Chapter 8
传输层

zibuyu

January 10, 2006

1 传输层服务

引入传输层原因 :

- 消除网络层不可靠性
- 提供从源端主机到目的端主机的可靠的、与实际使用的网络无关的信息传输.

传输层提供两种服务 :

- 面向连接的传输服务：连接建立，数据传输，连接释放;
- 无连接的传输服务.

定义 :

- 1~4层称为传输服务提供者(transport service provider);
- 4层以上称为传输服务用户(transport service user).

传输服务原语(Transport Service Primitives) : LISTEN, CONNECT, SEND, RECEIVE, DISCONNECT.

传输协议数据单元TPDU :Transport Protocol Data Unit.

拆除连接的两种方式 :

- 不对称方式:任何一方都可以关闭双向连接.
- 对称方式,每一方向的连接单独关闭,双方都执行DISCONNECT才能关闭整条连接.

2 传输层建立连接、数据传输、拆除连接的过程

服务程序 :创建socket=>bind=>listen=>accept=>send(receive)=>close.

客户程序 :创建socket=>connect=>receive(send)=>close.

3 寻址(Addressing)方法

传输服务访问点TSAP(Transport Service Access Point) :在Internet,TSAP为(IP address + local port).

远程客户程序如何获得服务程序的TSAP :

方法1 预先约定、广为人知的，象telnet是(IP地址，端口23);

方法2 从名字服务器（name server）或目录服务器（directory server）获得TSAP.

当服务程序很多时，使用初始连接协议（initial connection protocol） :一个称为进程服务器（process server）的进程(inetd)同时在多个端口上监听.

4 传输层建立连接和释放连接机制

4.1 建立连接

三次握手方案(three-way handshake) :

1. A发出初始序号为X的CR(CONNECTION REQUEST) TPDU;
2. B发出初始序号为Y的ACK TPDU,确认A的序号为X的CR TPDU;
3. A发送序号为X+1的数据TPDU,并确认B的序号为Y的CR TPDU.

4.2 释放连接

非对称释放 :存在丢失数据的危险.

对称释放 :由于两军问题(two-army problem),不存在安全的经过N次握手实现对称式释放连接的方法.

三次握手+定时器 :在绝大多数情况下成功.

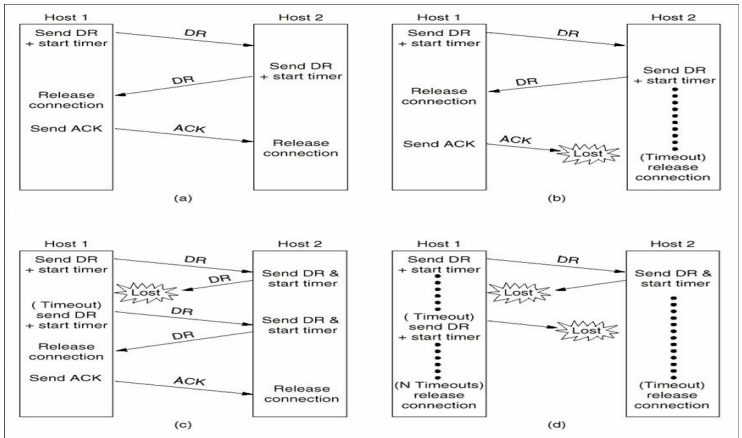

Fig. 6-14. Four protocol scenarios for releasing a connection. (a) Normal case of three-way handshake. (b) Final ACK lost. (c) Response lost. (d) Response lost and subsequent DRs lost.

4.3 流控和缓存(Flow Control and Buffering)

缓存 :网络层服务不可靠;传输层实体必须缓存所有连接发出的TPDU,便于错误情况下重传.三种缓冲区设计方式:

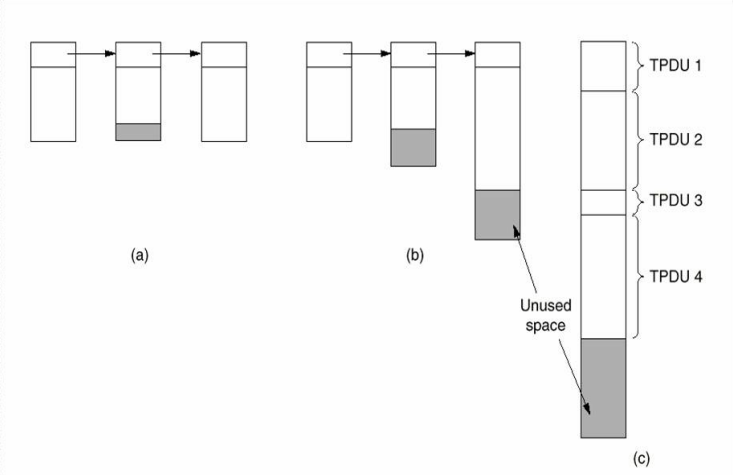

Fig. 6-15. (a) Chained fixed-size buffers. (b) Chained variable-size buffers. (c) One large circular buffer per connection.

