Transport-layer 传输层

3.2 multiplexing and demultiplexing

Multiplexing at sender： 多路复用在发送方

- gathering data chunks at the source host from different sockets.
- gathering data chunks at the source host from different sockets, encapsulating each data chunk(collection of bits) with header information (that will later be used in demultiplexing) to create segments
- passing the segments to the network layer.
- 在源主机上从不同的socket收集数据块。
- 在源主机上从不同的套接字收集数据块，用头信息(稍后将用于解复用)封装每个数据块(位的集合)来创建段
- 将网段传递给网络层。

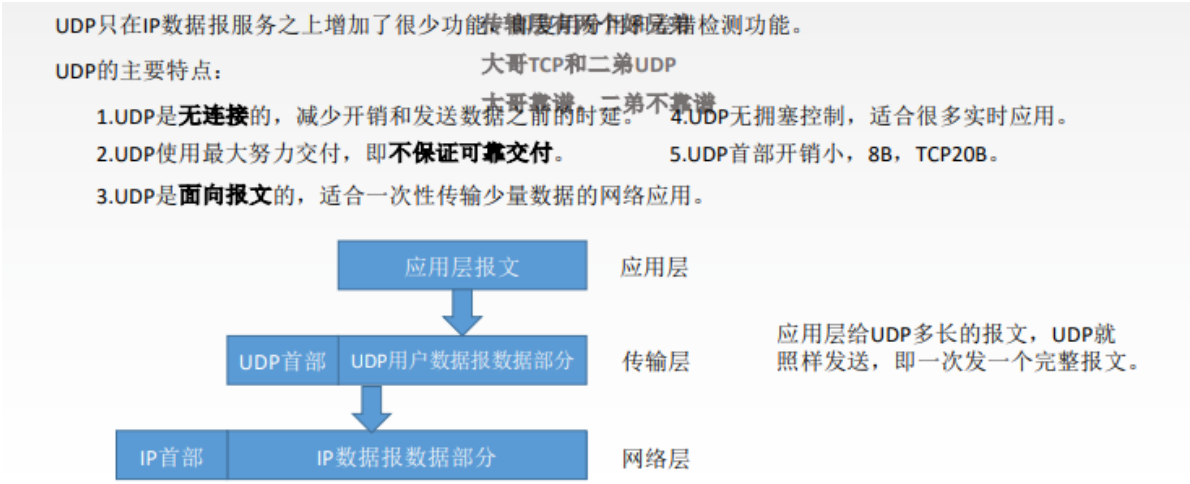
Demultiplexing at receiver：接收方多路分解

use header info to deliver received segments to correct socket

How demultiplexing works:

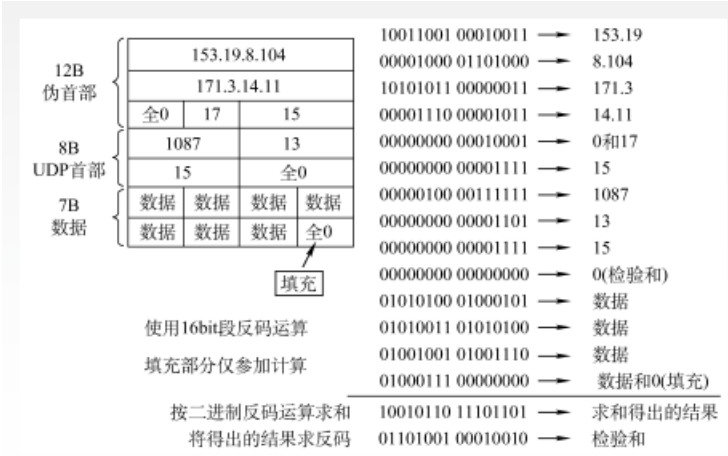
- host receives IP datagrams 数据报 (message, data)
    - i. each datagram has source IP address,
      destination IP address
    - ii. each datagram carries one transport□layer segment (packets)
    - iii. each segment has source, destination
      port number
- host uses IP addresses & port numbers to direct segment to appropriate socket

