序，UDP 报文的首部中并没有关于数据顺序的信息(如 TCP 所采用的序号)，而且报文不一定按顺序到达，所以接收端并不能保证数据报文是按照发送端的报文顺序向应用层递交的。

(5) 开销小。UDP 报文格式简单，报文头部简洁，进行数据传输时的额外信令开销较小。

(6) 通信灵活。UDP 支持一对一、一对多、多对多等多种通信模式。由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多台客户机传输相同的消息。这样的特性使得 UDP 在诸如视频会议等应用中被广泛使用。

(7) 无流量控制和拥塞控制。UDP 的吞吐量不受拥塞控制算法的调节，只受应用进程的数据报文生成速率、传输带宽、源端和目的端主机性能的限制。

虽然 UDP 在传输层提供的是不可靠的数据传输服务，由于排除了信息可靠传递机制，将安全和排序等功能移交给上层应用来完成，极大节省了执行时间，使其协议处理速度得到了保证，因此，UDP 广泛被多媒体应用所采纳为传输层协议。另外，UDP 能够支持一对多、多对多的通信方式，是一种信息分发的理想协议，因此在电话会议、大屏显示等应用中被广泛使用。此外，基于 UDP 信息分发效率高的特性，UDP 还被用于路由信息协议(Routing Information Protocol，RIP)中修改路由表。

3.2.2 UDP 报文结构

UDP 报文结构简单，其报文首部仅有 8 字节，如图 3-2 所示。

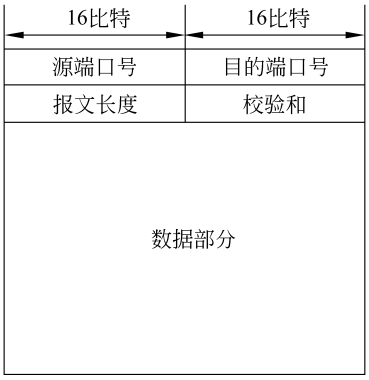


图 3-2 UDP 报文结构

(1) 源端口号(Source Port)：2 字节。当使用时，它表示发送程序的端口，同时它还被认为是没有其他信息的情况下需要被寻址的答复端口。如果不使用，设置值为全 0。

(2) 目的端口号(Destination Port)：2 字节，用来标记数据传输目的端应用进程的端口标识。

(3) 报文长度(Length)：2 字节，用来标记数据报文的长度，包括 UDP 报文首部及数据部分。长度最小值为 8，即仅传输 UDP 报文头部，而无任何载荷数据。

(4) 校验和(Checksum)：2 字节，可选字段。UDP 中用来进行差错检测的手段，具体方法将在后文进行阐述。该数据字段为 UDP 报文首部、数据部分和 UDP 伪报文首部的差错检测值。伪报头并不会在网络中传送，校验和中所包含的伪报头内容可以避免目的端错误地接收错误路由的数据报。一般而言，仅在局域网内部传输报文不需要 UDP 校验和，因为以太网帧的校验和已经提供了差错控制。而对于那些需要通过不同的、也许未知网络传输的报文而言，校验和可以让目的主机检测到错误数据。UDP 使用报头中的校验值来保证数据的安全。校验值首先在数据发送方通过特殊的算法计算得出，在传递到接收方之后，还需要再重新计算。如果某个数据报在传输过程中被第三方篡改，或者由于线路噪声等原因受到损坏，发送和接收方的校验计算值将不会相符，由此 UDP 可以检测是否出错。

UDP 报文伪首部和 UDP 报文的封装流程如图 3-3 所示。

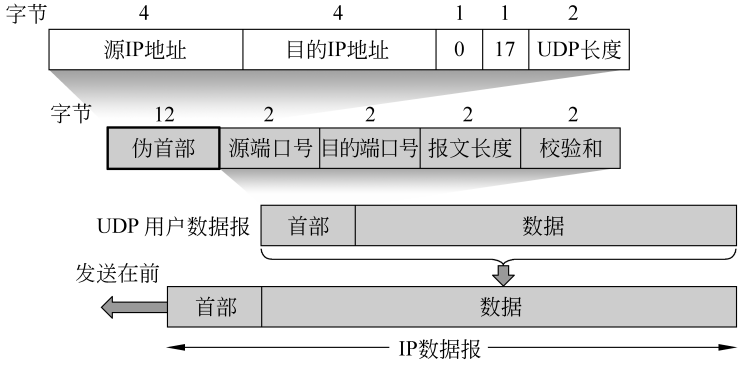


图 3-3 UDP 报文伪首部及封装流程

由图 3-2 可以看到，UDP 报文伪首部并不会在网络中进行发送，仅用于进行校验。UDP 报文伪首部包括源和目的 IP 地址，各占 4 字节，1 字节的全 0 填充，1 字节的协议号(IP 协议号为十进制的 17)，UDP 报文长度是 2 字节。从图 3-2 中也可以看出，UDP 接收到来自应用层的数据后，在数据前面加上 UDP 报文首部，之后发送到网络层。在 TCP/IP 架构中，网络层主要采用 IP，因此在图 3-2 中，将 UDP 报文放入 IP 报文数据部分，添加 IP 报文头部后形成网络层的 IP 数据分组格式。

UDP 校验和覆盖的内容超出了 UDP 数据报本身的范围。为了计算校验和，UDP 把伪首部引入数据报中，在伪首部有一个值为 0 的填充八位组用于保证整个数据报的长度为 16

比特的整数倍，这样才容易计算校验和。填充八位组和伪首部并不随着 UDP 数据报一起传输，也不计算在数据报长度之内。为了计算校验和，要先把校验和字段置为 0，然后对整个对象，包括伪首部、UDP 的首部和用户数据报，计算一个 16 比特的二进制反码和。使用伪首部的目的是检验 UDP 数据报已到达正确的目的地。理解伪首部的关键在于认识到：正确的目的地包括了特定的主机和机器上特定的协议端口。UDP 报文的首部仅指定了使用的协议端口号。因此，为了确保数据报能够正确到达目的地，发送 UDP 数据报的机器在计算校验和时把目的机的 IP 地址和应有的数据都包括在内。在最终的接收端，UDP 协议软件对校验和进行检验时要用到携带 UDP 报文的 IP 数据报首部中的 IP 地址。如果校验和正确，说明 UDP 数据报到达了正确主机的正确端口。

伪首部的源 IP 地址字段和目的 IP 地址字段记录了发送 UDP 报文时使用的源 IP 地址和目的 IP 地址。协议字段指明了所使用的协议类型代码(UDP 是 17)，而长度字段是 UDP 数据报的长度。接收方进行正确性验证时，必须要把这些字段的信息从 IP 报文的首部中抽取出来，以伪首部的格式进行装配，然后再重新计算校验和。

