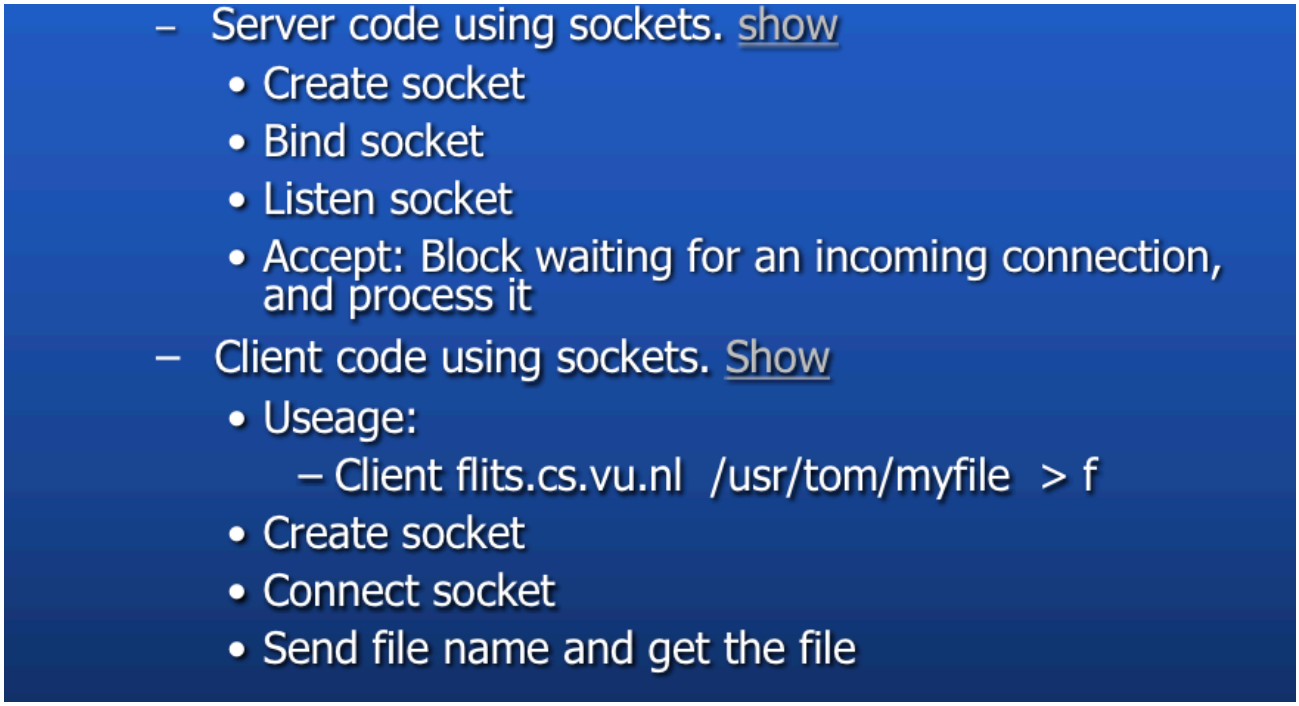
1 THE TRANSPORT SERVICE

1.1 Berkley Sockets

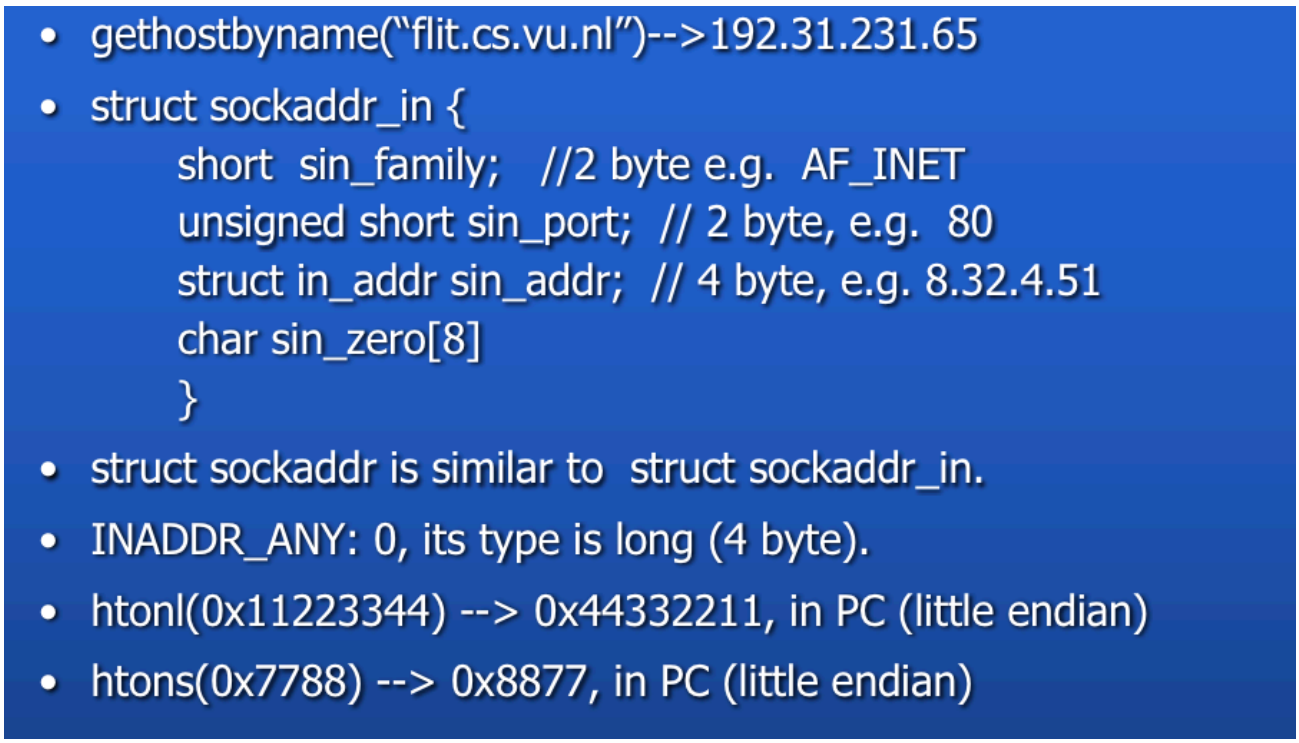
Primitive	Meaning
SOCKET	Create a new communication end pointCreate a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

- bind是创建socket后，将IP地址绑定于此，这样这个socket就是有名有姓，可以进行网络通讯

1.2 An Example of Socket Programming:An Internet File Server



- 有阻塞式和非阻塞式的
- 大概的创建形式过程就是上述的，具体代码可以看Chap6 12-15



- 有大端计算机和小端计算机之分，
- 小端计算机调用函数的时候要转换，大端计算机不用转换

2 ELEMENT OF TRANSPORT PROTOCOLS

2.1 Connection Establishment

- A complex process： the existence of delayed duplicates
- Solution ： three-way handshake-[see fig] show
  - To solve the problems of delayed duplicate control TPDUs and its duplicate ACK
- Two-way handshake will result in half-connection.
  - One side of a connection is open while the other side is close;
  - This will waste resource.

