计算机网络复习整理（第五章）

###### 第五章 运输层

本章重要内容：

1. 运输层为相互通信的应用进程提供逻辑通信。
2. 端口和套接字的意义。
3. 无连接的UDP的特点。
4. 面向连接的TCP的特点。
5. 在不可靠网络上实现可靠传输的工作原理，停止等待协议和ARQ协议。
6. TCP的滑动窗口、流量控制、拥塞控制和连接管理。

5.1运输层协议概述

从通信和信息处理的角度看，**运输层向它上面的应用层提供通信服务**，它属于面向通信部分的最高层，同时也是用户功能中的最底层。当网络边缘的两台主机利用网络的核心功能进行端到端的通信时，只有主机的协议栈才有运输层，而路由器在转发分组时只用到下三层的功能。

从IP层来说通信的两端是两台主机，IP数据报的首部明确标志了这两台主机的IP地址。但真正进行通信的实体是主机中的进程，因此严格地讲，两台主机进行通信就是两台主机中的**应用进程互相通信**。IP协议虽然能把分组送到目的主机，但是这个分组还停留在主机的网络层而没有交付主机中的应用进程。从运输层的角度看，**通信的真正端点并不是主机而是主机中的进程**。也就是说，**端到端的通信**是应用进程之间的通信。

运输层有一个很重要的功能——**复用**和**分用**。复用是指在发送方不同的应用进程都可以使用同一个运输层协议传送数据（加上适当的首部），而分用是指接收方的运输层在剥去报文的首部后能够把这些数据正确交付目的应用进程。**运输层提供应用进程之间的逻辑通信**，逻辑通信的意思是：从应用层来看，只要把应用层报文交给下面的运输层，运输层就可以把报文送到对方的运输层，**好像这种通信就是沿水平方向直接传送数据**。**但事实上这两个运输层之间并没有一条水平方向的物理连接**，**数据传送是经过多个层次传送的**。

**网络层为主机之间提供逻辑通信**，**而运输层为应用进程之间提供逻辑通信**。

运输层还要对收到的报文进行**差错检测**，在网络层，IP数据报首部中的检验和字段，只检验首部是否出现差错而不检查数据部分。

根据应用程序的不同需求，运输层需要两种协议，即**面向连接的TCP**和**无连接的UDP**。

**运输层向高层用户屏蔽了下面网络核心的细节**（网络拓扑、所采用的路由选择协议等），**它使应用进程看见的就好像是在两个运输层实体之间有一条端到端的逻辑通信信道**，但这条逻辑通信信道对上层的表现因运输层的不同协议而有很大差别。当运输层采用TCP时，**尽管下面的网络是不可靠的**，但这种逻辑通信信道就相当于一条**全双工的可靠信道**。但运输层采用UDP时，这种逻辑通信信道仍然是**不可靠的**。

按照OSI的术语，两个对等层实体在通信时传送的数据单位叫**运输协议数据单元TPDU。**但在TCP/IP体系中，根据所选协议分别称为**TCP报文段**（segment）或**UDP用户数据报**。

UDP在传送数据前**不需要先建立连接**，远地主机运输层收到报文后不需要给出任何确认。

TCP**提供面向连接的服务**，在传送前必须建立连接，传送后要释放连接。TCP不提供广播或多播服务。

使用TCP和UDP的各种应用和应用层协议：

UDP：名字转换 DNS、文件传送TFTP、路由选择协议IP、IP地址配送DHCP、网络管理SNMP、远程文件服务器NFS、IP电话 专用协议、流式多媒体通信 专用协议、多播 IGMP

TCP：电子邮件 SNMP、远程终端接入TELNET、万维网WWW、文件传送 FTP

应用层的所有应用进程都可以通过运输层再传送到IP层（网络层），这就是**复用**。运输层从IP层收到发送给各应用进程的数据后，必须分别交付指明的各应用进程，这就是**分用**。