在接收端，最底层的网络软件在接收到一个分组后将它提交给上一层模块。每一层都在向上送交数据之前剥去本层的首部，因此当最高层的协议软件将数据送到相应的接收进程时，所有附加的首部都被剥去了。也就是说，最外层的首部对应的是最底层的协议，而最内层的首部对应的是最高层的协议。研究首部的生成与剥除时，可从协议的分层原则得到启发。当把分层原则具体应用于 UDP 时，可以清楚地知道目的机上的由 IP 层送交 UDP 层的数据报就等同于发送机上的 UDP 层交给 IP 层的数据报。同样，接收方的 UDP 层上交给用户进程的数据也就是发送方的用户进程送到 UDP 层的数据。在多层协议之间，职责的划分是清楚而明确的，IP 层只负责在互联网上的一对主机之间进行数据传输，而 UDP 层只负责区分一台主机上的多个源端口或目的端口。

3.2.3 UDP 中的差错检测机制

UDP 虽然提供的是不可靠的数据传输服务，但并非完全不关心数据传输的可靠性。为了方便接收端判断数据在传输过程中是否发生了差错，UDP 采用了校验和作为数据传输差错检测的机制。校验和的详细计算可在 RFC 1071 中找到，并非只在 UDP 中使用，在 TCP 或 IP 中，也可以使用校验和判断报文在传输过程中是否发生了差错。

在 UDP 中，校验和字段的长度为 16 比特，因此，在对 UDP 报文首部和数据部分进行校验和运算时，一般需要将数据分成以 16 比特为单位的码字进行二进制加法运算。下面将通过示例对校验和的运算方式进行解释。

如图 3-4 所示，两个码字以 16 比特为一组，然后把所有字段的码字进行二进制进位加法运算。如图 3-3 所示，当最高位有进位时，需要把待进位的“1”放到得到的和的低位继续进行加法运算。当得到所有码字的和后，逐位进行取反，得到需要填充到校验和字段的校验和值。

从校验和的计算方法可以看到，所有字段经过校验并将得到的值填充到校验和字段后，所有字段(包括校验和)的和为全“1”。如果在接收端接收到的数据按 16 位二进制之和也全是“1”，则认为传输过程没有出差错。

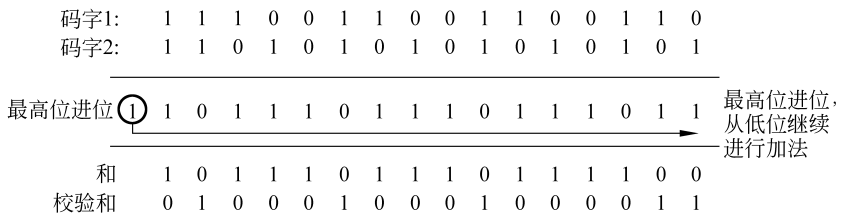


图 3-4 UDP 校验和示例

3.3 可靠数据传输机制概述

可靠数据传输是一种技术，它可以保证数据在传输过程中根据协议安全地传输，不受环境因素的干扰而被破坏或损坏，无论是从发送者到接收者传输数据，还是从接收者发回数据给发送者，都能保证可靠的传输。简单而言，对于可靠数据传输的定义可以归纳为：不出错、不丢失、不重复、不失序。

为了保证数据的可靠传输，协议的设计非常重要。可靠的协议提供了一种有效的方法来在传输过程中有效地处理任何可能发生的问题，如数据丢失、分组错误等。根据通信双方协商的协议，可以指定不同的处理方式，以便可靠传输。需要强调的是，可靠数据传输机制并非只在传输层中使用，每一层协议均有可靠数据传输的需求，也会根据协议所在层的特点采用不同的可靠数据传输技术。

传输介质或传输信道对数据传输的可靠性影响较大。若信道是理想信道，即信道不会发生任何误码、差错、丢失，那么就不需要复杂的可靠数据传输技术，收发端只需要连续地发送或接收数据分组即可。然而，实际情况中并不存在理想信道，信道都是非可靠的，会受到距离、环境、终端状态和服务器状态等因素的干扰，因此，需要特殊的机制解决数据传输中的延迟、误码、丢包等问题。

重传机制是保证数据传输的有效手段，虽然会带来额外的时延，但能保证发送端的数据均能被接收端所正确接收。一旦数据包丢失或在传输过程中发生错误，均可以通过数据重传解决。因此，自动重传请求（Automatic Repeat reQuest，ARQ）机制成为保证数据可靠传输的关键协议，ARQ 协议的基本原理是发送方在发送每个数据包之后等待接收方的确认信号，如果发送方在一定时间内没有收到确认信号，或者接收方发送了错误的确认信号，发送方将重新传输该数据包，这样可以确保数据在传输过程中的可靠性。ARQ 协议的重传机制是其中最重要的部分。当发送方没有在规定时间内接收到确认信号时，会将数据包进行重传。为了避免无限重传，ARQ 协议通常会设置一个重传次数的上限。如果在达到重传次数上限后仍然没有收到确认信号，发送方将停止发送并通知用户进行相应的处理。

常用的 ARQ 机制包括“停止-等待”ARQ 协议、回退 N 步 ARQ 和自动选择重传 ARQ 三种形式，下面将分别进行介绍。需要注意的是，ARQ 机制并非是传输层独有的机制，在数据链路层，为提升链路传输可靠性，也会采用自动重传请求的方式。

3.3.1 “停止-等待”ARQ 协议

“停止-等待”（Stop-Wait）ARQ 协议的基本思想是，数据发送端在发送一个数据分段

(segment)后,等待接收端正确接收,并发回一个确认信息后,数据发送端再开始发送新的数据分段。对于数据发送节点和接收节点,其发送流程分别如下。

数据发送节点流程如下。

(1) 主机应用进程有数据到达,从应用进程取一个数据分段。

(2) 将数据分段送到传输层的发送缓存,并进行存储。

(3) 从发送缓存中将数据进行协议封装后发送。

(4) 等待。

(5) 收到数据接收节点发送回来的反馈消息,若反馈消息是确认消息(ACK),即确认数据分段已经被接收端正确收到,则回到步骤(1)重新执行,发送新的数据分段;若收到的反馈消息是非确认消息(NAK),即表明收到的数据分段存在差错,接收端不能正确接收,则回到步骤(3)开始执行,重新传输未被成功接收的数据分段(仍然在缓存中存储)。

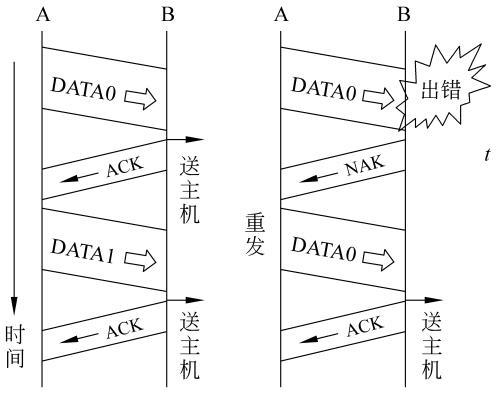
数据接收节点流程如下。

(1) 等待接收由底层递交的数据分段。

(2) 检测数据分段是否在传输过程中出现差错(例如采用校验和),若数据在传输过程中无差错,则将该数据分段递交给应用进程,并且向数据发送端发送一个确认接收消息;若数据在传输过程中出现差错,则丢弃该数据分段,并且向数据发送端发送一个非确认接收消息。

