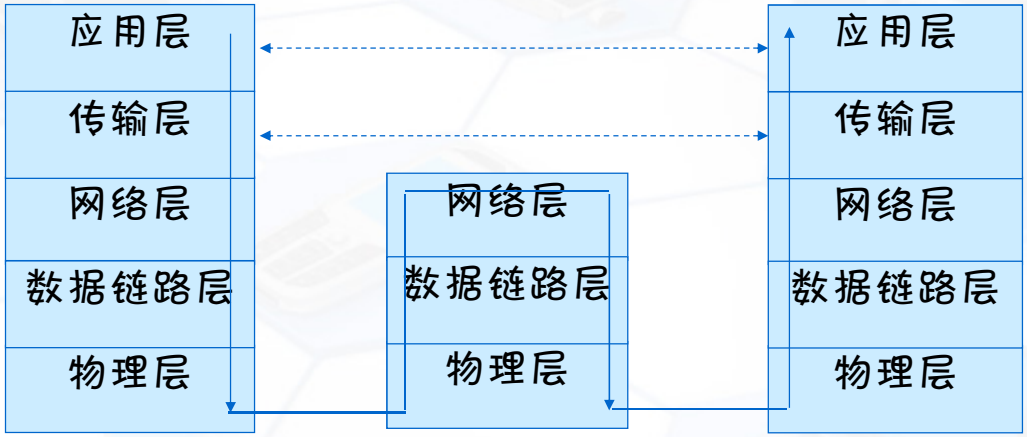
1. 为什么建立在不可靠服务的网络层之上的传输层可以提供可靠传输？
2. 传输层提供的都是可靠服务吗？为什么？

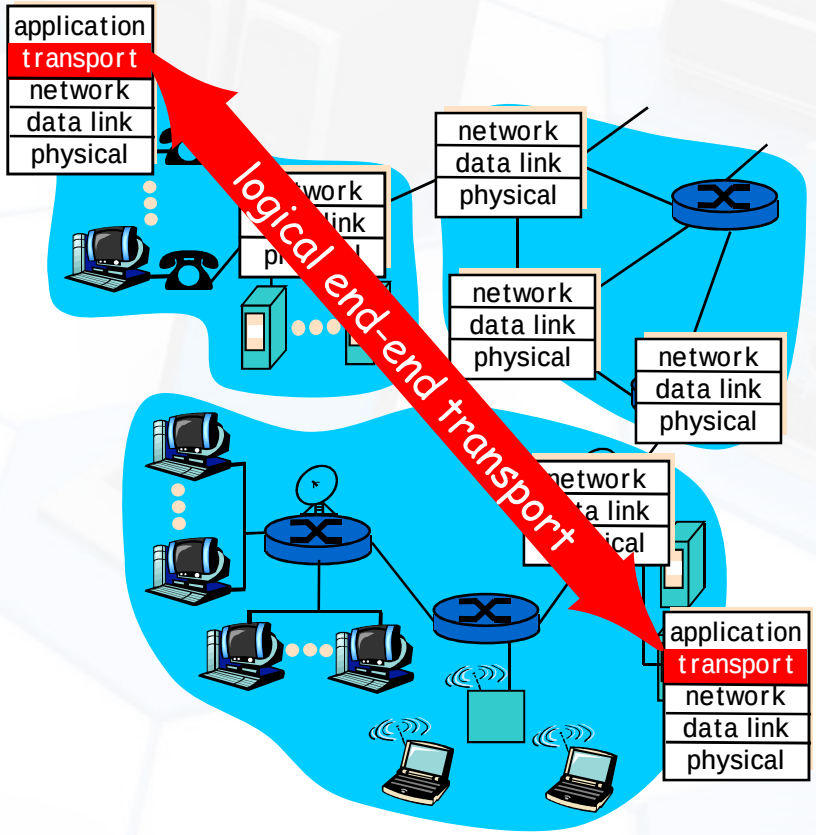
一、 传输层概述

1. 为什么需要传输层



- 介于通信子网和资源子网之间，对高层用户屏蔽了通信的细节
- 弥补了通信子网所提供服务的差异和不足，提供端到端之间的无差错保证
- 传输层工作的简繁取决于通信子网提供服务的程度

- 传输层为运行在不同主机上的应用进程间提供逻辑通讯
- 传输协议运行于终端系统
  - ü 发送端：将应用层的消息转换成传输层要求的数据段，并交给网络层
  - ü 接收端：将收到的数据段再组合成消息，并交给应用层
- 对应用层而言，可用的传输协议不只一种，互联网中主要是TCP 和 UDP