在运输层使用**协议端口号**，或称**端口**。虽然通信的终点是应用进程，但只要把传送的报文交到目的主机的某个合适的目的端口，剩下的工作（即交付目的进程）就由TCP或UDP完成。

这种在协议栈层间的端口是**软件端口**，和路由器或交换机上的**硬件端口**不同。硬件端口是**不同硬件设备**进行交互的接口，而软件端口是**应用层的各种协议进程与运输实体进行层间交互的一种地址**。

TCP和UDP的首部格式中都有**源端口**和**目的端口**，当运输层收到IP层上交的运输层报文时，就能根据其首部中的目的端口号把数据交付应用层的目的应用进程。

TCP/IP的运输层使用16位**端口号**来标志一个端口，端口号**只具有本地意义**，只是为了标志**本计算机**应用层中的各个进程在和运输层交互时的层间接口。在互联网不同计算机中，相同的端口号**没有关联**，16位端口号允许65535个不同的端口号。

两个计算机中的进程要相互通信，不仅要知道对方的IP地址，还要知道对方的端口号。互联网上的计算机通信采用客户-服务器方式，运输层端口号分为以下两种：

1. **服务器端口号** 最重要的一类叫做**熟知端口号**或**系统端口号**，数值0-1023。如：FTP21 、TELNET23、SMTP25、DNS53、TFTP69、HTTP80、SNMP161、SNMP(trap)162、HTTPS443。

另一类叫做**登记端口号**，数值1024-49151，为没有熟知端口号的应用进程使用的。

（2）**客户端端口号** 数值为49152-65535，又叫做**短暂端口号**。

5.2 用户数据报协议UDP

UDP只在IP的数据报服务上增加了很少一点的功能，复用、分用以及差错检测。UDP的主要特点是：

1. **无连接**，发送数据前不需要建立连接，减小了开销和发送时延。
2. **尽最大努力交付，**不保证可靠服务，主机不需要维持复杂的连接状态表。
3. UDP是**面向报文的**，UDP对应用层交下来的报文，既不合并也不拆分，而是保留这些报文的边界。因此应用程序需要选择合适大小的报文，太长（需要分片）太短（首部相对长度太大）都会降低IP等的效率。
4. UDP**没有拥塞控制**，网络出现拥塞不会使源主机的发送速率降低，适合很多实时应用（IP电话，视频会议等），这些应用要求源主机以恒定速率发送数据。
5. **支持一对一、一对多、多对一和多对多的交互通信**。
6. **首部开销小**，只有8字节，TCP需要20字节。

不使用拥塞控制的UDP有可能使网络产生严重拥塞。

UDP的首部格式

UDP有两个字段，数据字段和首部字段。首部字段只有8字节，由四个字段组成，每个字端都是2字节长。各字段意义如下：

1. **源端口** 源端口号，需要对方回信时使用，不需要全0。
2. **目的端口** 目的端口号，在终点交付报文时必须使用。
3. **长度** UDP用户数据报的长度，最小值为8（仅有首部）。
4. **检验和** 检测UDP数据报在传输中是否有错，有错则丢弃。（使用12字节的伪首部，既不向下传送也不向上递交）

如果接收方UDP发现接收报文中目的端口号不正确，即不存在对应于该端口号的应用进程，就丢弃该报文，并由网际控制报文协议ICMP发送端口不可达差错报文给发送方。ICMP的traceroute应用就是让发送的UDP数据报故意使用非法UDP端口。

5.3 传输控制协议TCP概述

TCP最主要的特点：

1. TCP是**面向连接**的运输层协议。
2. 每一条TCP连接只能有两个**端点**，只能是**点对点**的（一对一）。
3. TCP提供**可靠交付**的服务，通过TCP传送的数据，无差错、不丢失、不重复并且按序到达。
4. 提供**全双工通信**，TCP两端都有发送缓存和接收缓存，用来临时存放双向通信的数据。
5. **面向字节流**，流指的是流入到进程或从进程中流出的字节序列。面向字节流的含义是，虽然应用进程和TCP的交互是一次一个数据块，但TCP把应用程序交下来的数据仅仅看成一连串的**无结构字节流**。TCP不保证接收方应用程序收到的数据块和发送方应用程序发出的数据块具有对应大小的关系，但接收方收到的字节流必须和发送的字节流完全一样。