(3) 回到步骤(1)继续等待。

上述数据发送和接收过程如图 3-5 所示。然而,当传输过程中存在数据分段或者是反馈消息丢失的情况,可发现图 3-5 给出的数据重传流程陷入了“死锁”,数据发送端无法收到接收端发出的反馈消息,从而一直处于“等待”状态,导致整个数据传输进程被“挂起”。



(a) 正常情况　　(b) 数据帧出错

图 3-5　无数据丢失的数据重传流程

因此,为了避免这种情况的出现,当发送端发送数据分段时,需要启用一个超时定时器,当定时器达到重发时间而未收到反馈消息时,则不需要继续等待,重传该数据分段,并重置定时器时间,如图 3-6 所示。超时定时器设置的重发时间应仔细选择确定。若重发时间选得太短,则在正常情况下也会在对方的应答信息回到发送方之前就过早地重发数据。若重发时间选得太长,则会浪费时间。一般可将重发时间选为略大于“从发完数据帧到收到应答

帧所需的平均时间”。

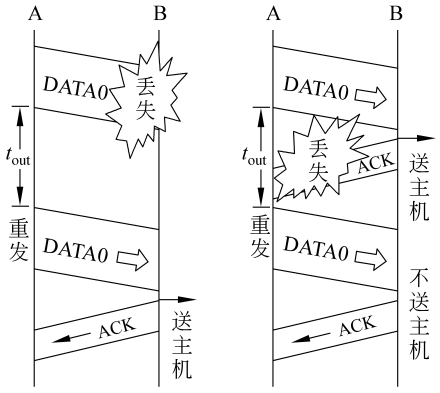


图 3-6 增加定时器的数据重传流程

然而，若只有上述机制，也不能完全保证数据的可靠传输。当出现反馈消息（假设消息中包含的是对数据分段正确接收的确认）丢失的场景，由于数据接收端现在无法识别重复的数据帧，因而在数据接收端收到的数据中出现了另一种差错——数据分段重复。要解决这个问题，必须为每一个数据分段编号，每发送一个新的数据分段就把它的发送序号加 1。若数据接收节点收到发送序号相同的数据分段，就表明出现了重复数据分段，这时应当丢弃重复的数据。注意，此时数据接收端还必须向数据发送端发送一个确认信息 ACK，因为接收端已经知道发送端还没有收到上一次发过去的确认信息 ACK。

序号所占用的比特数是有限的。因此，经过一段时间后，发送序号就会重复。对于停止等待协议，由于每发送一个数据分段就停止等待，因此用 1 比特来编号就够了。比特可以有 0 和 1 两种不同的序号。这样，数据分段中的发送序号就以 0 和 1 交替的方式出现在数据帧中。每发送一个新的数据分段，发送序号就和上次发送的不一样。用这样的方法就可以使接收方能够区分开新的数据分段和重发的数据分段了。

至此，“停止-等待”ARQ 协议的所有机制和步骤都已介绍完毕。其机制的核心是为数据分段编号，发送端发送一个数据包后等待接收端的确认消息，发送端的重传定时器超过提前设置的阈值时间或收到接收端发送的非确认消息（表示信息未成功接收），则触发重传，否则就发送新的数据分段。“停止-等待”ARQ 的方式能够避免数据丢失、重复或失序，实现数据的可靠传输。

“停止-等待”ARQ 协议实现简单，但在实际应用中却很少使用，因为这样的方式效率极低，极大浪费了网络/链路传输资源。

为了兼顾数据传输可靠性与资源有效性，在实际网络中更多使用连续 ARQ 协议，发送端可以在一定范围内连续发送，而不用发送一个数据分段后等待确认才能发送下一个数据分段，从而提升网络利用率。对于连续 ARQ 协议，根据其接收端发送确认信息的方式不同，又分为回退 N 步 ARQ 协议和选择重传 ARQ 协议，将在下述内容中详细叙述。

3.3.2 回退 N 步 ARQ 协议

回退 N 步 ARQ 协议称为 Go-Back-N（GBN），其核心思想是在发送端允许未收到接收

端确认消息的分段，也可持续发送，为了限制其发送的速率，在发送端建立了发送滑动窗口，只有分段序号在发送滑动窗口中的数据才允许发送，发送端每收到一个确认，就把发送窗口向前滑动；在接收端，采用的是累积确认方式，即仅对按序到达的最后一个可确认的分段发送确认信息，表示到这个分段为止的所有分段都已经正确收到了。

在 GBN 中，当接收端收到不按序到达的数据分段时，将不会对接收到的不按序到达分段进行存储，而是直接丢弃，发送的确认信息仍然是当前接收端能够确认的按序到达的最后一个分段编号；当发送端收到多个重复的确认消息，或者是相应分段超时，则会重传当前未确认的最早的分段及其在发送窗口内已经发送过的 N 个数据分段，其过程如图 3-7 所示。

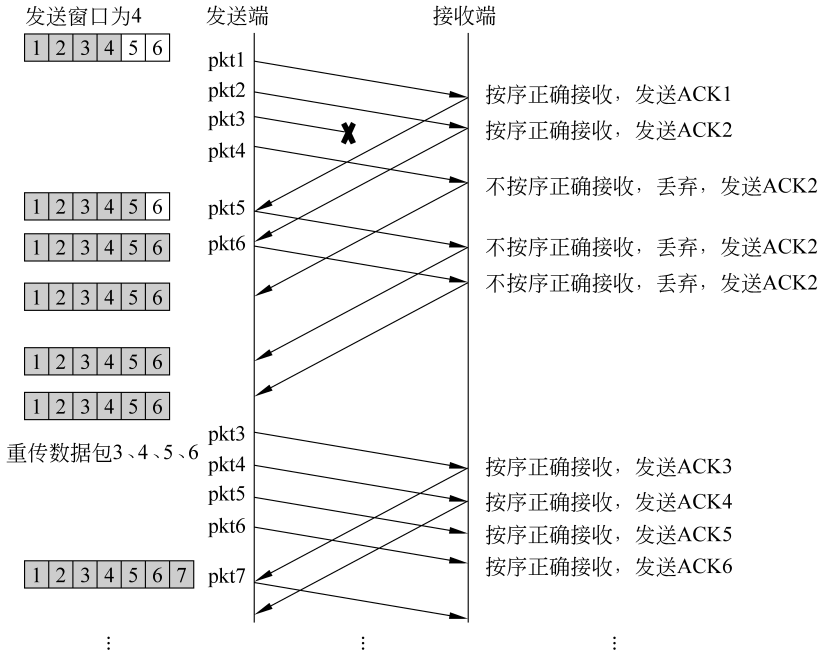


图 3-7　回退 N 步 ARQ 协议流程

从图 3-7 中可以看到，接收端由于只对按序到达的最后一个分段进行确认，当因数据丢失造成数据不按序到达时，接收端就会重复发送确认信息，并且丢弃不按序到达的数据帧；虽然不按序到达的数据分段 pkt4、pkt5、pkt6 均被丢弃了，但是在重传过程中，该分段都被重新传输了。由此可以看出，GBN 协议的本质是采用滑动窗口大于 1 的发送窗口，但接收端的接收窗口长度仅为 1。对于 GBN 协议，其发送端的定时器仅提供给当前发送窗口中未确认的最小编号的分段，而不用给所有分段都提供定时器。