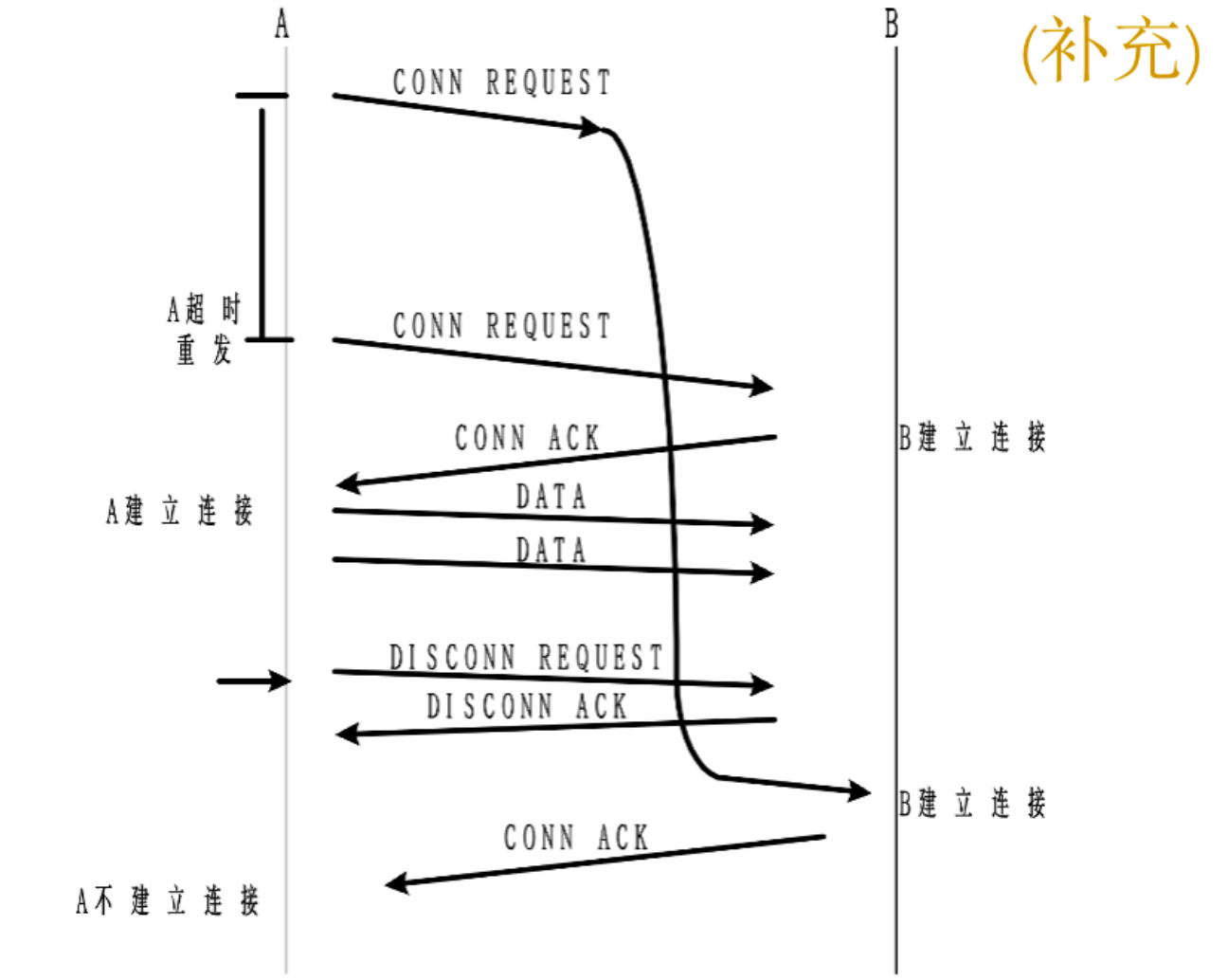


Fig K8-1 用两次握手建立TCP连接会产生问题

- 但是会出现问题：如果request传的很慢，A超时重发，b接收到两次request，A建立了两个连接，B处建立了两个连接（一个是全连接，一个是半连接，半连接没用，但会耗资源）
- 因此要三次连接
- 链路层是两次握手

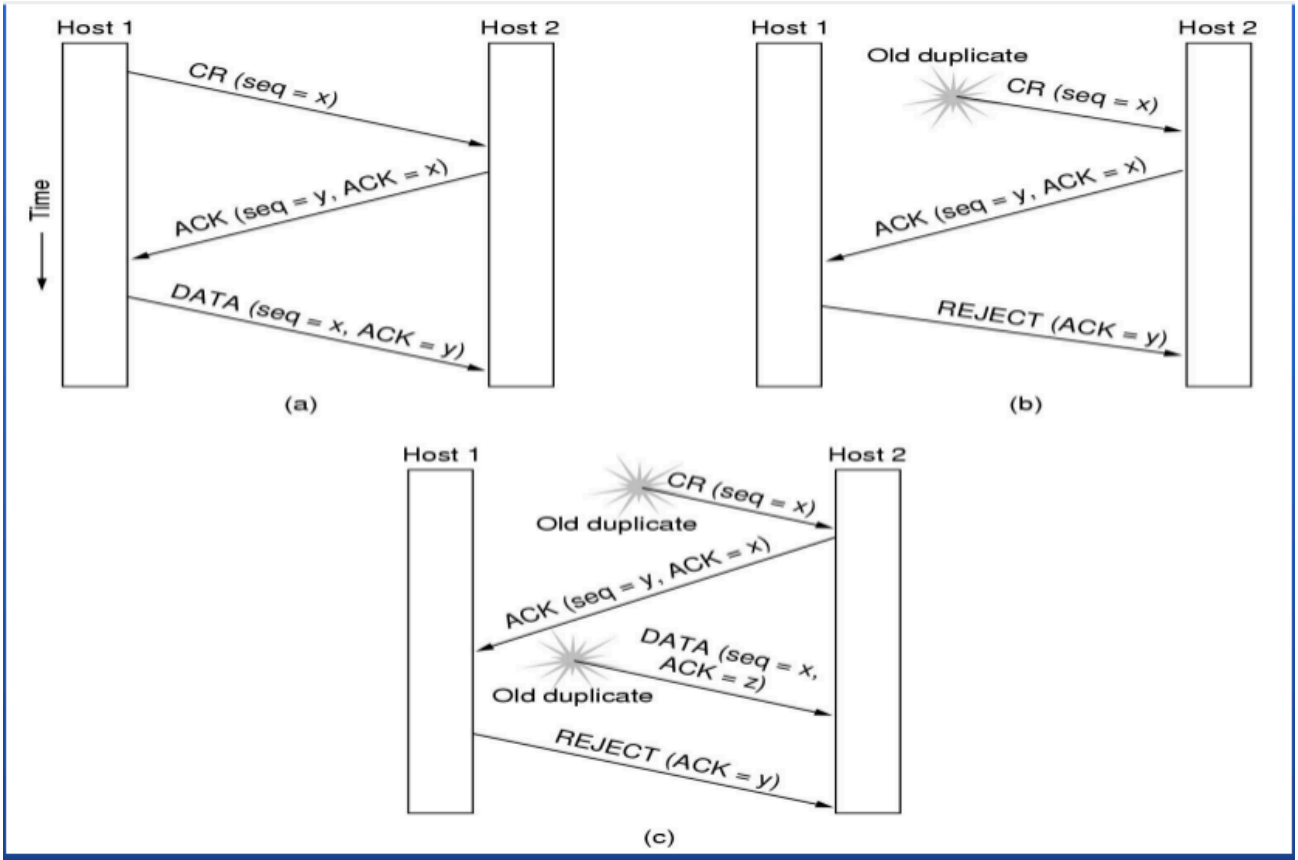


Fig. Three protocol scenarios for establishing a connection using a three-way handshake.CR denotes CONNECTION REQUEST.(a)Normal operation.(b)Old duplicate CONNECTION REQUESED appearing out of nowhere.©Duplicate CONNECTION REQUEST and duplicate ACK.

back

- 超时重发的时候发送的seq是一样的，服务器会把接收到的这个扔掉
- 可以理解成第三次多发了一次确认帧？？？这样即使哪里重发了，都会正常运行
- b图中的是旧的request发送，尽管主机2会发送一个ACK应答包，但是主机1 会拒绝；c图中是说如果有一个古早的requeset和data都发到了主机2，主机1同b图会进行丢弃，而主机2自己有记录发送报文段的序号，收到老的报文段的ACK会进行丢弃不予理睬，主机2收到主机1的reject，不会建立半连接。