TCP连接是一条**虚连接**（**逻辑连**接），而不是一条真正的物理连接。TCP报文端先要传送到IP层，加上IP首部后，再传送到数据链路层。再加上数据链路层的首部和尾部再离开主机发送到物理链路。

TCP发送报文时不关心应用进程一次把多长的报文发送到TCP的缓存中，而是根据对方给出的窗口值和当前网络拥塞的程度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）。如果应用进程发送到TCP缓存的数据块太长，TCP就把它划分短一些再传送，如果太短TCP也可以等待积累足够多的字节构成报文段再发送。

TCP把**连接**作为**最基本的抽象**，TCP的许多特性都与TCP是面向连接的这个基本特性有关。

TCP连接有两个端点，TCP连接的端点叫做**套接字（socket）**或**插口**。端口号**拼接到**IP地址即构成了套接字。因此套接字的表示方法是在点分十进制的IP地址后写上端口号，中间用冒号或逗号隔开。**每一条TCP连接唯一地被通信两端的两个端点（即两个套接字）确定**。

一定要记住：**TCP连接的端点是个很抽象的套接字**，即（IP地址：端口号）。还应记住：同一个IP地址可以有多个不同的TCP连接，而同一个端口号也可以出现在多个不同的TCP连接。

5.4 可靠传输的工作原理

TCP下面的网络所提供的是不可靠的传输，因此TCP必须采用适当的措施才能使两个运输层之间的通信变得可靠。

停止等待协议

全双工通信的双方既是发送方也是接收方。停止等待协议就是每发送完一个分组就停止发送，等待对方的确认，在收到确认之后再发送下一个分组。

**1 无差错情况**

A发送分组M1，发送完后就暂停发送，等待B的确认。B收到M1就向A发送确认，A收到了对M1的确认后就发送M2，同样在收到B对M2的确认后再发送M3。

**2 出现差错**

B收到M1时检测出来差错就丢弃M1，其他什么也不做（不通知A收到差错分组），也可能是在传输过程中丢失了。这两种情况下B都不会发送任何信息，A只要超过了一段时间仍然没有收到确认，就认为之前发送的分组丢失了，因而重传前面发送的分组，这就叫**超时重传**。因此要在每发送完一个分组时设置一个**超时计时器**，如果在计时器到期前收到确认就撤销计时器。

需要注意三点：

1. A发送完分组后，必须暂时保留已发送分组的副本。只有收到确认后才清除。
2. 分组和确认分组都必须进行编号。
3. 超时计时器设置的重传时间应当比数据在分组传输的平均往返时间更长一些。如果设置的太长就会降低通信效率，如果太短就会导致不必要的重传。

**3 确认丢失和确认迟到**

假定B发送的确认丢失了，A在设定超时重传时间内没收到确认，重传了M1，假定B又收到了重传分组，此时B应采取两个行动。第一，**丢弃重复分组**M1，不向上层交付。第二，向A**发送确认**。

当B的确认迟到时，A会收到重复的确认。对此的处理很简单，A收下后就丢弃。B仍然会收到重复的分组M1，并且同样要丢弃掉重复分组，并重传确认分组。

通常A最终总是可以收到对发出所有分组的确认，如果A不断重传分组但总是收不到确认，说明通信线路太差，不能通信。

使用上述的确认和重传机制就可以**在不可靠的传输网络实现可靠通信**。上述的这种可靠传输协议常称为**自动重传请求ARQ**，意思是重传的请求是自动进行的。接收方不需要请求发送方重传某个出错的分组。