总而言之，回退 N 步可以让发送端实现连续发送数据，是一种连续 ARQ 协议，其接收端处理方式简单，没有复杂的状态需要“记忆”，因而其实现也较为容易。然而，在通信线路质量不好的情况下，若经常发生数据包丢失，回退 N 步的连续 ARQ 协议由于会经常回退并进行大量的重传，从而影响其数据传输的连续性和有效性。

3.3.3　选择重传 ARQ 协议

与回退 N 步 ARQ 协议一样，选择重传也是一种连续的 ARQ 协议，但与 GBN 在接收

端的处理不同，选择重传允许不按序到达数据包的接收和确认，提升了数据传输的效率，但选择重传 ARQ 协议在接收端的处理也会比 GBN 复杂得多。

选择重传 ARQ 协议分别在收发两端定义两个滑动窗口，分别是发送窗口和接收窗口，只要在发送窗口规定的编号范围内的数据分段，均可以进行连续发送，对于接收窗口，若在接收窗口规定的编号范围内的数据分段被正确接收，即使该分段是不按序到达的，也可以给该数据分段发送确认信息。在发送端，发送窗口根据收到的确认信息情况按序进行滑动，即发送窗口的最左端一定是当前已发送但未收到确认信息的最小编号数据分段；在接收端，由于允许对不按序到达数据分段的确认，接收窗口内存在三种状态：第一种是不按序到达但已经发送了确认信息的数据分段；第二种是已发送但尚未收到的数据分段（如序号比已确认数据分段编号小的数据分段，说明已经发送了，但由于各种原因接收端尚未收到）；第三种是在窗口范围内可允许被接收但尚未发送的数据分段。接收端滑动窗口需要保证数据按序递交给上层应用程序，因此接收端窗口的最左端一定是目前期待收到，但尚未正确收到的最小编号的数据分段。需要注意的是，与 GBN 协议不同，选择重传 ARQ 协议为每个数据分段均提供了超时时钟。因此，对于未接收到的分段，通过超时触发重传。选择重传 ARQ 协议具体流程如图 3-8 所示。

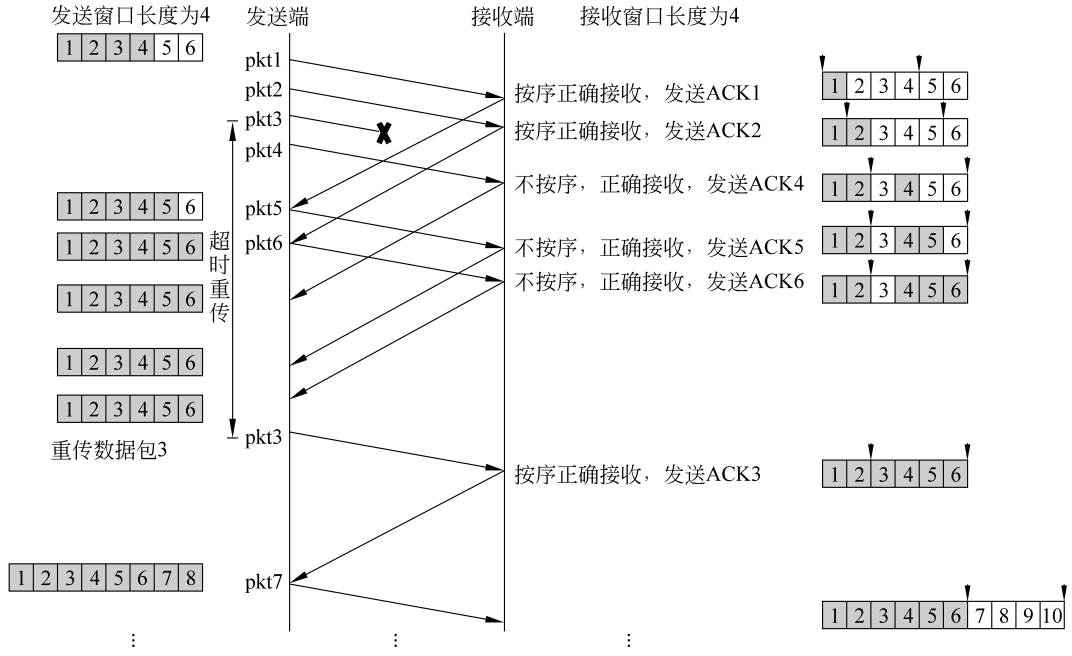


图 3-8　选择重传 ARQ 协议流程

值得注意的是，由于数据分段编号是循环使用的，假设采用 n 比特用于数据分段编号，即编号数为 2^n，假设发送窗口长度为 W_s，接收窗口长度为 W_r，则应满足 $W_s+W_r\leqslant 2^n$，且 $W_s\geqslant W_r$。

3.3.4　流量控制与拥塞控制

流量控制与拥塞控制是保证网络传输稳定可靠的重要机制。在一定程度上而言，流量

控制是解决网络拥塞的方法之一，通过控制发送端的数据流量，能够很大程度上缓解网络拥塞，保障网络的可用性。然而，流量控制与拥塞控制在本质上有较大差别，本节将对流量控制与拥塞控制的概念和原理进行阐述。

1. 流量控制原理及基本方法

流量控制是在发送方和接收方之间调节数据传输速率的过程，以保证接收方能够及时处理和接收数据，防止数据丢失或溢出。在计算机网络中，当发送方发送数据的速率高于接收方处理数据的速率时，会出现流量不平衡的情况，这时就需要流量控制来调整数据发送的速率。

流量控制通常通过使用滑动窗口机制来实现。发送方维护一个发送窗口的大小，表示当前可以发送的数据量。接收方通过调整发送窗口的大小来告知发送方自己当前的处理能力。当发送方收到接收方的窗口大小信息后，会根据接收方的处理能力进行相应的调整，确保发送的数据不会超过接收方的处理能力，从而避免数据的丢失和溢出。