2.2 Connection Release

- 断开连接上有两军问题：具体就是说建立一个100%的准确的释放连接是不可能的
- 实际上，三次握手的方式也在实际中运用，加上约束条件，设定丢包率

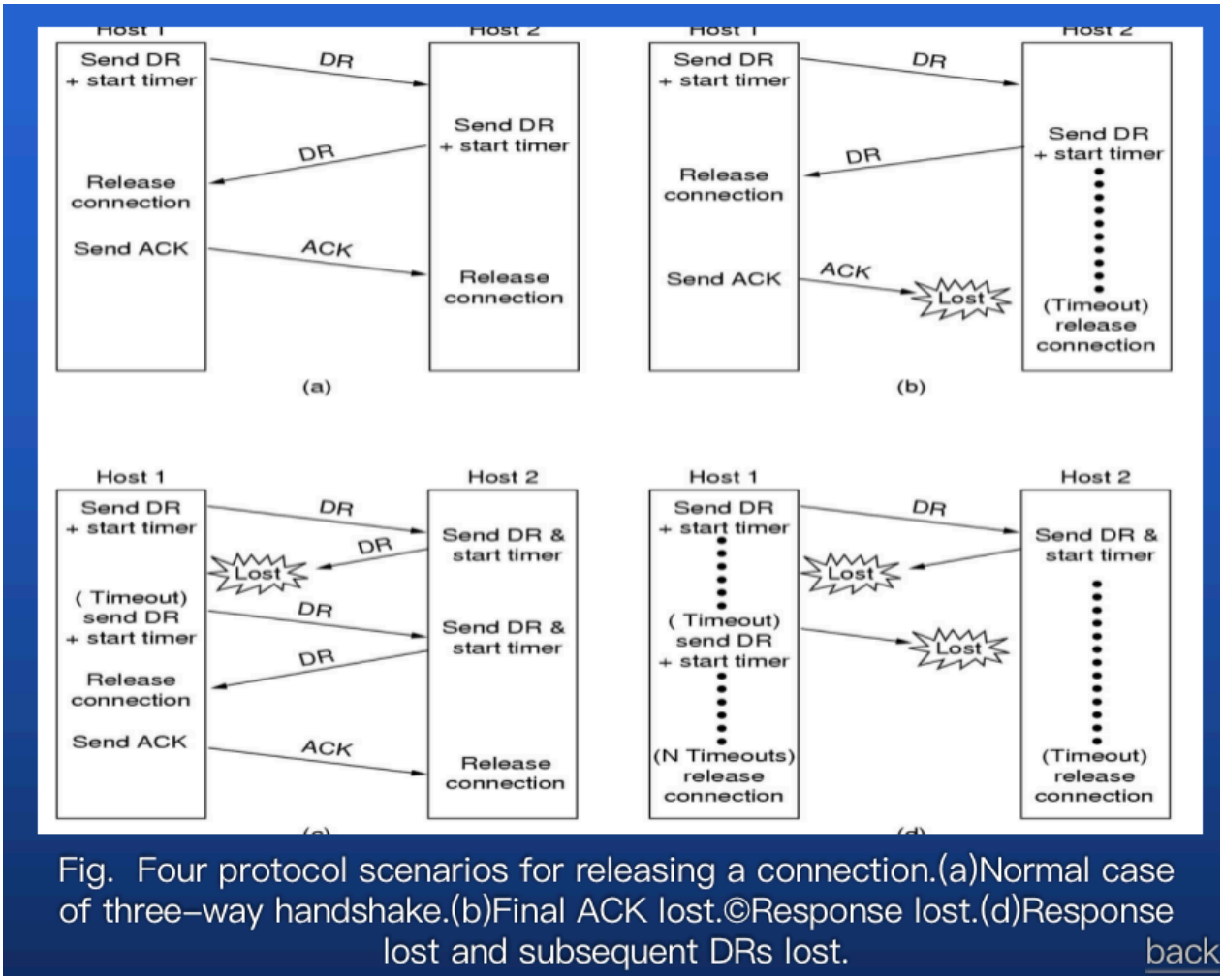
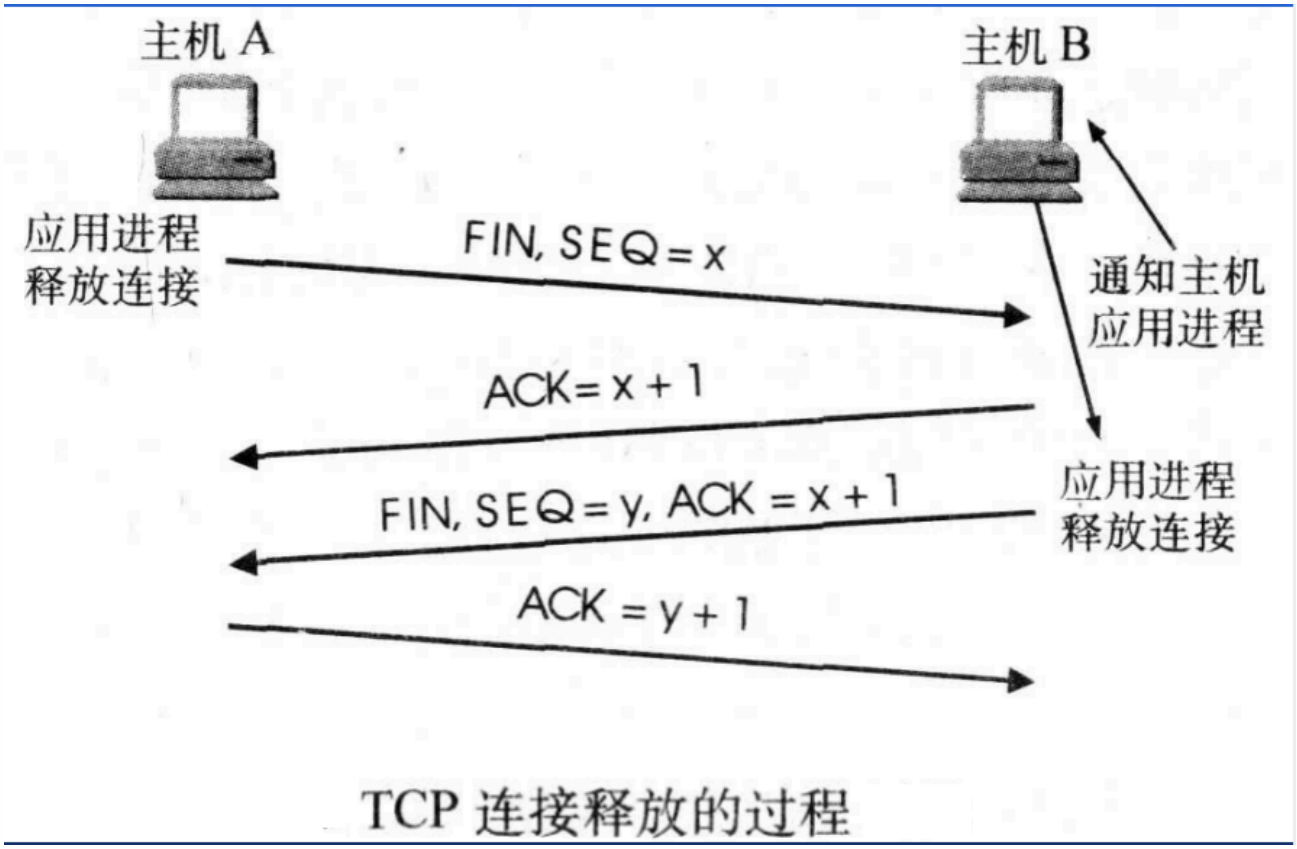


Fig. Four protocol scenarios for releasing a connection.(a)Normal case of three–way handshake.(b)Final ACK lost.©Response lost.(d)Response lost and subsequent DRs lost.

back

- After timeout,Host1 will retransmit DR, Host2 will not retransmit DR，but close the connection 主机2只要收到一个DR，就会开动定时器等待断开连接。也就是说主机2断开有两种方式，一种是ACK，一种是定时器到时。但是主机1只有一种方式，就是一定要收到DR的反馈包。
- While this protocol is not infallible, it is usually adequate.
- Problem of three-way handshake：the loss of initial DR and N retransmissions will finally result in a half-open connection 最初的DR包丢失可能会造成半连接，2断开，1还没断开
- One way to kill off half-open connections：if no TPDUs have arrived for a certain number of seconds,the connection is automatically disconnected.