**4 信道利用率**

停止等待协议的优点是简单，确点是信道利用率太低。

信道利用率U=TD/（TD+RTT+TA）

RTT：往返时间 TD:A发送分组的时间 TA :B发送确认分组时间。

当RTT远大于分组发送时间时，信道利用率会非常低。为提高传输效率，发送方可以不使用低效率的停止等待协议，而采用**流水线传输**，流水线传输就是发送方可连续发送多个分组，不必发送完每个分组就停顿下来等待确认。当使用流水线传输时，就要使用**连续ARQ协议**和**滑动窗口协议**。

连续ARQ协议

连续ARQ协议规定，发送方每收到一个确认，就把发送窗口向前滑动一个分组的位置。

接收方一般都是采用**累积确认**的方式，接收方不必对收到的分组逐个发送确认，而是在收到几个分组后，**对按序到达的最后一个分组发送确认**，这就表示，到这个分组为止的所有分组都已经正确收到了。

累积确认的优点：容易实现，即使确认丢失也不必重传。缺点：不能向发送方反映出接收方已经正确收到的所有分组信息。

例如发送方发送了前5个分组，而第3个分组丢失了。这时接收方只能对前2个分组进行确认，发送方无法知道后面3个分组的下落，只好把后面3个分组再重传一次。这就叫做回退N，表示需要再退回来重传已发送的N个分组。因此通信线路质量不好时，连续ARQ协议会带来负面影响。

5.5 TCP报文段的首部格式

TCP虽然面向字节流，但传送的数据单元是报文段。一个TCP报文段分为首部和数据两部分。

TCP报文段首部20字节是固定的，后面有4n字节是根据需要而增加的选项，因此TCP首部的最小长度是20字节。

首部固定部分各字段意义如下：

1. **源端口**和**目的端口** 各占2字节，分别写入源端口号和目的端口号，TCP的分用和UDP相似，也是通过端口实现。
2. **序号** 4字节，范围是[0，232-1]。在一个TCP连接中传送的字节流中的**每一个字节都按顺序编号**，整个要传送的字节流的起始序号必须在连接建立时设置。首部中的序号字段值则是**本报文段**所发送的数据的第一个字节的序号。这个字段的名称也叫**报文段序号**。
3. **确认号** 4字节，是**指期望收到对方下一个报文段的第一个数据字节的序号**。若确认号=N，表明到序号N-1为止的所有数据都已正确收到。一般情况下可保证当序号重复使用时旧序号的数据早已通过网络到达终点了。
4. **数据偏移** 4位，指出TCP报文段的数据起始处距离TCP报文段的起始处有多远。
5. **保留** 6位，目前置为0。
6. **紧急URG** 值为1时，表明有效。当URG置1时，发送应用进程就告诉发送方的TCP有紧急数据要传送，于是发送方TCP就把紧急数据插入到本报文段数据的**最前面**，而在紧急数据后面的数据仍是普通数据。这时要与首部中**紧急指针字段**配合使用。
7. **确认ACK** 仅当ACK=1时确认字段才有效。TCP规定在连接建立后所有传送的报文字段都必须把ACK置1。
8. **推送PSH** 当两个应用进程进行交互式通信时，有时在一端的应用进程希望在键入一个命令后立即就能够收到对方的响应，这种情况下TCP就可以使用推送操作。
9. **复位RST** 当RST=1时，表明TCP连接中出现严重差错，必须释放连接，然后再重新建立运输连接。RST置1还用来拒绝一个非法的报文段或拒绝打开一个连接。
10. **同步SYN** 在连接建立时用来同步序号。当SYN=1而ACK=0时，表明这是一个连接请求报文段。对方若同意建立连接，则在响应的报文段中使SYN=1和ACK=1。因此SYN=1表示这是一个连接请求或连接接受报文。
11. **终止FIN** 用来释放一个连接，当FIN=1时，表明此报文段的发送方的数据已发送完毕，并要求释放运输连接。
12. **窗口** 2字节，值为[0，216-1]之间的整数，指的是发送本报文段的一方的接收窗口。窗口值告诉对方，从本报文段首部中的确认号算起，接收方目前允许对方发送的数据量。
13. **检验和** 2字节，检验的范围包括首部和数据两部分。计算检验和时要在TCP报文段的前面加上12个字节的伪首部。
14. **紧急指针** 2字节，仅在URG=1时才有效，指出本报文段中的紧急数据字节数。
15. **选项** 长度可变，最长40字节。TCP起初只规定了一种选项，最大报文段长度MSS（数据字段的最大长度）。默认值536字节，为提高网络利用率。