流量控制的原理包括了两种主要的方式，一种是基于速率的流量控制，另一种是基于容量的流量控制。基于速率的流量控制通过限制数据传输的速率来控制网络流量，例如限制某个接口的最大传输速率，从而达到控制网络流量的目的。而基于容量的流量控制则通过限制数据包的数量来控制网络流量，例如限制某个接口的最大数据包数量，从而达到控制网络流量的目的。

前文提到的 GBN 协议或选择重传 ARQ 协议，都能通过发送窗口大小实现流量控制。当然，除了控制发送端流量发送速率外，流量控制的技术手段还包括流量分级、流量整形等，流量分级指通过对不同类型的数据流进行分类和优先处理，来保证网络中重要数据的传输质量。流量整形指通过对数据包的发送时间进行调整，来平滑地控制数据的传输速率，避免突发的大流量对网络造成冲击。

2. 拥塞控制原理及基本方法

拥塞控制是在计算机网络中控制数据传输速率以避免网络拥塞的过程。当网络的数据流量过大，导致网络中的路由器和链路资源无法及时处理和传输数据时，就会出现拥塞。拥塞会导致网络质量下降，造成数据丢失、延迟增加和带宽浪费等问题。因此，拥塞控制是保证网络性能的关键机制。

与流量控制不同，拥塞控制是一个全局性的过程，涉及所有主机、所有路由器，以及与降低网络传输性能有关的所有因素。拥塞控制生效的前提，就是网络能够承受现有的网络负荷。对于拥塞控制措施，并不能真正等到网络出现大量拥塞时才进行控制，而需要对网络中业务传输质量指标进行监控，一旦业务传输质量指标出现了恶化，如出现了重传，就认为网络负载到达了一定的程度，需要采取相应的拥塞控制措施；否则，一旦网络出现了严重拥塞，即使采取了拥塞控制措施，也会造成网络性能大幅下降。拥塞控制与吞吐量的关系如图 3-9 所示。

在计算机网络中，拥塞控制通常通过使用拥塞窗口控制和网络反馈机制来实现。发送方维护一个拥塞窗口的大小，表示当前网络的拥塞程度。发送方根据接收到的网络反馈信息来调整拥塞窗口的大小，以控制数据发送的速率。当网络拥塞时，接收方会向发送方发送拥塞通知，发送方收到通知后会减小拥塞窗口的大小，从而降低数据发送的速率，以缓解网络拥塞的情况。

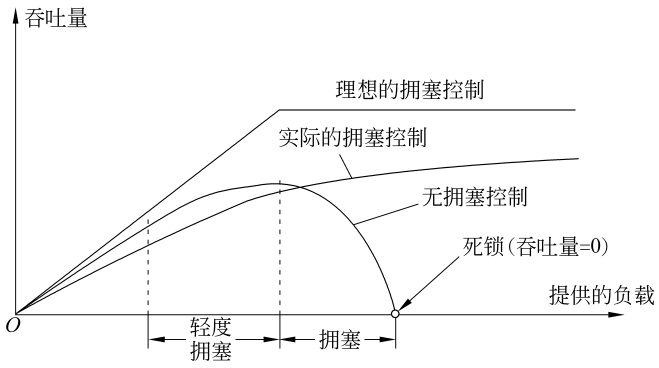


图 3-9 拥塞控制与网络吞吐量的关系

值得注意的是,流量控制和拥塞控制在计算机网络中都起着重要的作用,但是它们的目标和实现方式略有不同。

流量控制主要是为了保证接收方能够及时处理和接收数据,防止数据丢失或溢出。它是在发送方和接收方之间进行的,通过滑动窗口机制来调整数据的发送速率。

拥塞控制主要是为了避免网络拥塞,保证网络质量和性能。它是在发送方和网络之间进行的,通过拥塞窗口控制和网络反馈机制来调整数据的发送速率。

流量控制和拥塞控制都是通过动态地调整数据发送的速率来实现的,但是流量控制主要关注点在与发送方和接收方之间的数据流量平衡,而拥塞控制主要关注点在于网络中的拥塞状况和网络资源的利用情况。

3.4 传输控制协议 TCP

传输控制协议(Transmission Control Protocol, TCP)是互联网协议族的主要基础协议之一,是为了在不可靠的互联网络上提供可靠的端到端字节流传输而专门设计的传输协议。TCP 的正式定义由 1981 年 9 月的 RFC 793 给出,随着技术的发展,TCP 协议也在不断地完善和自我修正,满足飞速发展的互联网业务需求。其中重要的补充协议包括: RFC 1122 修复了原有协议中出现的漏洞和缺陷;RFC 1323 对 TCP 做了高性能扩展;RFC 2018 定义了选择性确认;RFC 2873 定义了为服务质量而重用的报文首部字段;RFC 2988 改进了重传计时器;RFC 3168 定义了显示拥塞通知。

总而言之,TCP 是传输层重要的基础协议,是当前诸多信息通信应用广泛采用的经典传输层协议。为了更好地了解 TCP 的各项关键机制,本节首先概述 TCP 的功能特征和关键机制要点;然后介绍 TCP 报文结构和 TCP 连接的建立和释放流程,并分两部分分别介绍 TCP 的可靠数据传输机制,包括 TCP 数据重传机制、TCP 流量控制和拥塞控制机制。

3.4.1 TCP 功能特征概述

TCP 是一种面向连接的传输层协议,能够为业务在不可靠网络上提供可靠的数据传输服务。为了实现数据传输的可靠性,相比于 UDP,TCP 在数据处理机制上更加复杂。

TCP 的主要特点如下。

(1) 面向连接。为了保证收发双方能够为 TCP 数据传输做好资源准备,需要在开始数据传输前,提前建立 TCP 连接;并在数据传输结束后,释放 TCP 连接。

(2) 双向通信。TCP 建立的是一个双向数据传输的通道,一旦 TCP 连接建立完成,允许收发双方在建立的连接上进行双向数据交互。

(3) 基于流的数据传输方式。与 UDP 面向数据报文、不会对数据报文进行任何处理不同,TCP 是面向字节的,会将来自应用层的数据流进行编号,TCP 则把数据流分割成适当长度的报文段(Segment)。在接收方,接收端 TCP 会逐字节确认,保证数据流的正确接收和按序交付。

(4) 数据分片。在发送端对用户数据进行分片,在接收端进行重组,由 TCP 确定分片的大小并控制分片和重组。

(5) 数据校验机制。TCP 将保持它首部和数据的校验和,这是一个端到端的校验和,目的是检测数据在传输过程中的任何变化。如果收到分片的检验和有差错,TCP 将丢弃这个数据分段,出现差错的数据将会进行重发。