2.2.1 Symmetric release



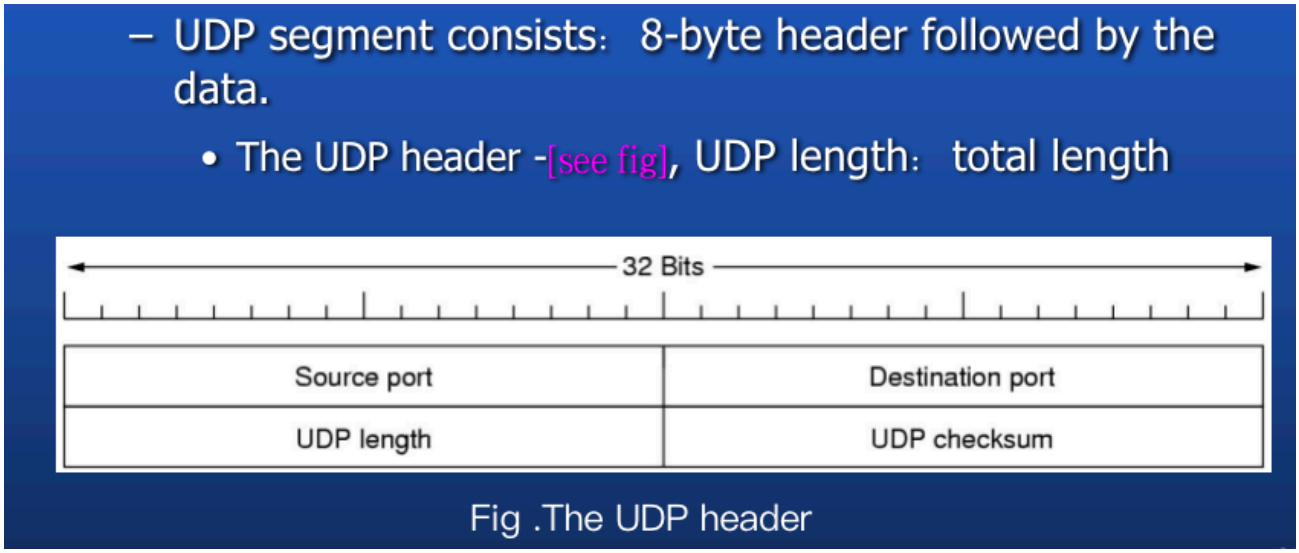
TCP 连接释放的过程

- Each direction of connection is released independently of the other one.
- TCP就是这种连接释放的，是对称的连接释放方式
- 前两个是A到B的连接断开
- 后两个是B到A的连接断开

3 THE INTERNET TRANSPORT PROTOCOLS:UDP

3.1 UDP

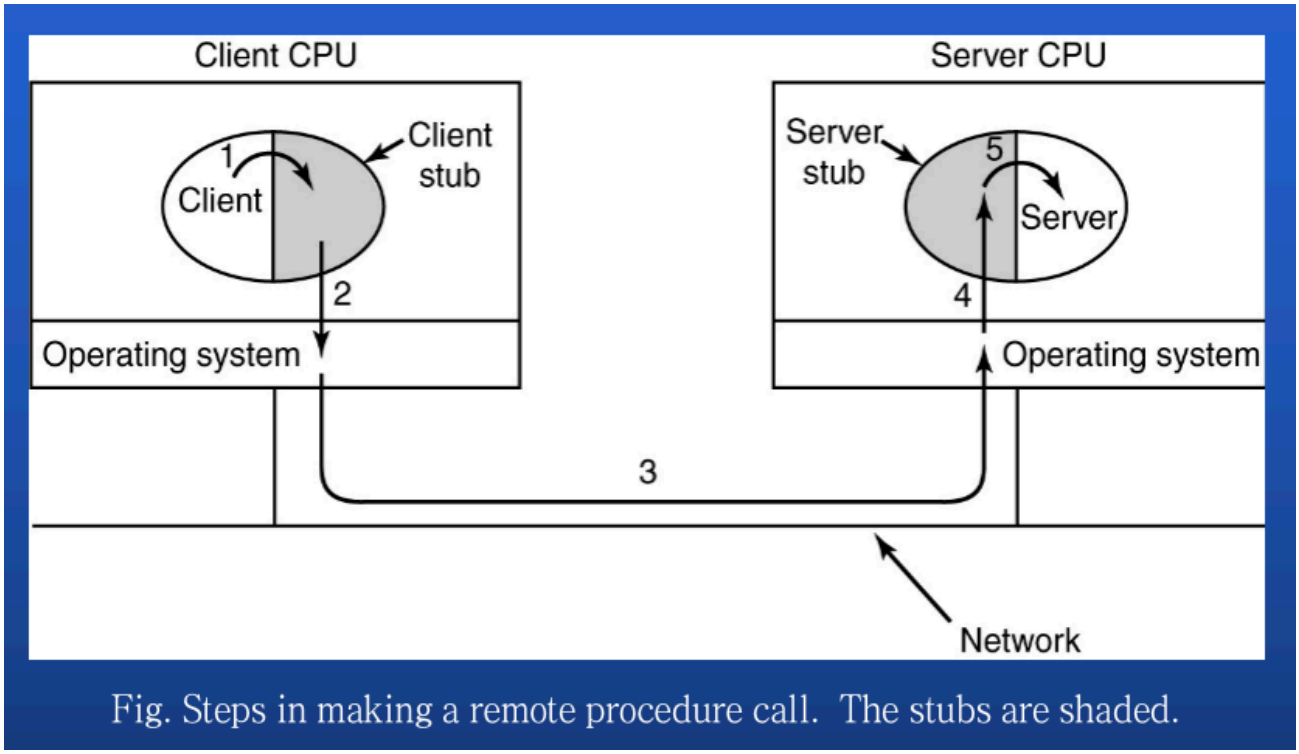
Some things UDP does not do: flow control, error control, or retransmission upon receipt of a bad segment.



- 速度快
- 可应用在Client- Server；DNS（解析域名请求，域名对应的IP地址）

3.2 Remote Procedure Call (RPC)

除了Socket，还有RPC来进行传输



- 客户端通过输出指令、参数通过网络，传到服务器端，进行操作
- 但是指针之类的比较麻烦

3.3 The Real-Time Transport Protocol （RTP）

- An application of UDP, described in RFC1889.
- The position of RTP in the protocol stack.-[see fig]
- The packet nesting. -[see fig]

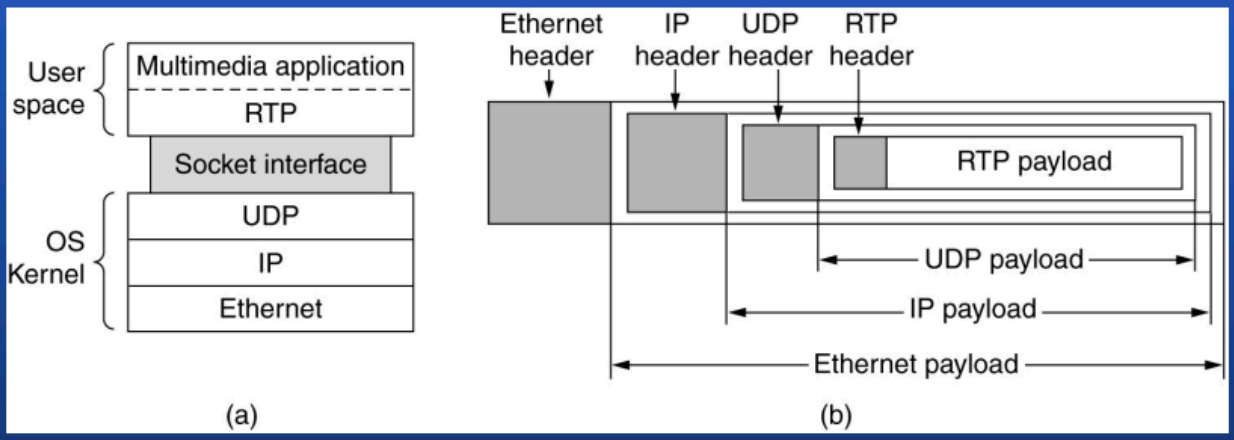


Fig. (a) The position of RTP in the protocol stack. (b) Packet nesting.