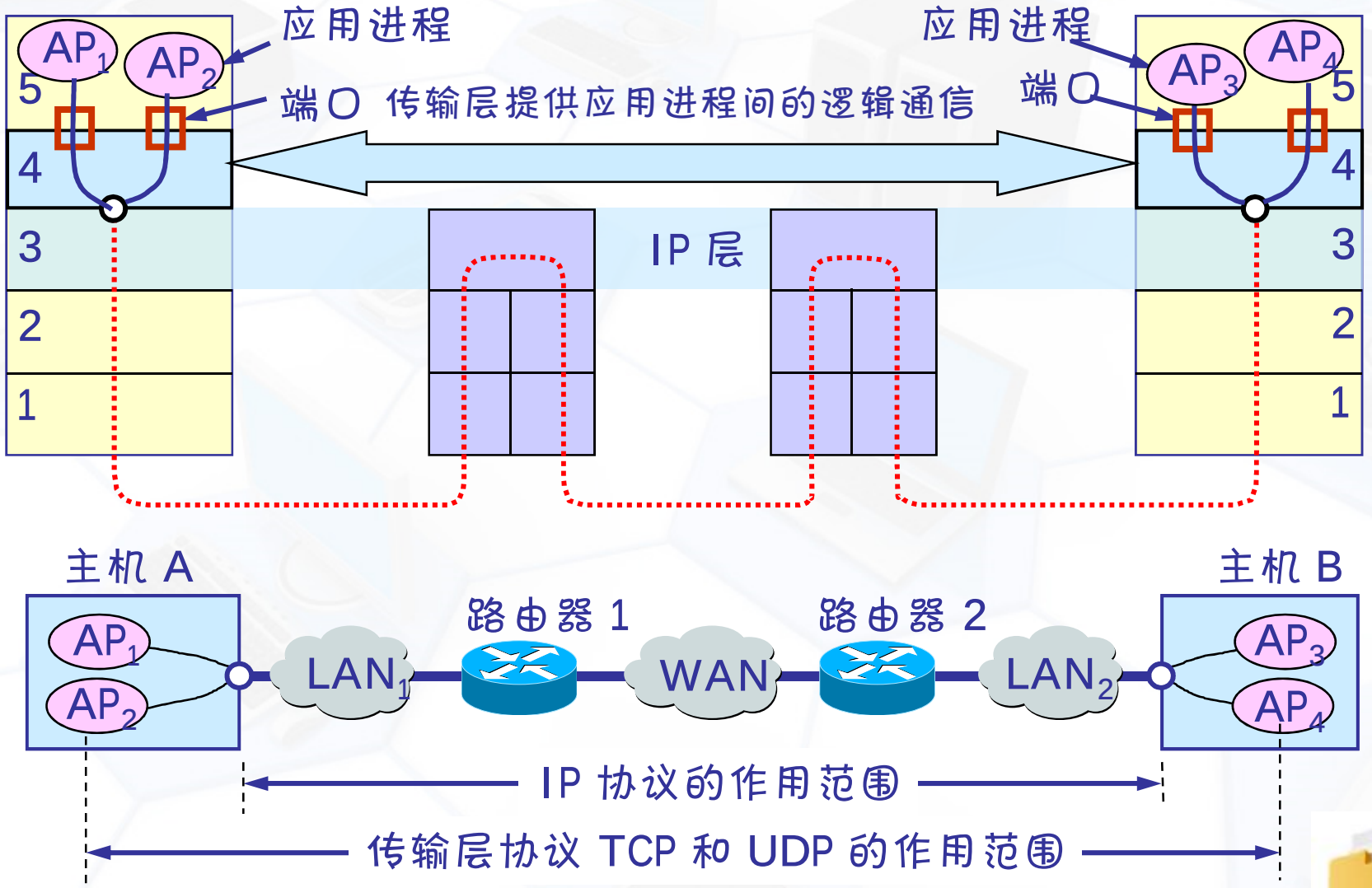
2. 传输层的服务

可以概括为：加强和弥补通信子网服务

- 传输层的“加强”是对服务质量而言，指提高服务的可靠性

- “弥补”是对服务类型而言，传输层提供端到端进程通信，而子网提供点到点通信

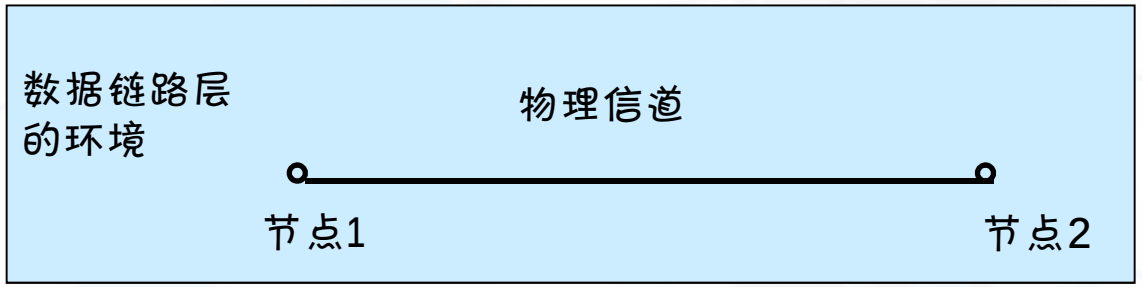
传输层为相互通信的应用进程提供了逻辑通信



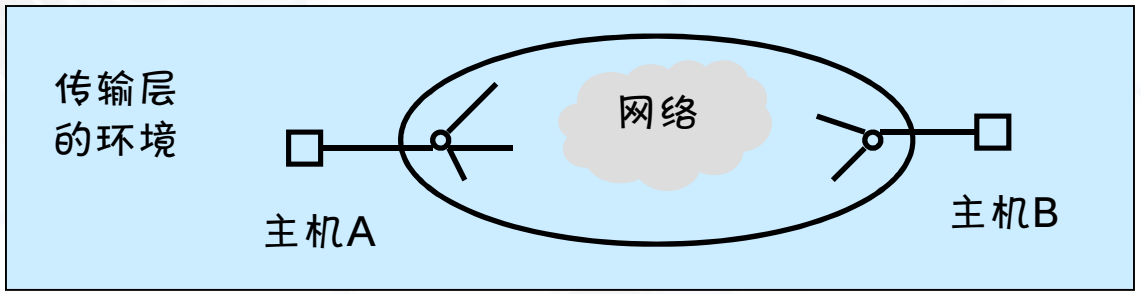
二、传输层要解决的问题

1. 编址

§ 数据链路层连接的双方是点到点连线的“两点”



§ 传输层的端到端传输连接可能跨越许多网络



2. 连接标识的获取

- 采用客户-服务器模型
- 初始连接协议 ——进程服务器作为应用服务器的代理，侦听传输连接请求；收到请求时，则装入应用服务器，并将连接转交
- 一台主机不可能提供所有的服务，改进的模型之一是增加公认的TSAP个数，使用名字服务器或目录服务器来提供应用服务器的TSAP地址