5.6 TCP可靠传输的实现

TCP的滑动窗口是以字节为单位的。发送窗口里的序号表示允许发送的序号，显然窗口越大，发送方就可以在收到对方确认之前连续发送更多的数据，提高传输效率。发送窗口的大小不能超过对方的接收窗口大小。

发送窗口的后沿部分表示已发送并且收到确认的序号，前沿部分表示不允许发送的序号。发送窗口后沿的变化情况有两种，不动（没有收到新的确认）和前移（收到了新的确认）。发送窗口的前沿通常不断向前移动，但也可能不动或向后收缩。

5.7 TCP的流量控制

所谓流量控制，就是让发送方的发送速率不要太快，要让接收方来得及接收。

发送方的发送窗口不能超过接收方给出的接收窗口的数值。**TCP窗口的单位是字节**，**不是报文段**。

B向A发送的接收窗口大小报文可能丢失，A一直等待B发送的非零窗口通知，B也在一直等待A发送的数据，如果没有其他措施，可能会形成死锁。

为了解决这个问题，TCP为每一个连接设有一个**持续计时器**，只要TCP连接的一方收到对方的零窗口通知，就启动持续计时器，若持续计时器的时间到了，就发送一个零窗口**探测报文段**（仅携带1字节数据），而对方就在确认这个探测报文段时给出了现在的窗口值。如果窗口仍然是0，那么收到这个报文段的一方就重新设置持续计时器。如果窗口不是0，那么死锁的僵局就打破了。

应用进程把数据发送到TCP的发送缓存后，剩下的发送任务就交给TCP来控制了。可以用不同的机制来控制TCP报文段的发送时机，例如第一种机制是TCP维持一个变量，它等于最大报文段长度MSS，只要缓存中数据到达此长度就组成一个TCP报文段发送出去。第二种机制是由发送方的应用进程指明要求发送报文段，即TCP支持的推送操作。第三种机制是发送方的一个计时器时限到了，这就把当前已有的缓存数据装入报文段（长度不超过MSS）发送出去。

5.8 TCP的拥塞控制

拥塞控制的一般原理

某段时间内，若对网络中某一资源的需求超过了该资源所能提供的可用部分，网络的性能就要变坏。这种情况就叫做**拥塞**。

网络拥塞的原因很多。例如某个结点缓存的容量太小，到达该结点的分组因无存储空间暂存而不得不丢弃。假设将该结点缓存的容量扩展到非常大，于是凡到达该结点的分组均可在结点的缓存队列中排队，不受任何限制。由于输出链路的容量和处理机的速度并未提高，因此队列中的绝大多数分组的排队等待时间将会大大增加，结果上层软件只好把它们重传（超时）。因此简单扩大缓存空间不能解决网络拥塞。

拥塞常常趋于恶化，拥塞引起的重传不会缓解网络拥塞，反而会加剧。

**拥塞控制**就是**防止过多的的数据注入到网络**，**这样可以使网络中的路由器或链路不致过载**。拥塞控制要做的都有一个前提，就是**网络能够承受现有的网络载荷**。拥塞控制是一个全局性过程，涉及到所有主机，所有的路由器，以及与降低网络传输性能有关的所有因素。TCP连接的端点只要迟迟不能收到对方的确认信息，就猜想当前网络的某处可能发生了拥塞，但发生的地点和原因不从得知。