- 这个打包的形式还挺形象的

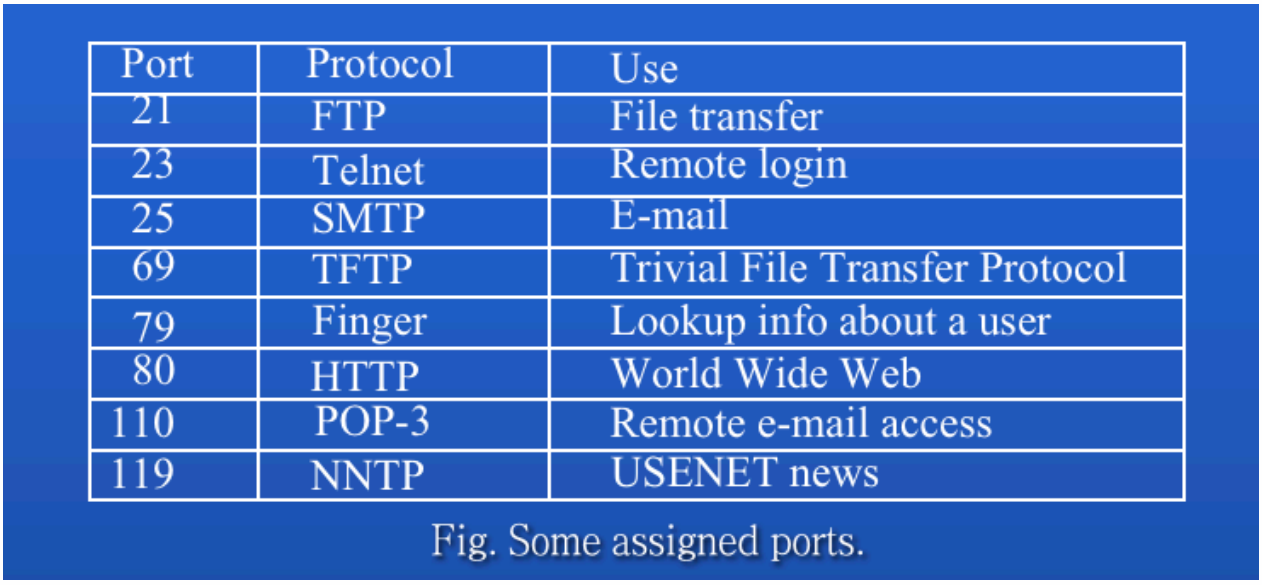
4 THE INTERNET TRANSPORT PROTOCOLS:TCP

4.1 Introduction

- A connection-oriented protocol ----TCP(Transimission Control Protocol) provides a reliable end-to-end byte stream over an unreliable internetwork. 可靠传输协议
- TCP transport entity: it manages TCP streams and interfaces to the IP layer.
- TCP furnish the reliability that most users want and that IP does not provide.
- 发送序号不是1、2、3这样的，是按照字节编号的，都是端到端的，直接送到端口号

4.2 The TCP Service Model

4.2.1 Socket

- Socket number(Address): the IP address + port(16bits)
- 

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Fig. Some assigned ports.

- 有些Socket port 是固定的，有固定的用途。<u>上面的端口号要背，考试要考！！！</u>
- 这些一般都小于1024，大于1024的一般是客户端
- TCP may send data immediately (with the PUSH flag)，e.g. Client side types carriage return, whole line must be send out immediately（这点同URG）. 另外接收方运输层收到后要把所有缓冲区中的数据立即交付应用层，而不是等接收缓冲区满了再上交（这点不同于URG）。
- Urgent data—with the URGENT flag, it must be sent out immediately, and the receiver must stop current work to process it. e.g., pressing Ctr+c, or del key. 例如：发送200字节(无URG)后发送Ctr + c(带URG)，接受方先处理加急数据ctr+c, 再处理200字节。

4.3 The TCP Segment Header - 20bytes

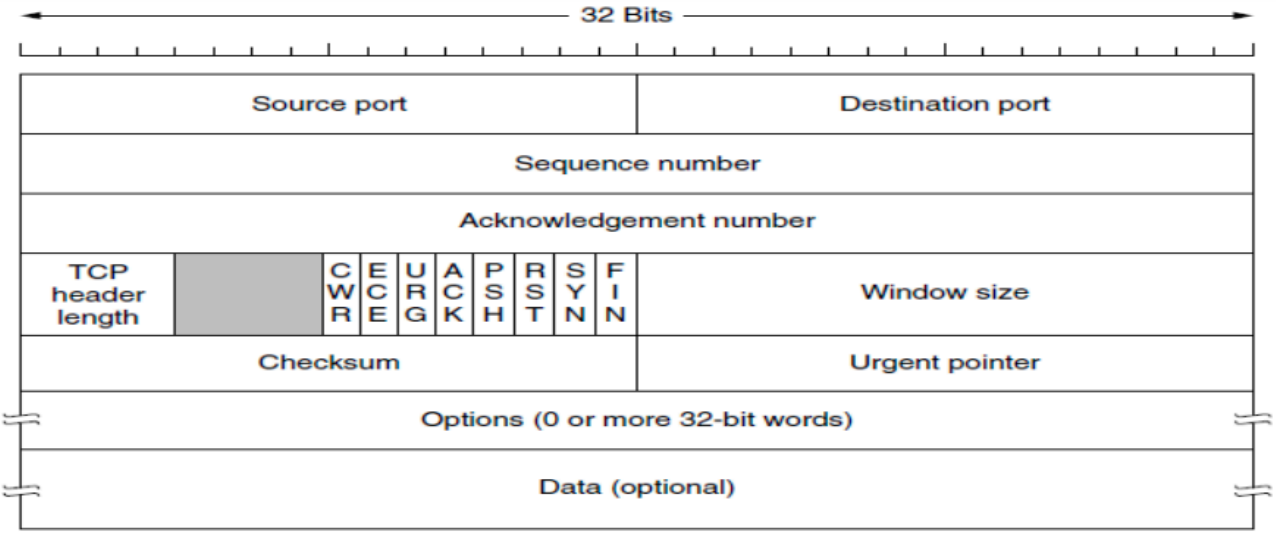


Figure The TCP header.

- include：20-byte header + + <0-N bytes data>
- **N=65535（2的16次方，由window size决定）-20(the IP header)-20(the TCP header)=65495（MAX）**
- Meaning of each field：
  - The Source port and Destination port，port number + the IP address = TSAP(48bits)
  - The Sequence number, Acknowledgement number, and the TCP header length(4-bit field, unit:32-bit word)
  - Six 1-bit flags：URG,ACK,PSH,RST,SYN（1:连接请求）,FIN（1:断开连接请求，DR）
  - Checksum （include the conceptual pseudoheader)-[see fig]
    - 发送方：先把Checksum字段清0,对三部分(伪头,TCP首部,TCP数据)求16位反码和(1's complement sum),结果取反,作为checksum值。例如，4f59 +' b224 +' 902b +' 0000= 1017d +' 902b = 017e +' 902b +' 0000 = 9229 +' 0000 = 9229，checksum=9229的反码=6dd6
    - 接收方：对三部分(伪头,TCP首部,TCP数据)求16位反码和,结果应是ffff. 例如，4f59 +' b224 +' 902b +' 6dd6 = ffff
    - Options
  - Sliding window
    - The selective repeat 选择重传协议
- Urgent pointer 加急指针：加急数据和普通数据的分界线，用来指示数据中哪些是加急数据
- Two limits restrict the segment size：
  - Segment max size < min(65515 bytes, MTU-20) 这个20是IP数据包头的字节
  - MTU---Maximum transfer unit (the maximum length of IP datagram), in practice, the MTU is usually 1500 bytes (The Ethernet payload size).
  - 以太网帧