(6) 到达确认机制。接收端接收到分片数据时,接收端会根据数据接收情况(数据是否出错)和分片数据序号向发送端发送一个确认。

(7) 数据重传机制。TCP 在进行数据分段发送时启动定时器,若数据分段丢失导致超时,或因为接收端接收到不正确数据分段(如传输过程发生错误)发送重复确认指令时,发送端会重新发送该数据分段。

(8) 流量控制机制。TCP 通过滑动窗口机制,能够根据接收端情况控制发送端数据发送速度,从而进行流量控制,避免接收端缓存溢出。

(9) 拥塞避免机制。TCP 发送端维持一个拥塞窗口,该拥塞窗口大小采用“加性增加,乘性减小”(AIMD)方式,根据网络拥塞状况动态调整控制发送端数据发送速率,避免网络拥塞。

3.4.2 TCP 报文数据格式

TCP 报文是 TCP 间进行数据交互的基础数据格式。如图 3-10 所示,该图给出了 TCP 的数据报文格式,在该数据报文中,每一行按照 32 位进行排列。

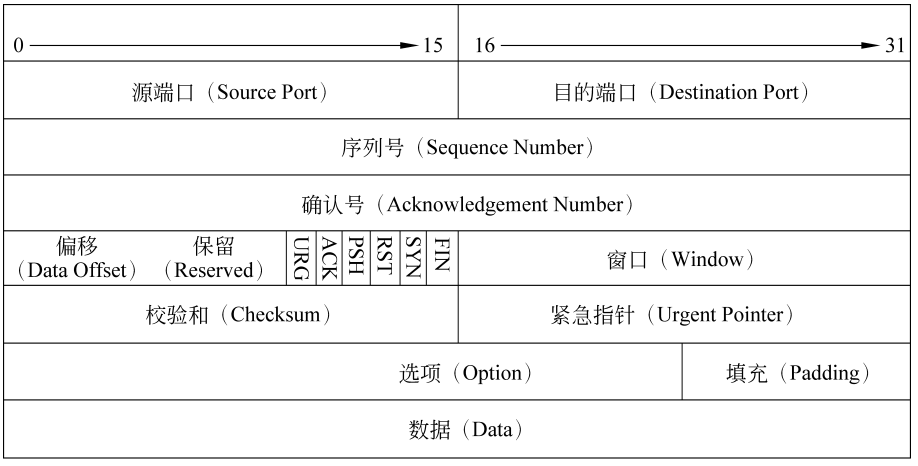


图 3-10 TCP 数据报文格式

(1) 源端口(Source Port)：16 位的源端口字段包含初始化通信的端口号。源端口和 IP 地址的作用是标识报文的返回地址。

(2) 目的端口(Destination Port)：16 位的目的端口字段定义传输的目的。这个端口指明接收方计算机上的应用程序接口。

(3) 序列号(Sequence Number)：该字段用来标识 TCP 源端设备向目的端设备发送的字节流，表示在这个报文段中的第几个数据字节。序列号是一个 32 位的数。

(4) 确认号(Acknowledge Number)：TCP 使用 32 位的确认号字段标识期望收到的下一个段的第一字节，并声明此前的所有数据已经正确无误地收到，因此，确认号应该是上次已成功收到的数据字节序列号加 1。收到确认号的源计算机会知道特定的段已经被收到。确认号的字段只在 ACK 标志被设置时才有效。

(5) 数据偏移(Data Offset)：4 位字段，包括 TCP 头大小。由于首部可能含有选项内容，因此 TCP 首部的长度是不确定的。首部长度的单位是 32 比特或 4 个八位组。首部长度实际上也指示了数据区在报文段中的起始偏移值。

(6) 保留(Reserved)：6 位置 0 的字段，为将来定义新的用途保留。

(7) 标志位(Flag Bits)：共 6 位，每一位标志可以打开一个控制功能。

① URG(Urgent Pointer Field Significant，紧急指针字段标志)：表示 TCP 包的紧急指针字段有效，用来保证 TCP 连接不被中断，并且督促中间设备尽快处理这些数据。

② ACK(Acknowledgement Field Significant，确认字段标志)：取 1 时表示应答字段有效，即 TCP 应答号将包含在 TCP 段中；取 0 则反之。

③ PSH(Push Function，推功能)：这个标志表示 Push 操作。所谓 Push 操作就是在数据包到达接收端以后，立即送给应用程序，而不是在缓冲区中排队。

④ RST(Reset the Connection，重置连接)：这个标志表示连接复位请求，用来复位那些产生错误的连接，也被用来拒绝错误和非法的数据包。

⑤ SYN(Synchronize Sequence Numbers，同步序列号)：表示同步序号，用来建立连接。

⑥ FIN(No More Data From Sender)：表示发送端已经发送到数据末尾，数据传送完成，发送 FIN 标志位的 TCP 段，连接将被断开。

(8) 窗口(Window)：目的主机使用 16 位的窗口字段告诉源主机能够接收的数据量大小。

(9) 校验和(Checksum)：TCP 头包括 16 位的校验和字段用于错误检查。源主机基于部分 IP 头信息，TCP 头和数据内容计算一个校验和，目的主机也要进行相同的计算，如果收到的内容没有错误，两个计算应该完全一样，从而证明数据的有效性。

(10) 紧急指针(Urgent Pointer)：紧急指针字段是一个可选的 16 位指针，指向段内的最后一字节位置，这个字段只在 URG 标志被设置时才有效。

(11) 选项(Option)：至少 1 字节的可变长字段，标识哪个选项(如果有)有效。如果没有选项，这一字节等于 0，说明选项的结束。这一字节等于 1，表示无须再有操作；等于 2，表示下 4 字节包括源机器的最大长度(Maximum Segment Size，MSS)。

(12) 填充(Padding)：加入额外的 0，以保证 TCP 头是 32 的整数倍。

从图 3-11 中可以看到，TCP 报文首部的固定部分是 20 字节，主要完成了 TCP 源端口

和目的端口、进行 ARQ 协议的序号和确认号、首部长度及标识位、接收窗口长度、校验和等主要功能。

对于 TCP 报文而言，其数据部分具有最大的分段长度，若应用层报文超过了 TCP 最大分段长度，则需要对应用层报文进行拆分传输。TCP 的封装过程如图 3-11 所示。