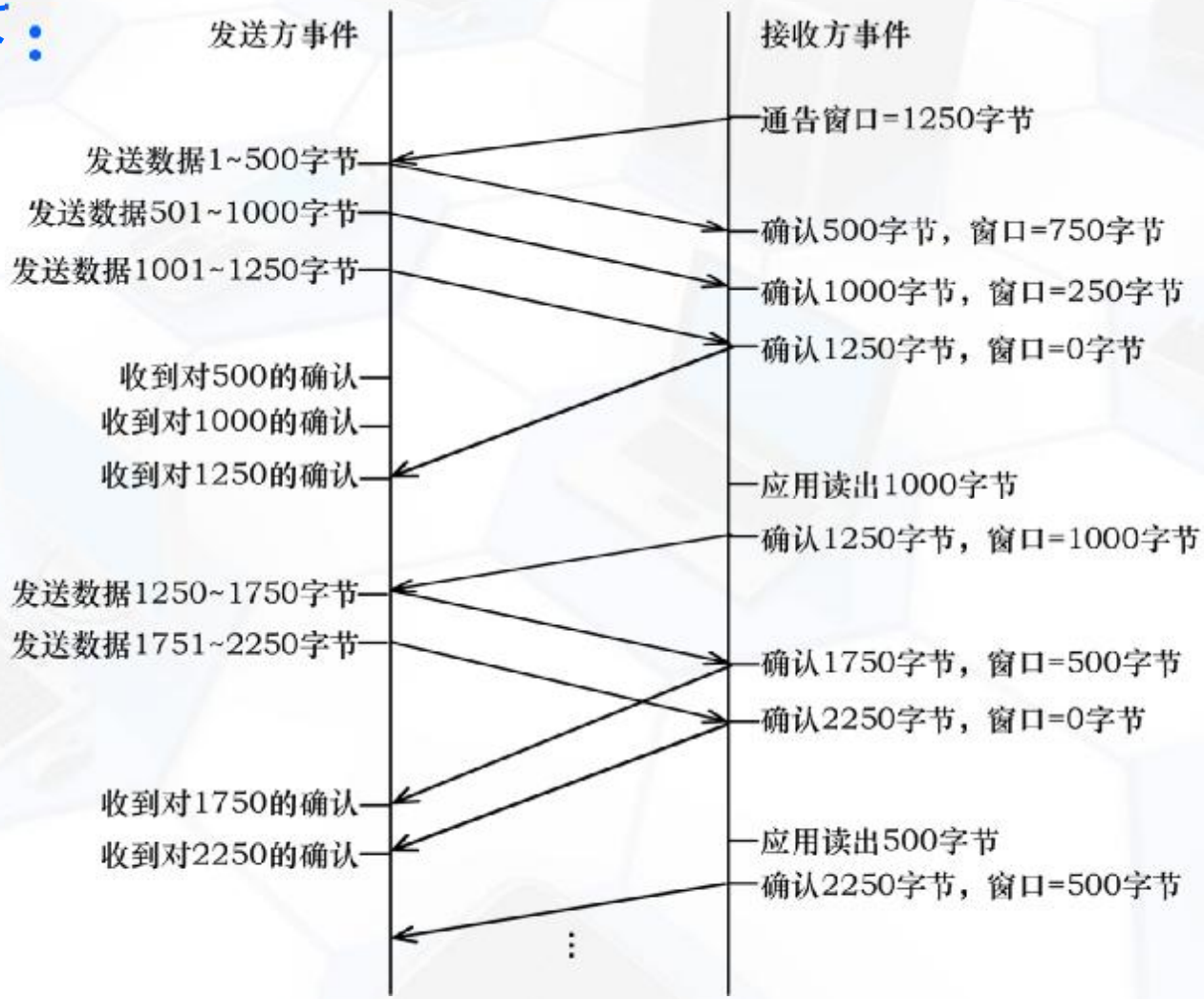
3. 流量控制

- 在每个连接上使用滑动窗口技术
- 在传输层，发送方的发送速率不仅受接收方处理能力限制，而且还要受通信子网处理能力的限制
- 采用动态分配的方法。建立连接的双方都将分配一个缓冲区作为接收数据的存放空间，并相互通知对方。此后，每次对接收数据确认的同时发布一个窗口通告，报告剩余窗口的大小，以此控制发送的速度