4.4 TCP Connection Establishment

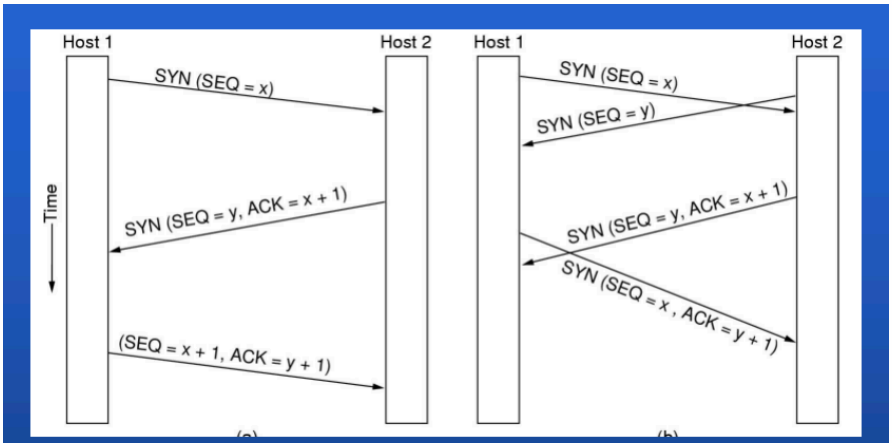
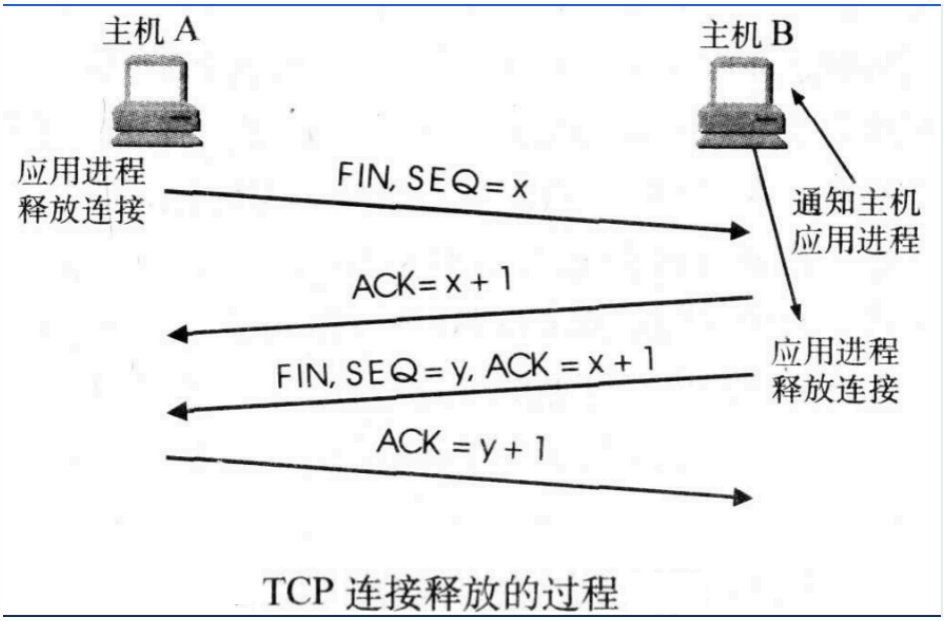


Fig.(a) TCP connection establishment in the normal case. (b)Call collision

- 三次握手：SEQ是包的序号，ACK是收到的对方帧号+1代表我下一个希望收到的SEQ是ACK=SEQ+1，之前的SEQ所代表的那些数据我都已经接收到了。
- 虽然建立连接的时候，没有data，但是这么理解会比较好，
- 图b是呼叫冲突，两个主机不约而同发送连接请求
- 收到呼叫冲突的时候建立连接，也能成功

4.5 TCP Connection Release

- Connections released：both sides need to send total 4 or 3 segment.



TCP 连接释放的过程

- Each direction of connection is released independently of the other one.
- TCP就是这种连接释放的，是对称的连接释放方式
- 前两个是A到B的连接断开
- 后两个是B到A的连接断开

4.6 TCP Connection Management Modeling

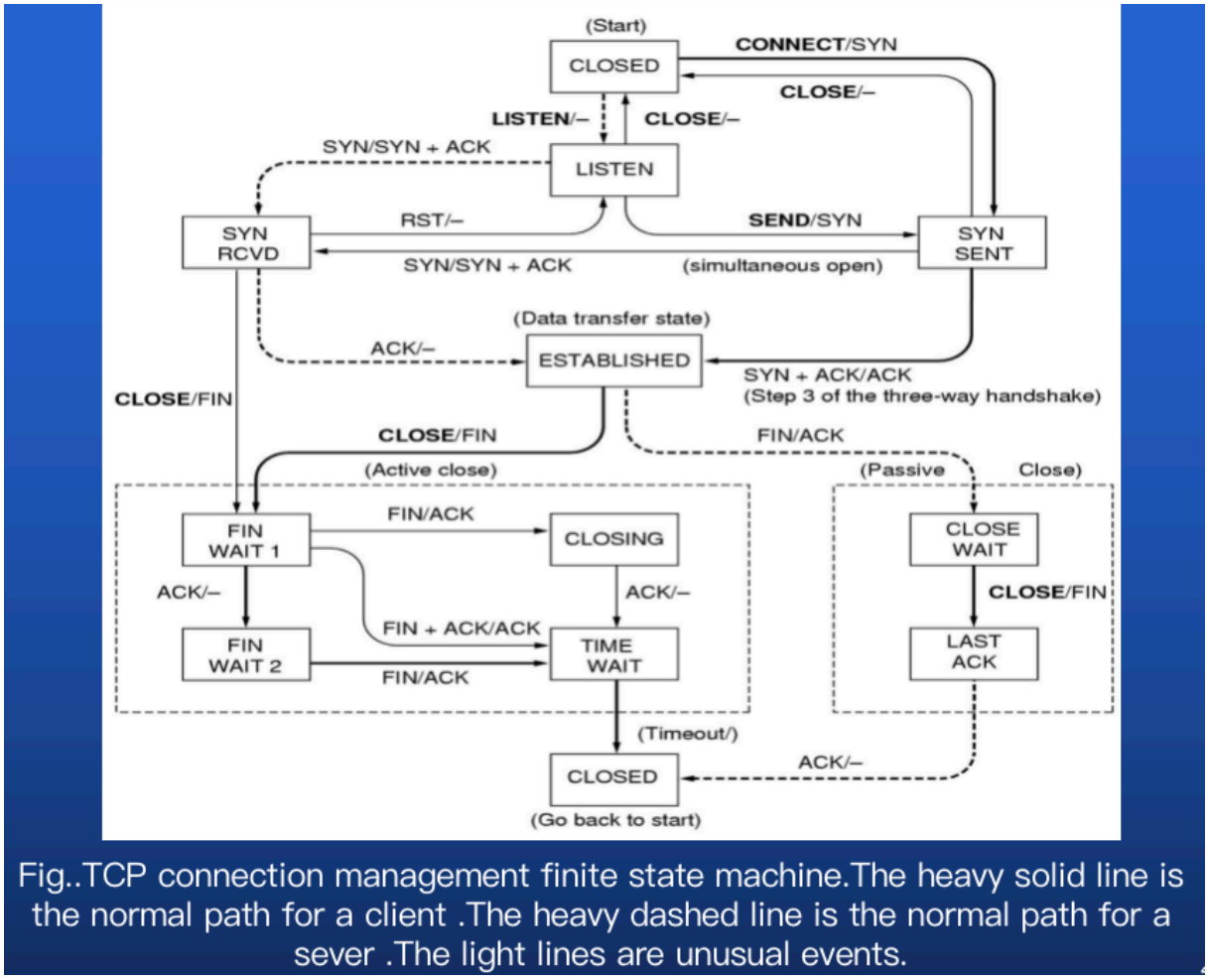


Fig..TCP connection management finite state machine.The heavy solid line is the normal path for a client .The heavy dashed line is the normal path for a sever .The light lines are unusual events.