3.3 connectionless transport: UDP



校验：注意反码运算 1's compomect 最高位溢出则结果加一

校验和为结果值的反码



在发送端：

1.填上伪首部

2.全0填充检验和字段

3.全0填充数据部分（UDP数据报要看成许多4B的字串接起来）

4.伪首部+首部+数据部分采用二进制反码求和

5.把和求反码填入检验和字段

6.去掉伪首部，发送

在接收端：

1.填上伪首部

2.伪首部+首部+数据部分采用二进制反码求和

3.结果全为1则无差错，否则丢弃数据报/交给应用层附上出差错的警告。

3.4 connection-oriented transport: TCP

面向连接传输（可靠传输 reliable data transport）

1.TCP是面向连接（虚连接）的传输层协议。**打call**

2.每一条TCP连接只能有两个端点，每一条TCP连接只能是点对点的。

3.TCP提供可靠交付的服务，无差错、不丢失、不重复、按序到达。**可靠有序，不丢不重**

4.TCP提供全双工通信。⟹ 发送缓存　准备发送的数据&已发送但尚未收到确认的数据

接收缓存　按序到达但尚未被接受应用程序读取的数据&不按序到达的数据

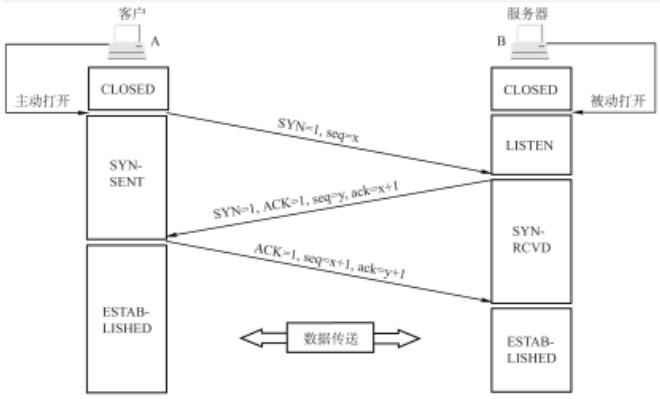
5.TCP面向字节流 ⟹ TCP把应用程序交下来的数据看成仅仅是一连串的**无结构的字节流**。

流：流入到进程或从进程流出的字节序列。

TCP 连接建立（三次握手）

seq:请求确认信息，ack:确认信息并依赖于ack

1. 客户机发起：**SYN=1,seq=x**
2. 服务器响应并返回确认信息：**SYN=1,ACK=1,ack=x+1,seq=y**
3. 客户端收到并返回确认信息：**ACK=1,ack=y+1,seq=x+1**



ROUND 1:

客户端发送**连接请求报文段**，无应用层数据。

SYN=1，seq=x(随机)

ROUND 2:

服务器端为该TCP连接**分配缓存和变量**，并向客户端返回**确认报文段**，允许连接，无应用层数据。

SYN=1，ACK=1，seq=y(随机)，ack=x+1

ROUND 3:

客户端为该TCP连接**分配缓存和变量**，并向服务器端返回确认的确认，可以携带数据。

SYN=0，ACK=1，seq=x+1，ack=y+1

TCP连接释放（四次握手）

1. 客户机主动发起连接释放报文段，停止发送数据，主动关闭TCP连接The client initiates a connection to release the packet segment, stops sending data, and closes the TCP connection：

   FIN = 1,seq = u

2. 服务器收到连接释放报文段后确认The server acknowledges the connection release packet after receiving the segment：