§ 零窗口通告

发送方收到一个零窗口通告时，必须停止发送，直到接收方重新通告一个正的窗口

§ 举例如下：



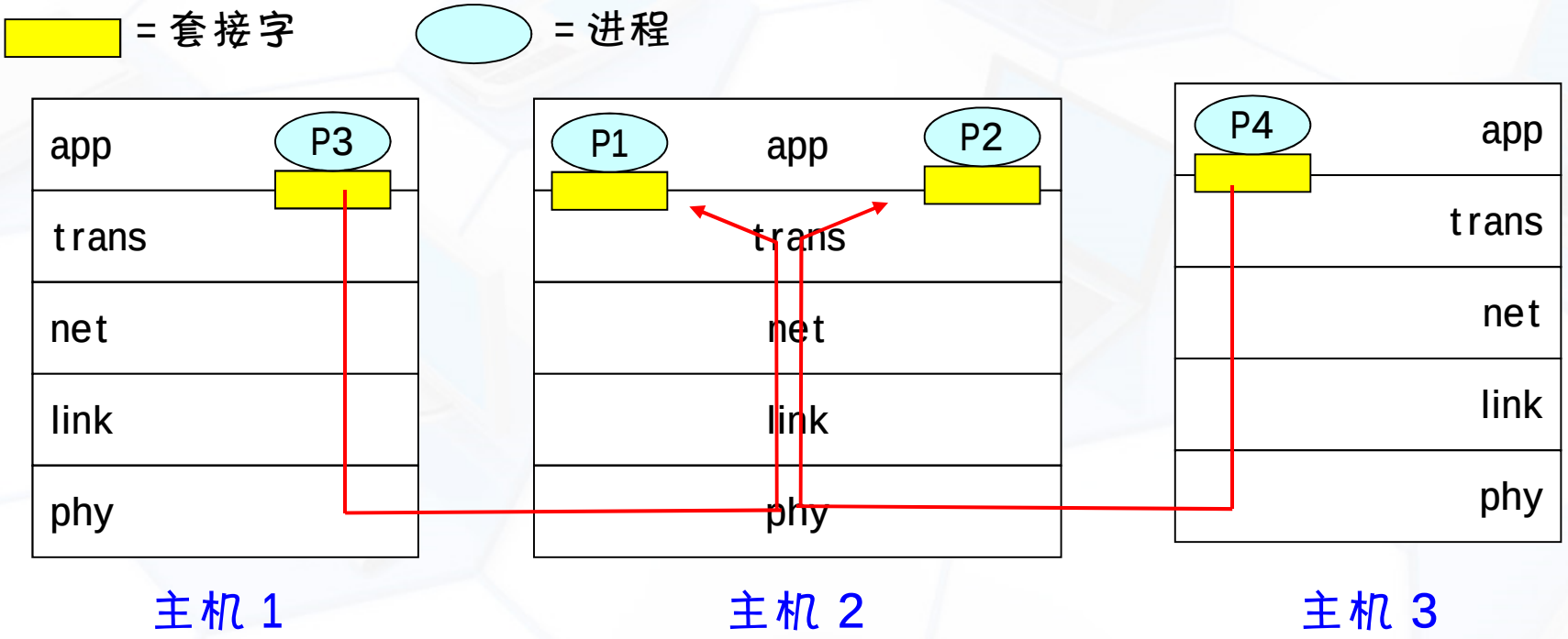
4. 多路复用

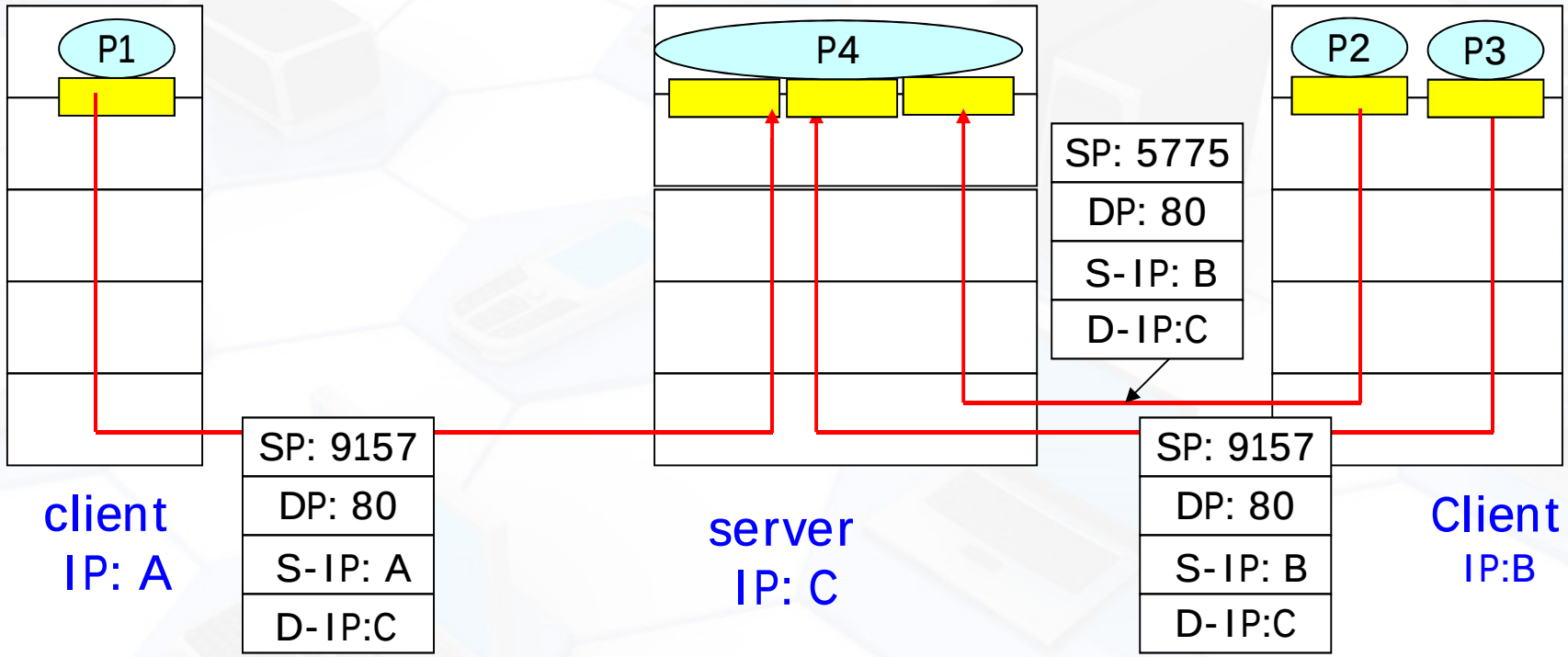
接收方进行多路复用分解

将收到的数据段交给正确的套接字

发送方进行多路复用

从不同套接字中收集数据，并给数据封装上头部信息(用于今后的多路复用分解)