相反，流量控制是指**点对点通信的控制**，是个**端到端**的问题（接收端控制发送端）。流量控制所要做的就是抑制发送端发送数据的速率以便接收端来得及接收。

理想情况下，随着**输入负载**（单位时间内输入给网络的分组数目）或**网络负载**的增大，**吞吐量**应等于提供的负载，当吞吐量饱和时，理想拥塞控制的情况下，吞吐量仍然可以维持最大值。但实际上随着负载增大，网络吞吐量的增长速率逐渐减小，当网络的吞吐量明显小于理想吞吐量时，网络就进入了**轻度拥塞**的状态。当提供负载到达某一数值时，网络的吞吐量反而随着负载的增大而下降，这时网络就进入了**拥塞状态**。当负载继续增大时，网络的吞吐量就下降到0，网络无法工作，这就是所谓的**死锁**。

拥塞控制是一个动态的问题，从大的方面可以将拥塞控制分为**开环控制**和**闭环控制**两种方法。开环控制就是在设计网络时事先将有关发生拥塞的因素考虑周到，力求网络工作时不产生拥塞，但系统一旦运行就不在途中进行改正了。

闭环控制是基于反馈环路的概念，主要措施：

1. 监测网络系统以便检测到拥塞在何时、何处发生
2. 把拥塞发生的信息传送到可采取行动的地方
3. 调制网络系统的运行以解决出现的问题

TCP拥塞控制的算法有四种：（1）**慢开始**（2）**拥塞避免**（3）**快重传**（4）**快恢复**。

5.9 TCP的运输连接管理

运输连接有三个阶段：**连接建立**、**数据传送**、**连接释放**

TCP连接建立中要解决以下三个问题：

1. 要使每一方能够确知对方的存在
2. 要允许双方协商一些参数
3. 能够对运输实体资源进行分配

TCP连接的建立采用客户服务器方式，主动发起连接建立的应用进程叫做**客户**，而被动等待连接建立的应用进程叫做**服务器**。

TCP建立连接的过程叫做握手，握手需要在客户和服务器之间交换三个TCP报文段。

最初两端的TCP进程都处于CLOSED状态，假设**A主动打开连接**，**B被动打开连接**。

一开始B的TCP服务器进程先创建**传输控制块TCB**，准备接收客户进程的连接请求。然后服务器进程就处于LISTEN状态，等待客户的连接请求。如有则立即响应。

A的TCP客户进程也是首先创建**传输控制块TCB**，然后打算建立TCP连接时，先向B发出连接请求报文段，SYN=1，同时选择一个初始序号seq=x。TCP规定SYN报文段不能携带数据，但要消耗掉一个序号，这时TCP客户进程进入SYN-SENT（同步已发送）状态。

B收到连接请求报文段后，如同意建立连接则向A确认。在确认报文段应把SYN和ACK位都置1，确认号是ack+1，同时也为自己选择一个初始序号seq=y。这个报文段也不能携带数据，但同样**要消耗一个序号**。这时TCP服务器进程进入SYN-RCVD（同步收到）状态。

TCP客户进程收到B的确认后，还要向B给出确认。确认段ACK=1，确认号ack=y+1，而自己的序号seq=x+1。TCP规定ACK的报文段可以携带数据，如果不携带数据则不消耗序号，这种情况下下一个报文段的序号仍是seq=x+1，这时TCP连接已建立，A进入ESTABLISHED（已建立连接）状态。

B收到A的确认后，也进入ESTABLISHED状态。

为什么A要发送一次确认呢，主要是为了防止已失效的连接请求报文段突然又传送到了B，因而产生错误。假设A发出的连接请求报文在网络某些结点滞留了，以至于连接释放后的某个时间才到达B。本来这是失效报文段，但B收到后误认为是A发出的新连接请求，就发出确认。假设不采取报文握手，只要B发出确认，连接就建立了，由于A并没有发出请求因此不会理睬B，但B误认为连接已建立会一直等待A发送数据，白白浪费B中的资源。