4.7 TCP Sliding window

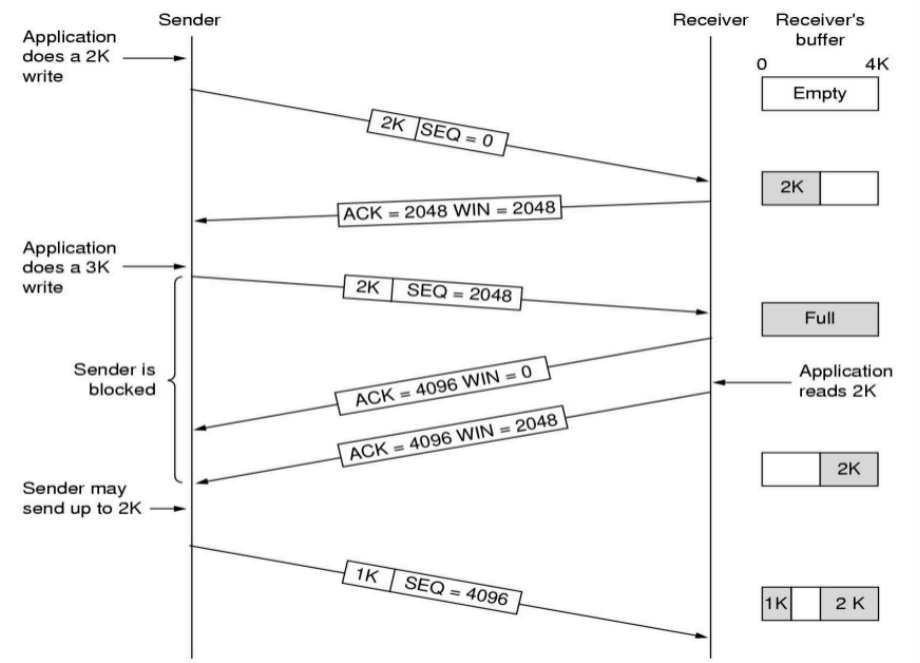


Fig. Window management in TCP

- 可以读一下上面的示意图
- WIN代表你可以发的最多的字节（我目前空的窗口），WIN=0就是让对方暂停发送

4.7.1 发送方浪费问题

- The worst case: Telnet—162 bytes of bandwidth are used and four segments are sent for each character typed
  - 41 byte IP datagram (1 char) ---->
  - <---- 40 byte IP datagram (TCP ACK segment)
  - <---- 40 byte IP datagram (TCP WIN update segment)
  - <---- 41 byte IP datagram (echo 1 char)
- Solution: TCP ACK segment and WIN update segment are delayed 500 ms, so that the last 3 segment can be merged into one. But the efficiency is still less than 1/41.

> Solution(Nagle's algorithm):
>
> 用来解决发送方每次只发送一个字节问题。
>
> BUFFER 就是来解决，网络慢、输入快，每次只发一点点东西很浪费。所以加入BUFFER来帮助，达到半BUFFER或者额定最高segment再一起发送。
>
> - when data come into the sender one byte at a time, just send the first byte and buffer all rest until the outstanding byte is acknowledged. Then send all the buffered characters in one TCP segment and start buffering again until they are all acknowledged.
> - If the user is typing quickly and the network is slow, a substantial number of characters

may go in one segment, greatly reducing the bandwidth used.

- The algorithm additionally allows a new packet to be sent if enough data have trickled in to fill half the sending window or a maximum segment.

4.7.2 接收窗口愚钝问题 Silly window syndrome

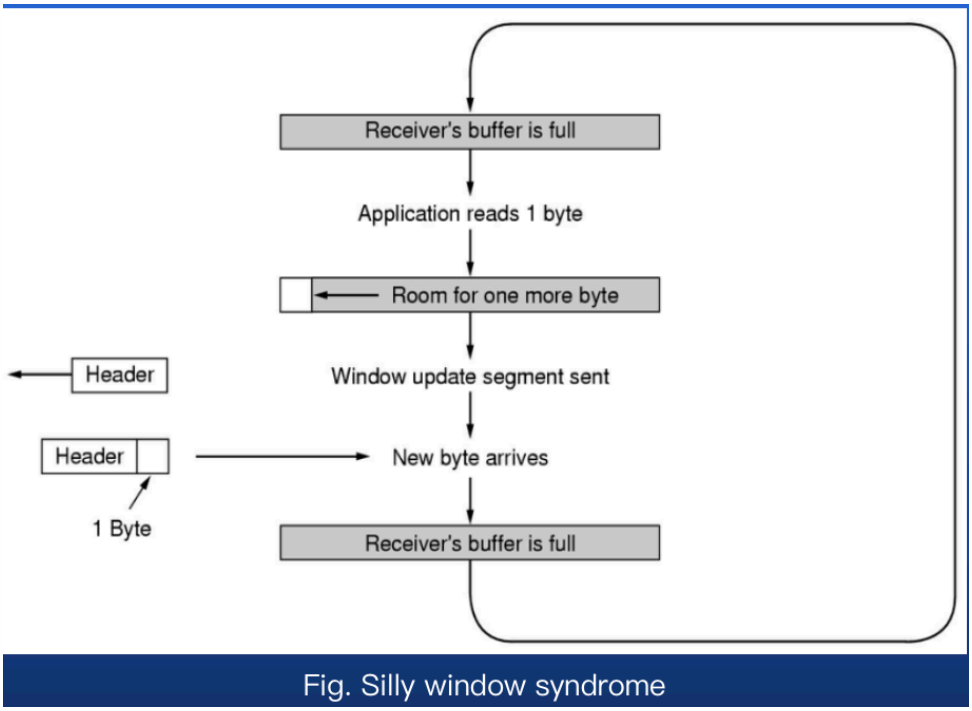


Fig. Silly window syndrome

- 接收窗口的应用程序读取很慢，BUFFER空出来很慢，隔一段时间发送WIN，很浪费信道
  - Clark's solution
    - preventing the receiver from sending a window update for 1 byte.解决接收方每次只读一个字节
    - The receiver should not send a window update until it can handle the maximum segment size it advertised when the connection was established, or its buffer is half empty, which ever is smaller.
    - Futhermore, the sender can also help by not sending tiny segments. It should wait until it has accumulated enough space in the window to send a full segment or at least one containing half of the receiver's buffer size.
  - **Nagle solves problem: sending application delivers TCP data a byte at a time.**
  - **Clark solves problem: receiving application sucks TCP data up a byte at a time.**
  - 都是出于信道浪费的问题，现在不怎用了。

- What to do with out-of-order segments: 丢弃不按序到达的报文段(使用后退n协议), 或者保留不按序到达的报文段(使用选择性重传协议). **TCP的报文段经常是错序的，因此选择重传协议比较适用**