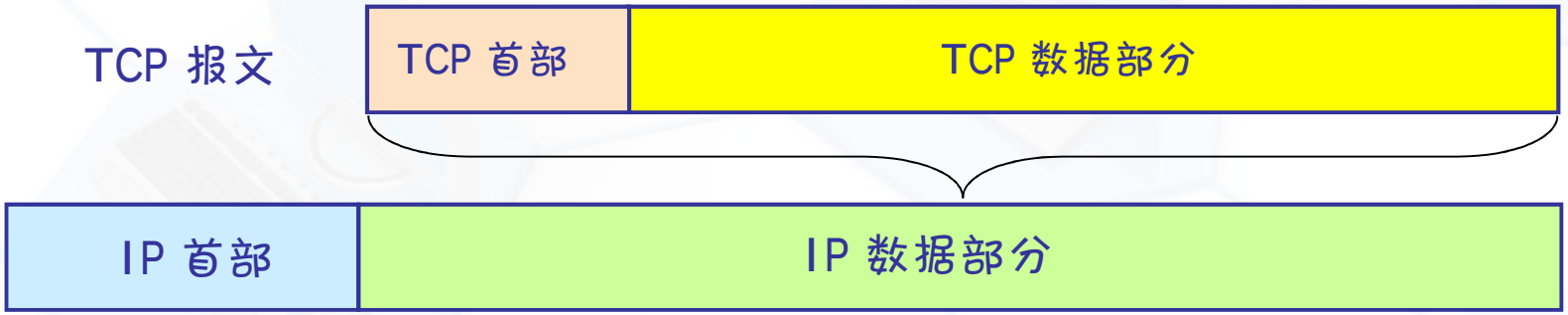




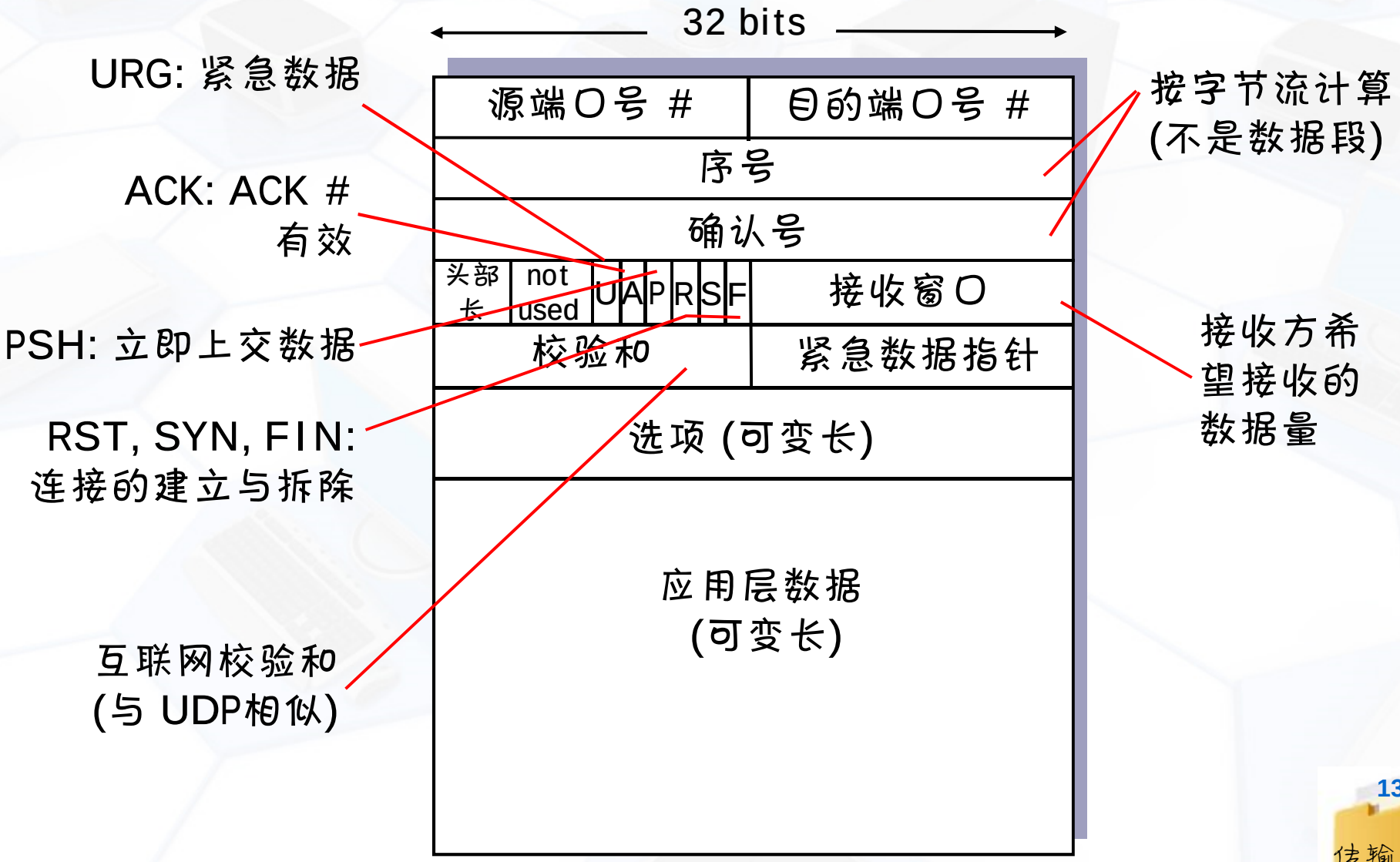
使用相同端口号访问同一个Web Server

三、传输控制协议TCP

- TCP常用于一次传输要交换大量报文的情形，如文件传输、远程登录等。它提供面向连接的流传输。
- TCP的报文段格式



1. TCP 报文结构

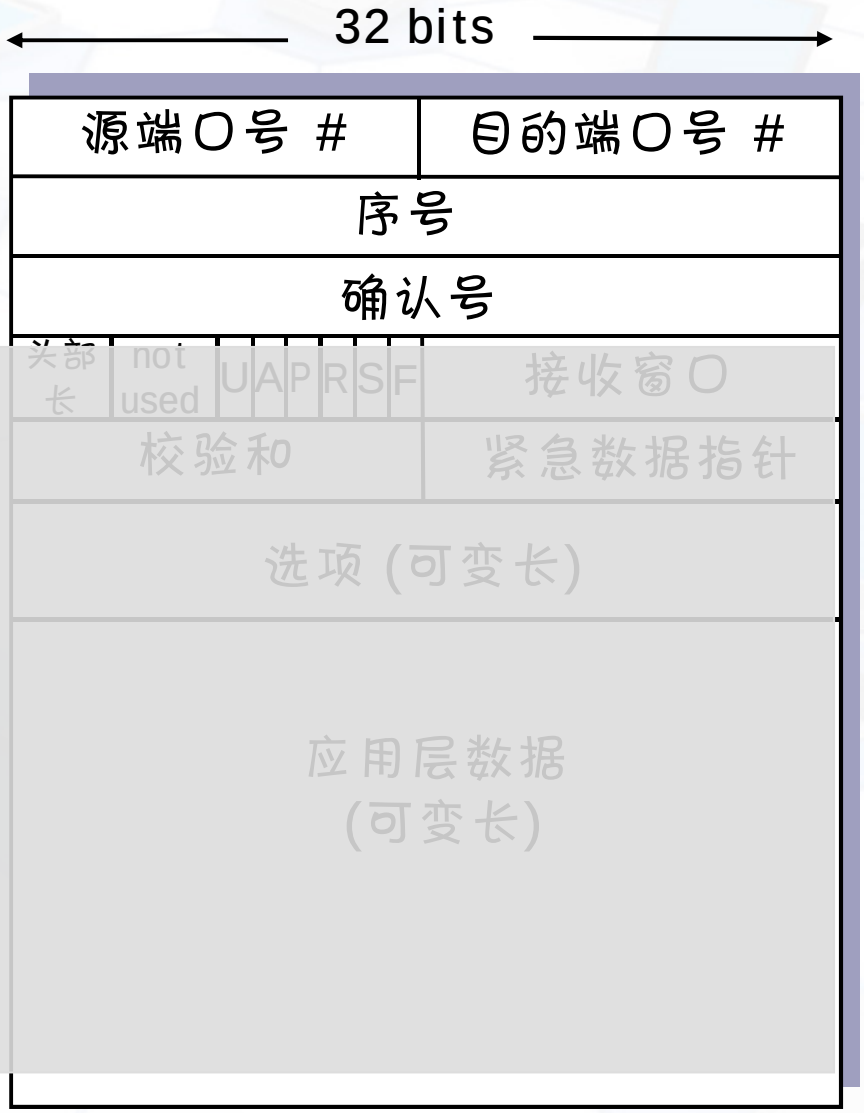

32 bits
源端口号 #
目的端口号 #
序号
确认号
头部长
not used
U
A
P
R
S
F
接收窗口
校验和
紧急数据指针
选项 (可变长)
应用层数据
(可变长)
URG: 紧急数据
ACK: ACK #
有效
PSH: 立即上交数据
RST, SYN, FIN:
连接的建立与拆除
互联网校验和
(与 UDP相似)
按字节流计算
(不是数据段)
接收方希
望接收的
数据量

§ 源端口和目的端口：各占2个字节，标明了一个连接的两个端点

§ 序号和确认号：各占4个字节，共同用于TCP服务中的确认和差错控制

ü 序号表示该报文段在发送方的数据流中的位置。TCP把数据流中的每一个字节都编上一个序号，该字段的值是指本报文段发送的数据的首字节的序号

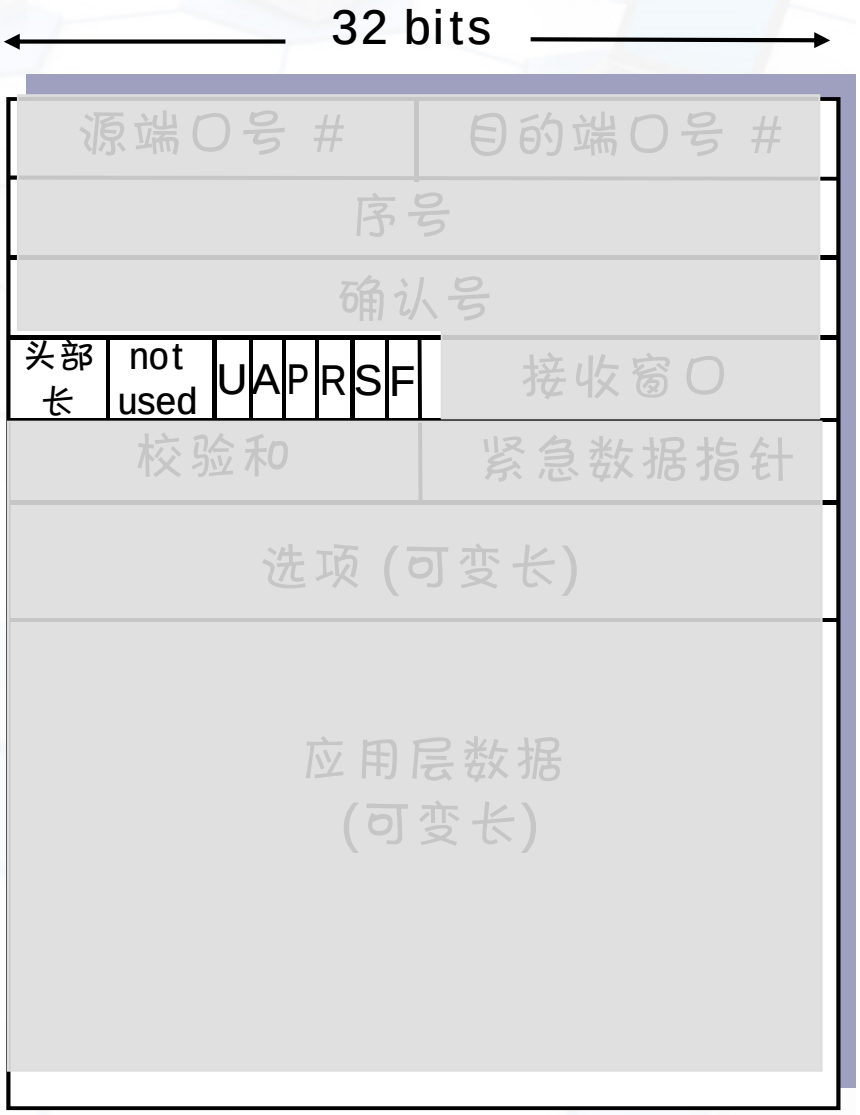
ü 确认号指下一个期望接收的TCP报文段数据的首字节序号



§ 头部长度：占4位，TCP头部长以32位字长为单位，接收方可以根据该字段确定TCP数据在段中的开始位置

§ 保留：未用的6位，为将来的应用而保留，目前设置为0

§ 控制位：占6位，TCP报文段有多种应用，如建立或关闭连接、传输数据、携带确认等，这些控制位用于给出与报文段的作用及处理有关的控制信息(见下表)