流控 :利用可变滑动窗口协议实现流控.发送方的窗口根据接收方的实际缓存情况变化.

5 TCP & UDP

5.1 传输控制协议TCP(Transmission Control Protocol)

面向连接的、可靠的、端到端的、基于字节流的传输协议.

服务模型 :

- 应用程序访问TCP服务是通过在收发双方创建套接字来实现的;
- 套接字的地址是用（IP地址，主机端口号）来表示的。256以下的端口号被标准服务保留;
- 每条连接用(套接字1,套接字2)来表示，是点到点的全双工通道;
- TCP不支持多播（multicast）和广播（broadcast）;
- TCP连接是基于字节流(按字节分配序号)的，而非消息流，消息的边界在端到端的传输中不能得到保留;

解决问题 ：

- 可靠传输:滑动窗口;
- 流控和拥塞控制:
  - 可变滑动窗口;
  - 慢启动(slow start)和拥塞避免(congestion avoidance)
- 连接管理：
  - 建立连接：三次握手;
  - 释放连接：三次握手+定时器.

TCP头

- 按字节分配序号，每个字节有一个32位的序号;
- 传输实体之间使用段(segment)(TPDU)交换数据;
- 每个段包含一个20字节的头（选项部分另加）和0个或多个数据字节。段的大小必须首先满足65535字节的IP包数据净荷长度限制，还要满足数据链路层最大传输单元（MTU）的限制，比如以太网的MTU为1500字节;
- TCP实体使用滑动窗口协议，确认序号等于接收方希望接收的下一个序号,这与之前p3 p6不同;

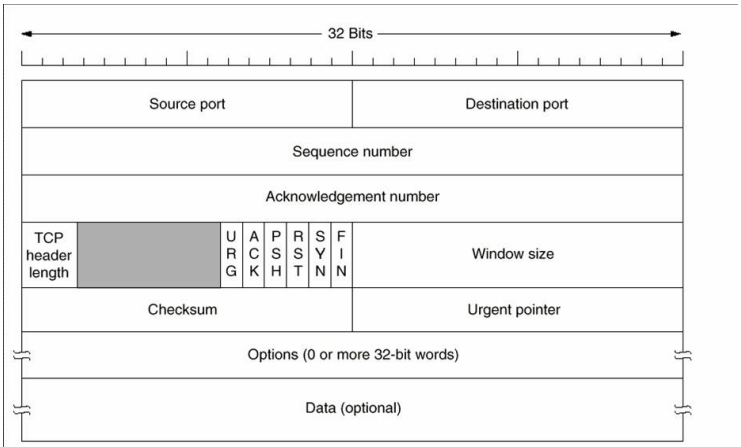

Fig. 6-24. The TCP header.

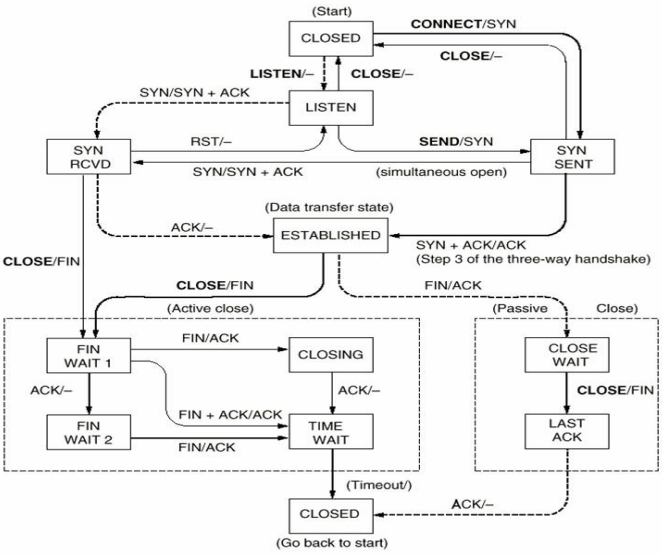

Fig. 6-28. TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events.

5.2 UDP(User Data Protocol)

无连接的、端到端的传输协议.