TCP的连接释放

A和B都处于ESTABLISHED状态，A的应用进程先向其TCP发出释放连接报文段，并停止再发送数据，主动关闭TCP连接。A把连接释放报文段首部的终止控制位FIN置1，seq=u，它等于前面已传送的最后一个字节的序号加1。这时A进入FIN-WAIT-1（终止等待1）的状态，等待B的确认。FIN报文段不携带数据也要消耗掉一个序号。

B收到连接释放报文段后发出确认，确认号是ack=u+1，而这个报文段自己的序号是v，等于前面B已传送的最后一个字节的序号加1。然后B就进入CLOSE-WAIT（关闭等待）状态。这时A到B这个方向的连接就释放了，这时TCP的连接处于**半关闭**状态，即A已经没有数据要发送了，但若B发送数据，A仍要接受。

A收到来自B的确认后，就进入FIN-WAIT-2（终止等待2）状态，等待B发出的连接释放请求报文。若B已经没有要向A发送的数据，其应用进程就通知TCP释放连接，这时B发出的连接释放报文段必须使FIN=1。假定B的序号为w（半关闭状态时B可能又发送了一些数据）。B还必须重复上次已发送过的确认号ack=u+1，这时B就进入了LAST-ACK（最后确认）状态。

A收到B的连接释放报文段后必须对此确认，在确认报文段中把ACK置1，确认号ack=w+1，自己的序号是u+1（前面发送的FIN报文消耗一个序号）。然后进入TIME-WAIT状态。必须经过时间等待计时器设置的时间2MSL后，A才能进入到CLOSED状态。MSL叫做最长报文段寿命。

A必须等待2MSL时间的理由：

1. 为了保证A发送的最后一个ACK报文段能够到达B。这个ACK报文段有可能丢失，B会超时重传这个FIN+ACK报文段，而A就能在2MSL时间内收到这个报文段，接着A再重传一次确认，重新启动2MSL计时器。最后AB都正常进入到CLOSED状态。如果A不等待一段时间就收不到B重传的报文段，也不会再发送一次确认报文段，B就无法按照正常步骤进入CLOSED状态。
2. 为了防止前述的“已失效连接请求报文段”出现在本连接中。

TCP还设有保活计时器，服务器每收到一次客户的数据就重新设置保活计时器，时间的设置通常是2小时，若2小时内没有收到客户的数据，就发送一个探测报文段，以后每隔75秒发送一次，如果连续10个探测报文段后仍无客户的响应，服务器就认为客户端出了故障，关闭连接。

5.10 本章的重要概念

运输层提供应用进程间的逻辑通信，运输层的通信并不是真正在两个运输层之间直接传送数据。运输层向应用层屏蔽了下面的网络细节（网络拓扑、路由选择协议等），它使应用进程看见的就好像在两个运输层实体之间有一条端到端的逻辑通信线路。

网络层为主机之间提供逻辑通信，而运输层为应用进程之间提供端到端的逻辑通信。

运输层有两个主要的协议：TCP和UDP。它们都有复用和分用，以及检错的功能。当运输层采用TCP协议时，尽管下面的网络是不可靠的，但这种逻辑通信信道就相当于一条全双工通信的可靠信道。当运输层采用UDP协议时，这种逻辑通信信道仍然是一条不可靠信道。

运输层用一个16位端口号来标志一个端口，端口号具有本地意义，它只是为了标志本计算机应用层中的各个进程在和运输层交互时的层间接口。在互联网的不同计算机中，相同的端口号没有关联。