控制位	该位置“1”的含义
URG	紧急数据指针字段中的数据有意义
ACK	确认字段中的数据有意义
PSH	也称为“推”，请求急迫操作。指出接收方不必等待一定量的数据再向应用提供数据，而是立即提供数据
RST	也称为“复位”，在有异常情况发生时，发送方通知接收方暂时终止连接，释放与连接有关的缓冲区，中断TCP传输
SYN	也称为“同步”，与ACK合用以建立初始连接，如SYN=1，ACK=0表示连接请求；SYN=1，ACK=1表示同意建立连接
FIN	也称为“完成”，标志本次TCP连接的最后报文段

TCP首部各控制位的含义

§校验和：占2个字节，用于对TCP首部和数据进行校验

§紧急指针：占2个字节，当URG位置1时，表示有紧急数据，该字段给出从当前顺序号到紧急数据位置的偏移量

§选项：可变长度，提供一种增加额外设置的方法，如最大TCP报文段的大小约定

§填充：当TCP首部长度不是32位字长的整数倍时，需要加以填充

2. 端口

- 传输协议要赋给每个服务一个唯一的标识，指明连接双方的端点，称作“套接字”（Socket）
- 套接字=（IP地址，端口号）
- 每个TCP连接用发送端和接收端的套接字标识符来标识，形如（Socket1，Socket2）
- 端口就是传输层服务访问点TSAP
- 端口的作用—让应用层的各种应用进程都能将其数据通过端口向下交付给传输层，以及让传输层知道应当将其报文段中的数据向上通过端口交付给应用层相应的进程
- 端口具有本地意义

§ 端口号的分配

知名端口— 数值一般为0~1023，被规定作为公共应用服务的端口，它们被保留用于一些标准的服务

传输层协议	应用层协议名称	协议内容	使用的端口号
TCP	FTP（控制）	文件传输协议	21
	FTP（数据）		20
	TELNET	远程登录	23
	HTTP	超文本传送协议	80
	SMTP	简单邮件传送协议	25
	POP3	接收邮件	110
UDP	DNS	域名服务	53
	SNMP	简单网络管理协议	161
	TFTP	简单文件传输协议	69
	OICQ	聊天软件	8000

一般端口—1023以上端口未做规定，即作为自由端口，用于随时分配给请求通信的客户进程

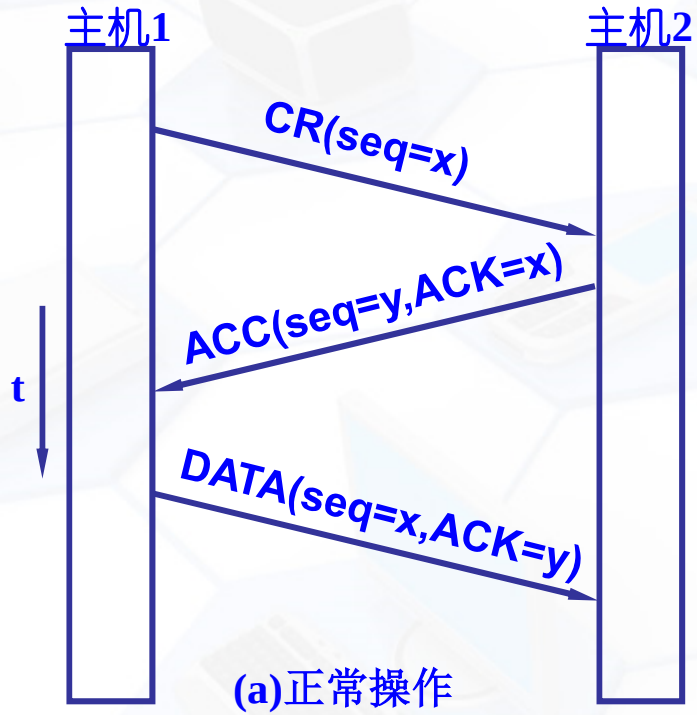
3. 连接的建立

§ 问题：被延误的重复分组

- ü 通信子网的不可靠性
- ü 通信子网中存在着延时和分组的丢失，以及由于延时和丢失而带来的重复分组
- ü 由于通信子网的尽力而为的传输原则，一个早已超时的分组最终还是到达了目的端，所以有必要将分组的生命周期限制在一个适当的范围内 。
- ü 连接建立时，如何处理过期分组，保证连接的唯一性是连接建立过程中首要考虑的问题

§ 常用的方法是： 三次握手

1) 正常连接的三次握手过程

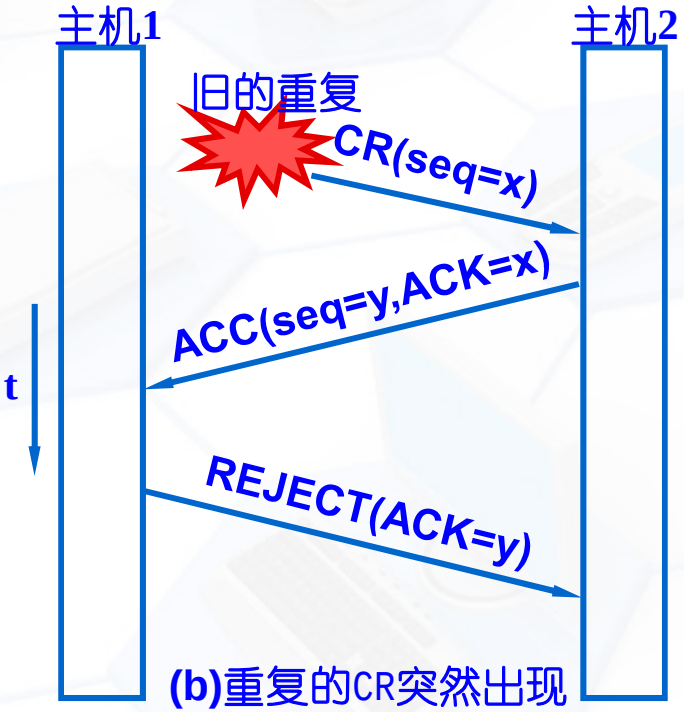


采用三次握手方法建立的正常情况

主机1发出连接请求序号为x(seq=x)，主机2应答接受连接请求主机1的请求，并声明自己的序列号为y(seq=y,ACK=x)，主机1收到确认后发送第一个数据TPDU并确认主机2的序列号(seq=x,ACK=y)，至此，整个连接建立过程正常结束，数据传输已正式开始