优势 :

- 不需要建立连接,延迟小;
- 简单,无连接状态;
- 报文头小;
- 无拥塞控制,可尽快发送.

应用 :

- 流媒体:容忍丢包,对速率敏感;
- RIP:路由信息周期发送;
- DNS:避免TCP连接建立延迟;
- SNMP
- 基于UDP的可靠传输:应用程序自己定义错误恢复.

6 TCP的窗口管理机制& 改进传输层性能的策略

6.1 TCP窗口管理机制

- 基于确认和可变窗口大小;
- 窗口大小为0时,正常情况下,发送方不能再发TCP段,但有两个例外
  - 紧急数据可以发送
  - 防止死锁,发送方可法送1字节TCP段,让接收方重新声明确认号与窗口大小.

6.2 改进传输层性能的策略

策略1 发送方缓存应用程序的数据，等到形成一个比较大的段再发出;

策略2 在没有可能进行“捎带”的情况下，接收方延迟发送确认段;

策略3 使用Nagle算法：当应用程序每次向传输实体发出一个字节时，传输实体发出第一个字节并缓存所有其后的字节直至收到对第一个字节的确认；然后将已缓存的所有字节组段发出并对再收到的字节缓存，直至收到下一个确认;

策略4 使用Clark算法解决傻窗口症状（silly window syndrome）:限制收方只有在具备一半的空缓存或最大段长的空缓存时，才产生一个窗口更新段.

7 慢启动和拥塞避免算法

出现拥塞的两种情况 ：

- 快网络小缓存接收者;
- 慢网络大缓存接收者;
- 导致网络拥塞的两个潜在因素是：网络能力和接收能力.

TCP处理第一种拥塞的措施 ：

- 在连接建立时声明最大可接受段长度;
- 利用可变滑动窗口协议防止出现拥塞.

TCP处理第二种拥塞的措施 ：

- 发送方维护两个窗口：可变发送窗口和拥塞窗口，按两个窗口的最小值发送;
- 拥塞窗口依照慢启动（slow start）算法和拥塞避免（congestion avoidance）算法变化.

7.1 慢启动(slow start)算法

- 接建立时拥塞窗口（congwin）初始值为该连接允许的最大数据段长，阈值（threshold）为64KB;
- 发出一个最大数据段长的TCP段，若正确确认，拥塞窗口变为两个最大段长;
- 发出(拥塞窗口/最大段长)个最大数据段长的TCP段，若都得到确认，则拥塞窗口加倍;
- 重复上一步，直至发生丢包超时事件，或拥塞窗口大于阈值;

7.2 拥塞避免(congestion avoidance)算法

- 若拥塞窗口大于阈值后，从此时开始，拥塞窗口线性增长，一个RTT周期增加一个最大段长，直至发生丢包超时事件;
- 当超时事件发生后，阈值设置为当前拥塞窗口大小的一半, 拥塞窗口重新设置为一个最大段长;
- 执行慢启动算法.

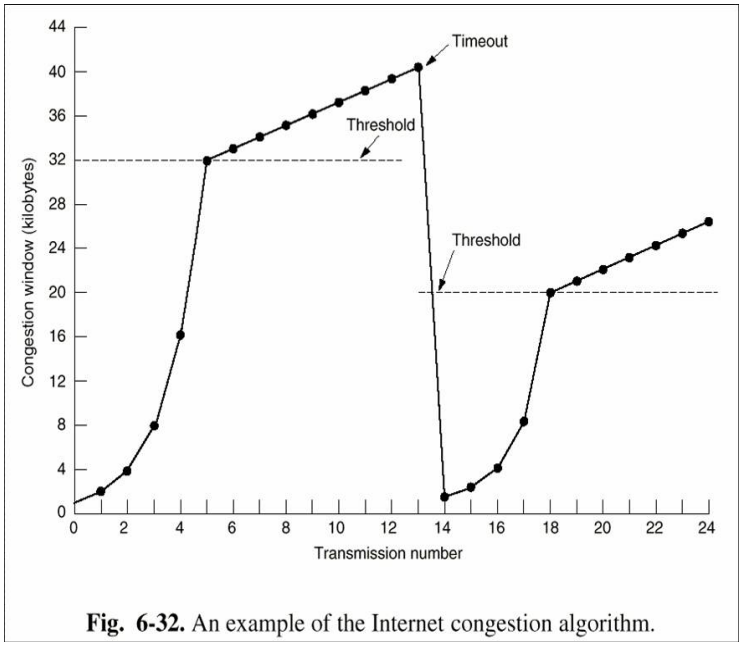

Fig. 6-32. An example of the Internet congestion algorithm.