两台计算机中的进程要互相通信，不仅要知道对方IP地址，还要知道对方的端口号。

运输层的端口号分为服务器端端口号（0-1023指派给数值端口，1024-49151是登记端口号）和客户端暂时使用的端口号（49152-65535）。

UDP的主要特点：（1）无连接（2）尽最大努力交付（3）面向报文（4）无拥塞控制（5）支持一对一、一对多、多对一和多对多的交互通信（6）首部开销小（只有源端口、目的端口、长度、检验和）。

TCP的主要特点：（1）面向连接（2）每一条TCP连接只能是点对点的（3）提供可靠的交付服务（4）提供全双工通信（5）面向字节流

TCP用主机的IP地址加上主机的端口号作为TCP连接的端点，这样的端点叫做套接字或插口。套接字用（IP地址：端口号）来表示。

停止等待协议能够在不可靠的传输网络实现可靠通信。每发完一个分组就停止发送，等待对方的确认。在收到确认后再发送下一个分组，分组需要编号。

超时重传是只要超过了一段时间仍然没有收到确认，就重传前面发送过的分组。因此每发送完一个分组需要设置一个超时计时器，其重传时间应比数据在分组传输中的平均往返时间更长一些。这种自动重传方式常称为自动重传请求ARQ。

在停止等待协议中，若接收方收到重复分组，就丢弃该分组，但同时还要发送确认。

连续ARQ协议可提高信道利用率，发送方维持一个发送窗口，凡位于发送窗口内部的分组都可以连续发送，而不需要等待确认。接收方一般采用累积确认，对按序到达的最后一个分组发送确认，表明到这个分组为止的所有分组都已正确收到了。

TCP报文段首部的前20个字节是固定的，后面有4N字节是根据需要而增加的选项（N是整数）。在一个TCP连接中传送的字节流中的每一个字节都按顺序编号。首部中的序号字段值指的是本报文段发送的数据的第一个字节的序号。

TCP首部中的确认号是期望收到对方下一个报文段的第一个数据字节的序号，若确认号位N，则表明到序号N-1为止的所有数据都已正确收到。

TCP首部中窗口的字段指出了现在允许对方发送的数据量，窗口是动态变化的。

TCP使用窗口滑动机制，发送窗口里的序号表示允许发送的序号，发送窗口后沿的部分表示已发送并且收到确认的，前沿的前面部分表示不允许发送。发送窗口后沿的变化情况有两种，即不动（没有收到新的确认）和前移（收到了新的确认）。发送窗口前沿通常是不断向前移动的。

流量控制就是让发送方的速率不要太快，要让接收方来得及接收。

某段时间，若对网络中某一资源的需求超过了该资源所能提供的可用部分，网络的性能就要变坏。这种情况叫做拥塞。拥塞控制就是防止过多的数据注入到网络，这样可使网络中的路由器或链路不至于过载。

流量控制是一个端到端的问题，使接收端抑制发送端发送数据的速率，以便使接收端来得及接收。拥塞控制是一个全局性的过程，涉及到所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素。

为了进行拥塞控制，TCP的发送方要维持一个拥塞窗口cwnd的状态变量。拥塞窗口的大小取决于网络的拥塞程度，并且动态变化。发送方让自己的发送窗口取为拥塞窗口和接收方的接收窗口中较小的一个。

TCP的拥塞控制采取了四种算法，即慢开始、拥塞避免、快重传和快恢复。在网络层也可以使路由器采用适当的分组丢弃策略（如主动队列管理AQM），以减小网络拥塞的发生。

运输连接有三个阶段：连接建立、数据传送、连接释放。

主动发起TCP连接请求的应用进程叫客户，而被动等待连接建立的应用进程叫服务器。TCP的连接建立采用三报文握手机制，服务器要确认客户连接请求，然后客户要对服务器的确认进行确认。

TCP连接的释放采用四报文握手机制，任何一方都可以在数据传送结束后发出连接释放的通知，待对方确认后就进入半关闭状态。当另一方也没有数据再发送时，则发送连接释放通知，对方确认后就完全关闭了TCP连接。