CR：Connection Request（连接请求）
ACC：ACK Connection（接受连接）

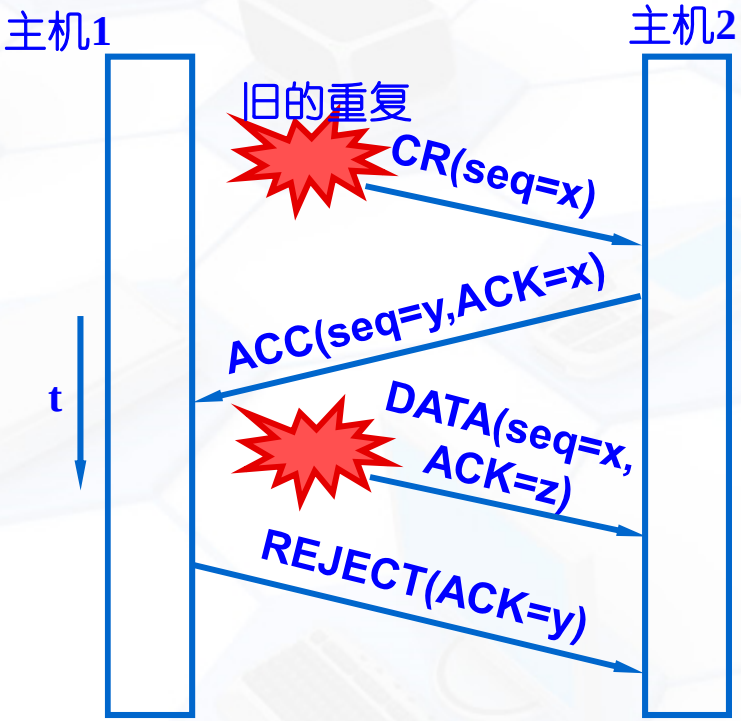
2) 非正常的连接建立过程1
—出现延迟的重复TPDU时三次握手的工作过程



采用三次握手建立的第二种情况

来自一个已经释放连接的主机1的延迟重复的连接请求，该TPDU在主机1毫不知晓的情况下到达主机2。主机2通过向主机1发送一个接受连接请求的TPDU来响应该TPDU，并声明自己的序号为y(seq=y,ACK=x)，主机1收到这个确认后感到莫名其妙并当即拒绝，主机2收到了主机1 的拒绝才意识到自己受到了延时的重复TPDU的欺骗并放弃该连接。据此，延时的重复请求将不会产生不良后果

3) 非正常的连接建立过程2
——子网中同时有作废的CR和ACC的情况



(c)重复的CR和重复的ACK

采用三次握手建立的第三种情况

与上例一样，主机2收到了一个延时的CR并做了确认应答。在这里，关键是要认识到主机2已经声明使用y作为从主机2到主机1进行数据传输的初始序号，因此主机2十分清楚在正常情况下，主机1的数据传输应捎带对y确认的TPDU。于是，当第二个延时的TPDU到达主机2时，主机2根据它确认的是序号z而不是y知道这也是一个过时的重复TPDU。因此也不会无故建立无人要求的连接

4. 连接的释放

连接的释放包括非对称释放和对称释放两种

§ 非对称释放

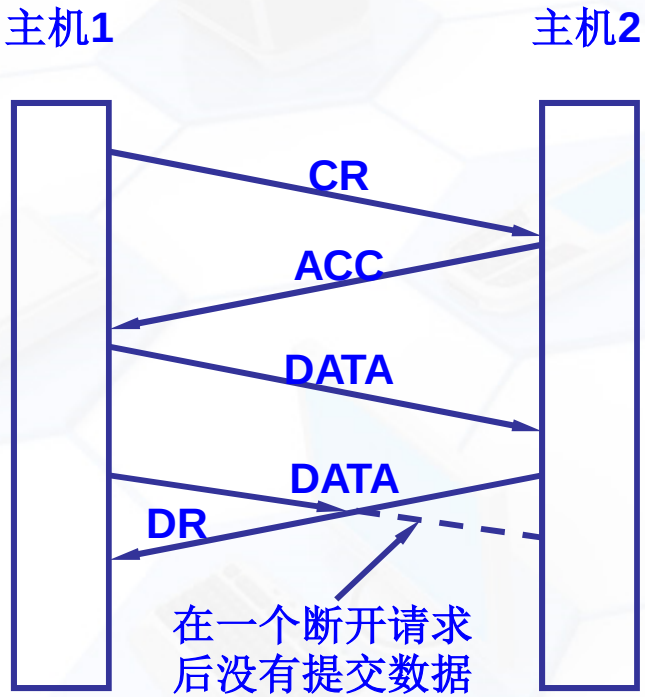
- ü 一方中止连接，则连接即告中断
- ü 缺陷：可能导致数据丢失

§ 对称释放

- ü A提出中止请求，B同意即中止
- ü 问题：B如何知道A 收到了它的确认?

1) 非对称释放

一方中止连接，则连接即告中断



突然释放连接将造成数据丢失

当连接建立后，主机1发送了一个数据TPDU并正确抵达主机2，接着，主机1发送了第二个数据TPDU，然而，主机2在收到第二个TPDU之前先突然发出了DISCONNECT(释放连接请求)，结果是连接立即被释放，数据被丢失

缺陷：可能导致数据丢失

2) 对称释放

A提出中止请求，B也同意即中止

§ 问题：B如何知道A 收到了它的确认?

- ü 对称释放方式适用于每个用户进程有固定数量的数据需要发送，而且清楚地知道何时发送完毕的情况
- ü 其他情况下，决定所有工作是否已经完成，连接是否应该释放，可能是没有把握的
- ü 可以假想一种协议：

A说：“ 我发送完了。你呢?”

如果B响应：“ 我也发送完了。再见。”

A收到了B的确认，连接便可以被安全释放

§ 想完全确信对方也准备释放连接时再释放连接是不可能的，因为没有百分之百可靠的协议

§ 对称释放的正常情况

三次握手的连接正常释放情况

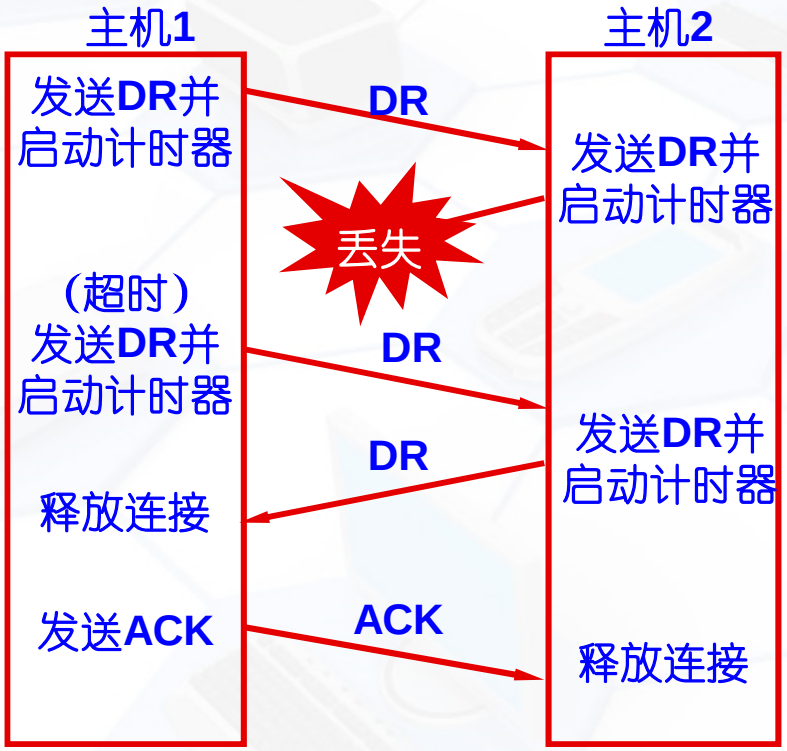
主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，主机1 在计时器没有超时前收到主机2的DR，便正式释放连接并发送ACK，主机2也在计时器没有超时前收到主机1 的ACK，于是也释放了连接，至此整个数据传输过程，包括建立连接、传输数据和释放连接的过程正常结束