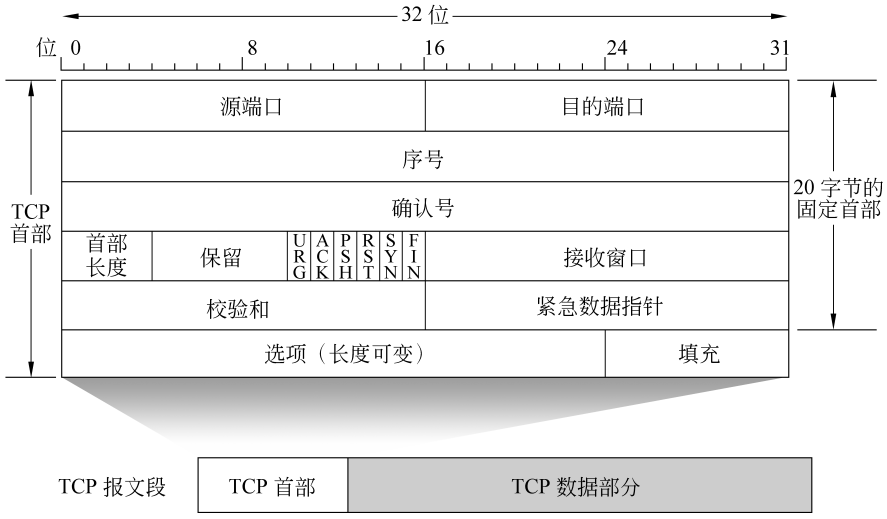


图 3-11　TCP 的封装过程

3.4.3　TCP 连接建立与释放流程

与 UDP 是无连接的传输方式不同，TCP 是一种面向连接的传输方式，即在进行 TCP 数据报文进行传输前，需要先建立 TCP 连接，通知收发双方做好本地资源准备；只有当 TCP 连接建立后，双方才开始进行 TCP 报文数据的交互；当通信完成后，再释放 TCP 连接。通过这样的方式 TCP 能够为应用层提供有保障的数据传输保障。本节将重点针对 TCP 连接的建立和释放流程进行阐述。

TCP 连接有三个阶段，即连接建立、数据传送和连接释放。传输连接的管理就是使传输连接的建立和释放都能正常地进行。连接建立的过程中要解决以下三个问题：①使每一方能够确知对方的存在；②允许双方协商一些参数（如最大报文段长度、最大窗口大小、服务质量等）；③能够对传输层的实体资源（如缓存大小等）进行分配。

1. TCP 连接建立过程

TCP 连接建立过程一般分为三个步骤，因此 TCP 建立过程也称为“三次握手”，握手的目的是确保双方的状态同步，并建立可靠的数据传输逻辑通道。TCP 连接建立流程如图 3-12 所示。

在图 3-12 中，客户端 A 需要与服务器 B 之间建立 TCP 连接，由客户端 A 先发送同步报文进行第一次握手，此时 TCP 报文中 SYN 标识位为 1，并将序列号置为一个随机数 x，表明客户端要请求建立 TCP 连接。

第二次握手是服务器端对客户端 SYN 信号的响应，具体过程为：服务器端收到客户端同步信号后，向客户端发送一个同步确认信号，此时 TCP 报文中 SYN 标识位为 1，ACK 标

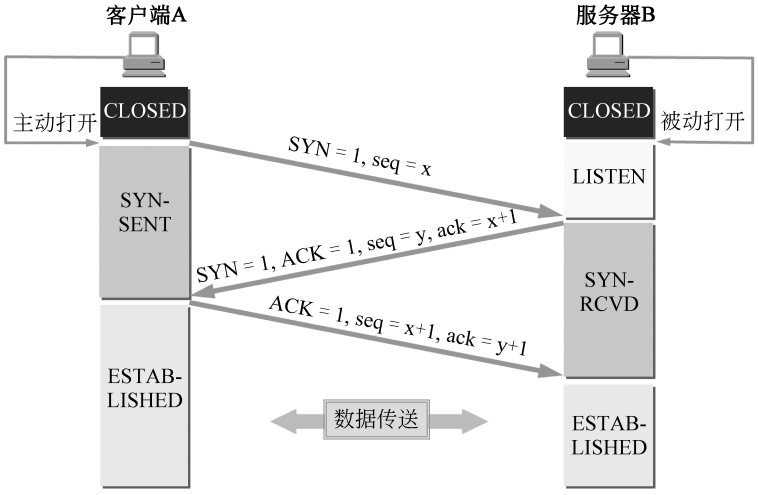


图 3-12 TCP 连接建立流程

识位也为 1，确认号为 x+1，并将序列号置为随机数 y。该信息表示服务器收到了客户端的请求，并准备建立 TCP 连接。

第三次握手是客户端对服务器端发送的同步确认信号的再次确认，具体过程为：客户端发送对同步确认信号的回复，TCP 报文中 ACK 标识位为 1，需要注意的是，由于 TCP 连接建立的同步过程已经完成，此时 SYN 位将不再置位。基于前面两个报文中交换的序列号，此时 TCP 报文中序列号为 x+1，确认号为 y+1。进行第三次握手的目的是表示客户端收到了服务器的响应，并确认可以建立连接。

2. TCP 连接释放过程

当完成 TCP 数据报文交互后，需要释放 TCP 连接，由于该过程需要收发双方进行四次交互，因此也将 TCP 连接释放过程称为“四次挥手”，其流程如图 3-13 所示，挥手的目的是通知对方连接即将关闭，并确保双方都完成了数据传输。

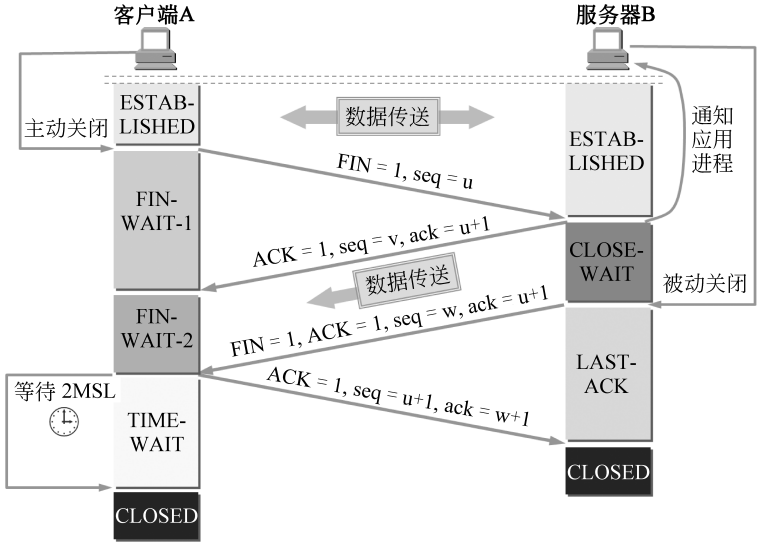


图 3-13 TCP 连接释放流程

第一次挥手：当客户端完成了 TCP 数据交互，则在 TCP 报文中将 FIN 进行置位，表示客户端将希望结束 TCP 连接，并在此消息后不再发送数据。

第二次挥手：服务器收到客户端的 FIN 信号后，向客户端发送一个 ACK 信号，表明收到了客户端结束 TCP 连接的请求；此时，服务器端若有数据还可以继续向客户端进行发送，而客户端将不再回复。