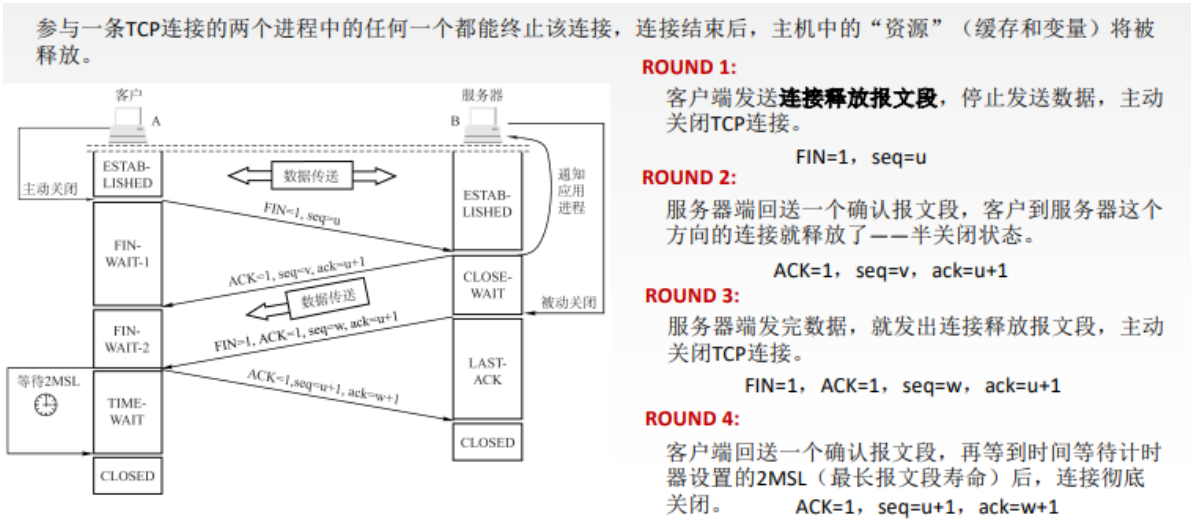
   ACK = 1,seq = v,ack=u+1

3. 服务器通知客户端释放连接The server notifies the client to release the connection：

   FIN = 1, ACK=1,seq=w,ack=u+1

4. 客户端收到连接释放报文后，发出确认After receiving the connection release packet, the client sends an acknowledgement：

   ACK=1,seq=u+1,ack=w+1



流量控制

TCP利用滑动窗口机制实现流量控制。

★在通信过程中，接收方根据自己接收缓存的大小，动态地调整发送方的发送窗口大小，即接收窗口rwnd （接收方设置确认报文段的窗口字段来将rwnd通知给发送方），发送方的**发送窗口**取**接收窗口rwnd**和**拥塞窗口cwnd**的最小值min

接收窗口rwnd：

拥塞窗口cwnd：

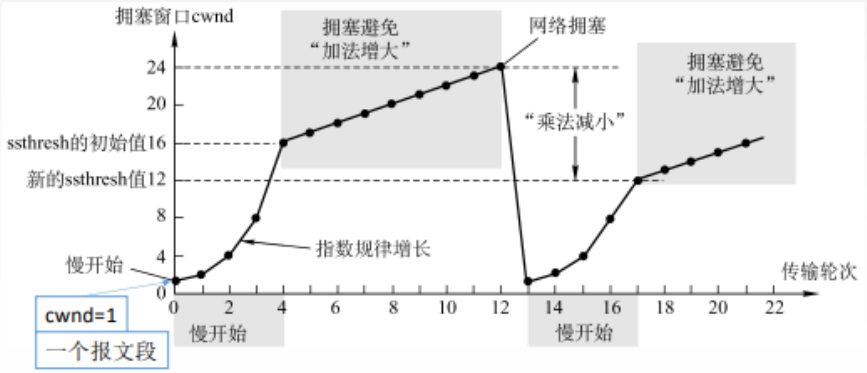
作用：（为什么要通知窗口大小）

- 防止接收方内存溢出，分组丢失 **Prevent receiver memory overflow and packet loss**

3.5 TCP congestion control TCP拥塞控制

- 慢开始 Slow start
- 拥塞避免 Congestion avoidance

慢开始和拥塞避免



一个最大报文段长度MSS

一个**传输轮次**：

发送了一批报文段并收到它们的确认的时间。

一个往返时延RTT。

开始发送一批拥塞窗口内的报文段到开始发送下一批拥塞窗口内的报文段的时间。

- 快重传 fast retransmission
- 快恢复 quick recovery

快重传和快恢复