4.8 TCP Timer Management

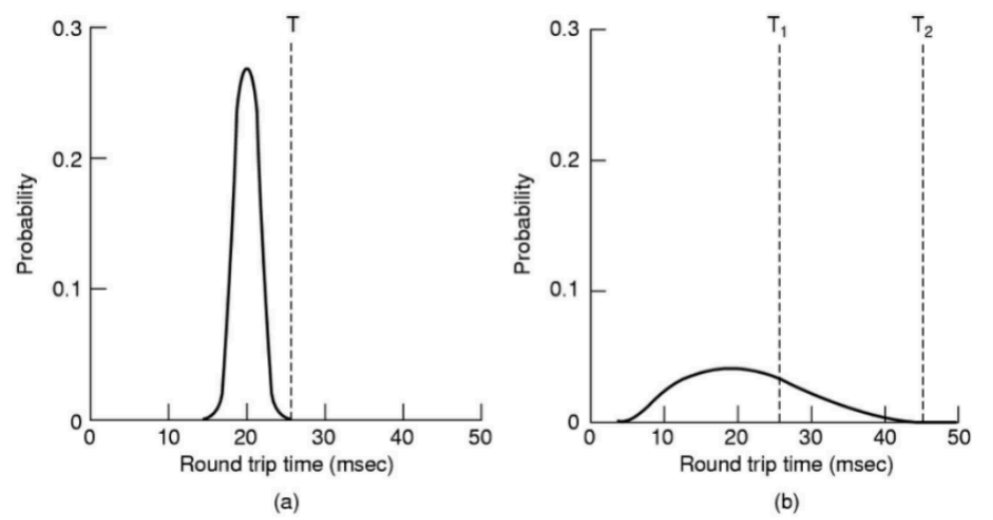
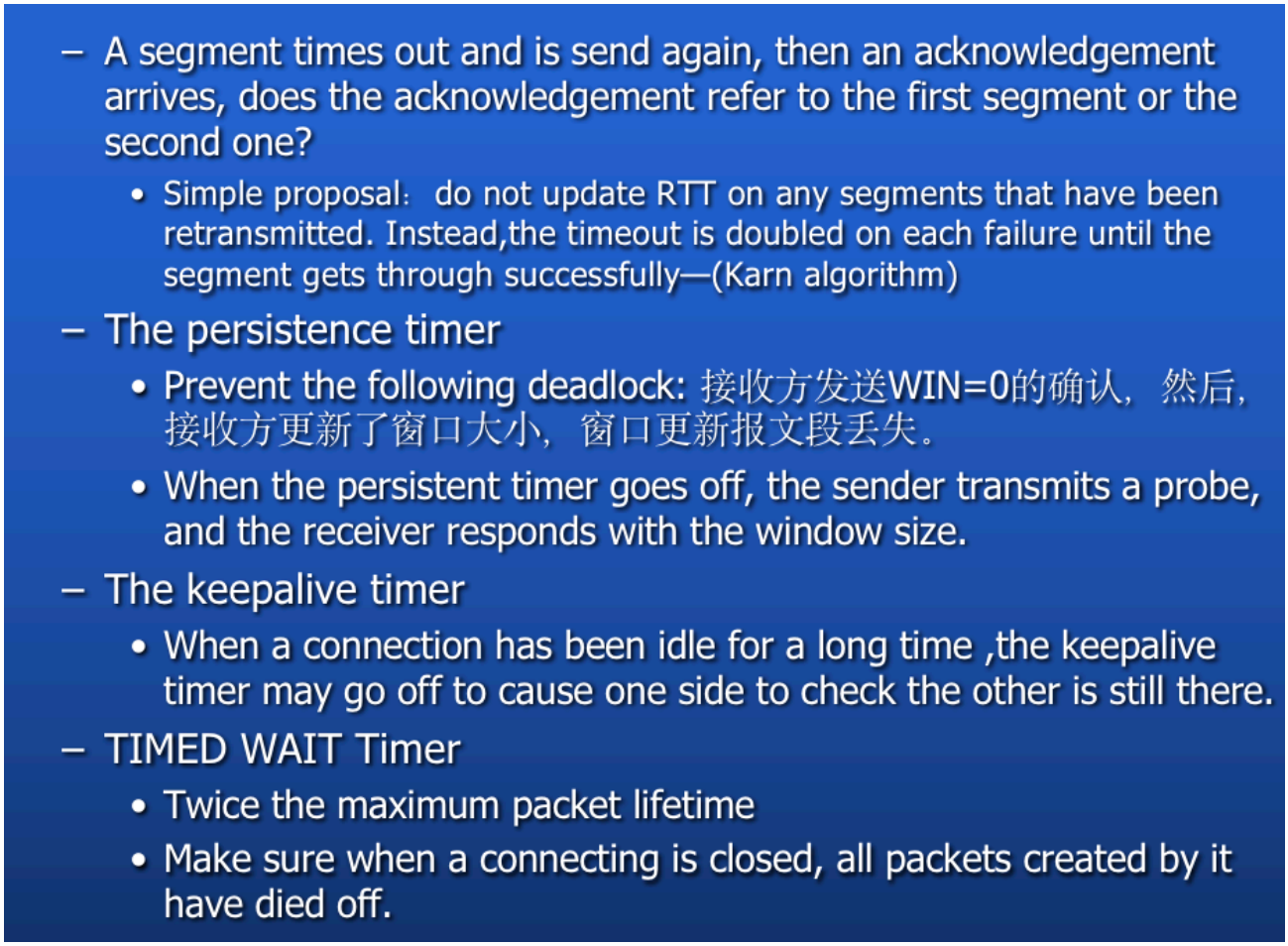


Fig . (a)Probability density of acknowledgement arrival times in the data link layer.(b)Probability density of acknowledgement arrival times for TCP

- 链路层的trip time是有极大的某一概率密度
- 但是TCP trip范围很大因此需要考虑一个合适的

- Solution：
  - Using a highly dynamic algorithm that constantly adjusts the timeout interval, based on continuous measurements of network performance.（**Jacobson**）
  - **Principle： SRTT = α * SRTT+(1-α) * R 要背！！！**
    - SRTT: Smoothed Round-trip Time
    - R: measured round-trip time，最近测量出来的trip time，对SRTT进行更新
    - α: typical value is 7/8
  - Early in TCP/IP, Timeout = β * SRTT, typical β is 2.
  - At present, Timeout = SRTT + 4 * D 现在用这个公式比较多
    - $D = \alpha * D + (1 - \alpha) * |SRTT - R|$



- 还有一些timer

4.9 TCP 拥塞控制

4.9.1 why

机器上接收快，发送慢，数据包超过内存，缓存区不够用，出现丢包

- Two potential problems：Network’s capacity（网络太拥堵了）, Receiver’s capacity（接收方BUFFER太小，丢包）

4.9.2 Solution

- Each sender maintains two windows:
  - The window receiver has granted 接收方会批准发送方的发送窗口大小 TCP-WIN
  - The congestion window: Network's capacity estimated by the sender. 发送方会估算拥塞窗口 网络拥堵的情况
  - 发送方窗口大小是取上述两者的最小值。
- 如何取估算拥塞情况?
  - Congestion window size grows exponentially until it reaches to the window size granted by receiver. 指数增长发送包
  - When a timeout occurs or the receiver's window is reached，congestion window size stop growing. 如果出现超时 / 接收方窗口到最大了，发送方窗口停止

4.9.3 具体 - 估算拥塞情况 - 慢启动算法:

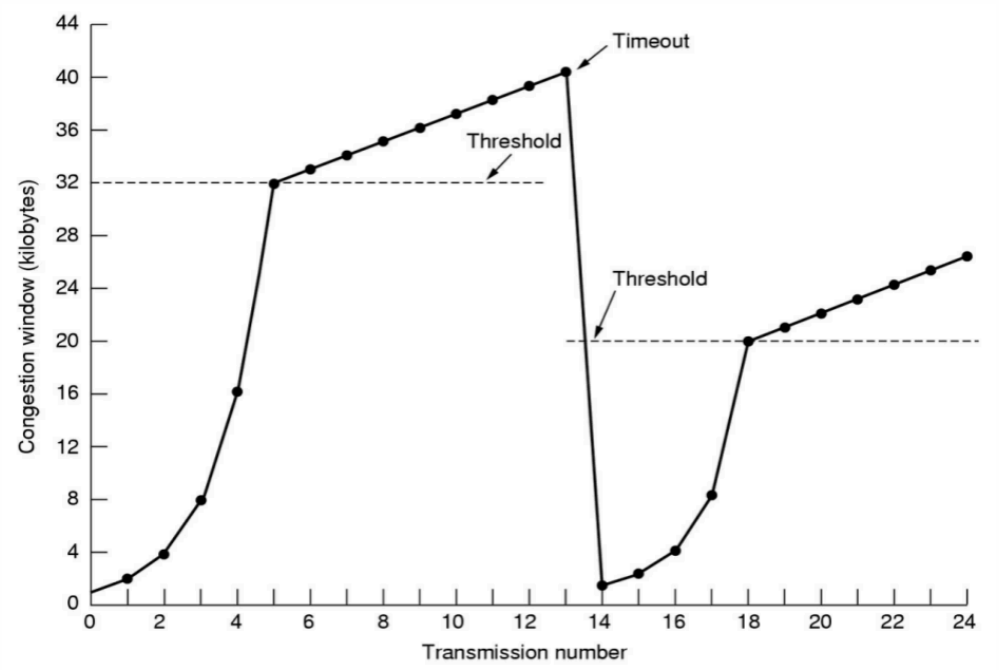


Fig .An example of the Internet congestion algorithm

- 具体中还会增加一个 Threshold 门限值，在此之前大步增长（指数增长），在此之后谨慎搜索（线性增长）
- 出现丢包后，window归最小，从头开始，Threshold变成最大的窗口大小的一半（40 / 2 = 20）
- 你也能从当前的 threshold 读出之前达到的最大窗口大小

5 题目

data rate 和 TPDU