第三次挥手：当服务器也没有数据发送给客户端后，服务器向客户端发送一个 FIN 信号，服务器端准备释放 TCP 连接。

第四次挥手：客户端收到服务器的 FIN 信号后，向服务器发送一个 ACK 信号，表示收到了服务器的结束请求。

当完成四次挥手，服务器端收到确认报文后，就完成了 TCP 连接的释放，而客户端仍然需要等待两个最大分组生存时间(Maximum Segment Lifetime，MSL)后，才最终关闭 TCP 连接。需要注意的是，MSL 指一个 TCP 报文段从源主机传送到目的主机的最长时间。

3.4.4 TCP 的数据重传机制

为了提升数据传输的可靠性，TCP 采用了连续型 ARQ 协议。与 UDP 是面向报文的协议不同，TCP 是面向字节流的协议，即 TCP 不关心应用报文的边界，TCP 根据对方给出的窗口值和当前网络拥塞的程度来决定一个报文段应包含多少字节(UDP 发送的报文长度是应用进程给出的)。此外，在数据传输过程中，TCP 也是进行逐字节的传输确认，确保数据传输不重复、不丢失、不出错、按序到达。

由图 3-10 给出的 TCP 报文数据格式中可以看到，在报文中有两个字段，一个是序列号，另一个是确认号，这两个字段就是在 TCP 传输和确认消息中的字节编号信息。当然，对于确认号，只有在 TCP 报文中标识符 ACK 置位时才有效。对于 TCP 报文中序列号和确认号的使用如下。

(1) TCP 会对来自应用层的报文进行逐字节编号，并且会将本次 TCP 分段中首字节的编号填入 TCP 报文中的序列号字段。

(2) 如果接收端正确接收 TCP 报文，则会发送确认消息，会将 ACK 标识符置位，并且此时的确认号应该为所收到数据分段的最后一字节的下一字节，即预计要收到的下一数据分段的首字节编号。例如，当前接收端当前收到的 TCP 数据分段的序列号为 1，该数据分段的长度为 100，则接收端应该确认的是该数据分段从 1 到 100 字节均已正确接收，但是在给发送端的确认消息中的确认号应该为 101，表示前 100 字节已正确接收到，期待接收到的下一数据分段的首字节编号为 101。通过这样的方式，既可以完成对正确接收报文的确认，又能同时指示下一数据分段的首字节。

TCP 在发送端维持一个发送窗口，使得发送数据时可以连续进行发送，但在接收端，TCP 采用的是累积确认机制。“累积确认”在这里有两方面的含义：一方面，TCP 仅对按序接收到的能确认的最小编号进行确认；另一方面，接收端可以一次性确认多个连续的数据分段，而不是对每个数据分段都单独发送确认。TCP 累积确认机制过程如图 3-14 所示。

由图 3-14 中可以看出，服务器 B 在收到两个数据分段后，统一进行确认消息回复，根据累计累积确认规则，当前可以确认的是编号为 120 前的 119 字节都已经接收到了，期望接收到的下一个数据分段的编号为 120，所以 ACK 中确认号为 120。发送端收到该消息后，虽

然已经发送了两个数据分组，但从确认消息中读取确认号后，知道发送的两个数据分组均已经被正确接收，继续发送下一个数据分组，并且该分组的编号为120。

可以看出，通过累计确认机制，TCP可以提高传输效率和可靠性。发送方可以根据接收方的确认号来调整发送速率，避免网络拥塞。同时，接收方可以一次性确认多个数据包，减少确认的开销，提高传输效率。

TCP中一般有两种情况触发重传，一种是超时重传，另一种是快速重传。超时重传即数据分组发送后在规定的时间没有收到确认消息，发送端自动进行数据分段重传。需要注意的是，TCP中仅为当前尚未确认的编号最小的数据分段保留一个计时器，超时重传的示例如图3-15所示。

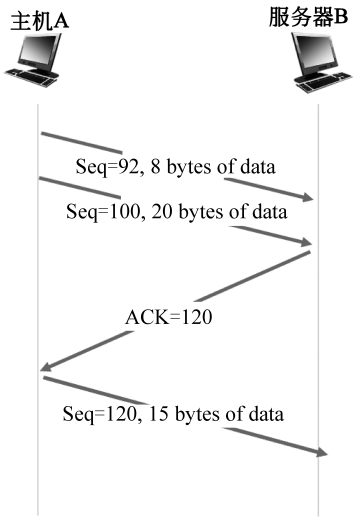


图3-14 TCP累积确认机制过程

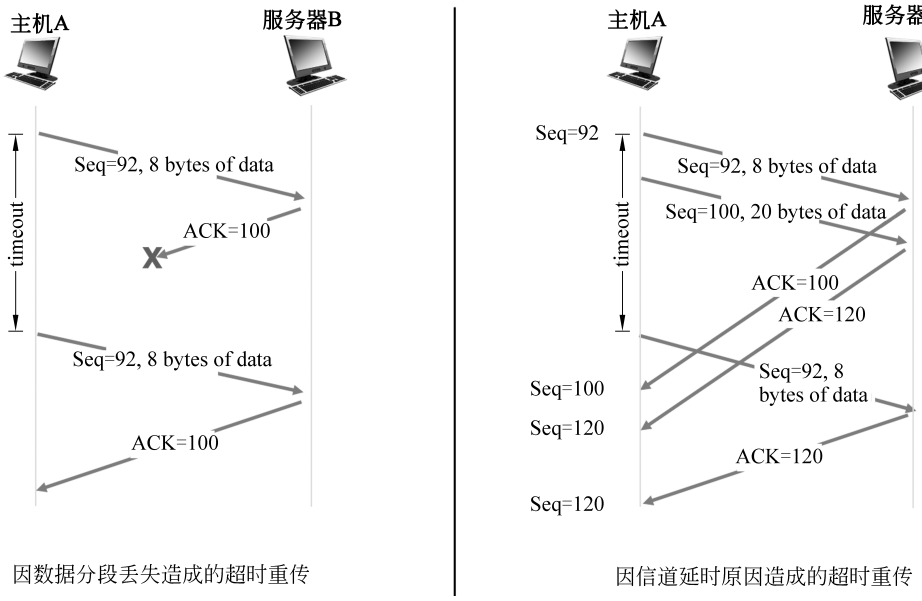


图3-15 TCP中超时重传的示例

如图3-15所示，该图中给出了两个超时重传的示例，其中第一个因为数据分段的确认消息在传输过程中丢失，导致主机A无法收到确认消息，超时后重新传输。在另外一个例子中，则是由于信道造成的延时，导致在计时器规定时间内没有收到数据分段的确认消息，重传后，由于服务器B已经正确接收到了编号为120之前的数据分段，因此在收到编号为92的数据分段后，回复的确认消息中确认号仍然为120，使得发送端可以发送编号为120及以后的数据分段。

触发TCP重传的另一种情况是快速重传，TCP规定当发送端收到三次重复的确认信息，即三次ACK的确认号相同，则不用等待超时，直接重传确认信息中确认号指定的数据