§ 最后的确认TPDU丢失

最后的确认TPDU丢失的情况

主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1 的DR后同意释放连接，也发送DR并启动计时器，主机1 在计时器没有超时前收到主机2 的DR，便正式释放连接并发送ACK，然而主机2在计时器超时后还未收到主机1 的ACK，但是由于已经超时，于是也释放了连接

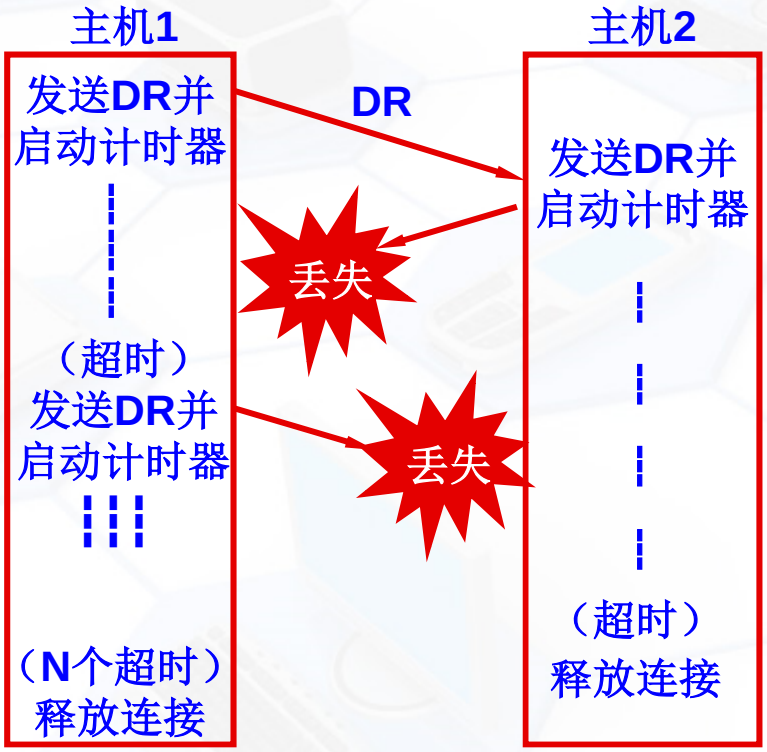
§ 应答丢失



应答丢失的情况

主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1 的DR后同意释放连接，也发送DR并启动计时器，然而，主机1 在计时器超时后还未收到主机2 的DR，于是又重新发送DR并启动计时器，下面便是一个正常的三次握手，并最后正常释放连接，即整个数据传输过程正常结束

§ 应答丢失以及后续的DR丢失



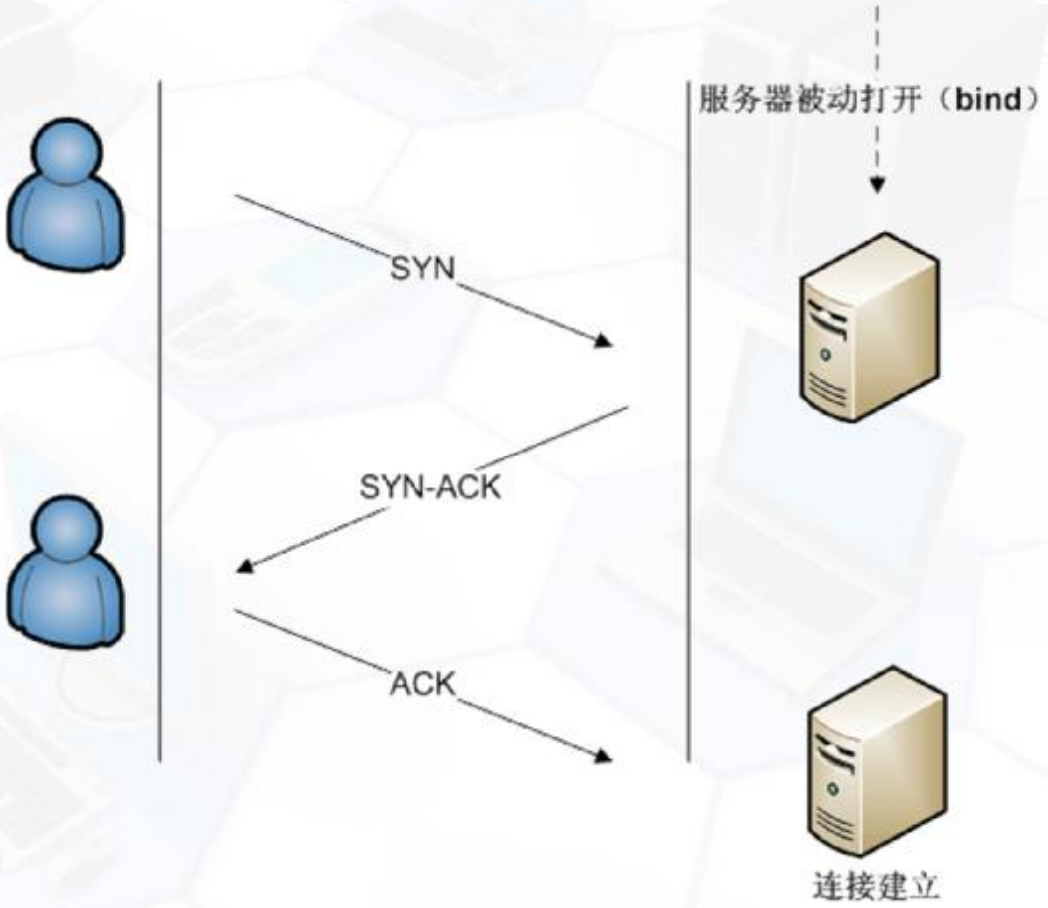
应答丢失以及后续的DR丢失

主机1在结束数据传输后决定释放连接，于是发送DR并启动计时器，主机2在收到主机1的DR后同意释放连接，也发送DR并启动计时器，然而，紧接着的一段时间内，线路遇到了灾难性的干扰，无论是哪一方的超时重发的TPDU都不能到达对方，最终，接收方计时器的超时而也释放连接，发送方经过n次重发和超时后只能无奈地放弃努力并释放连接

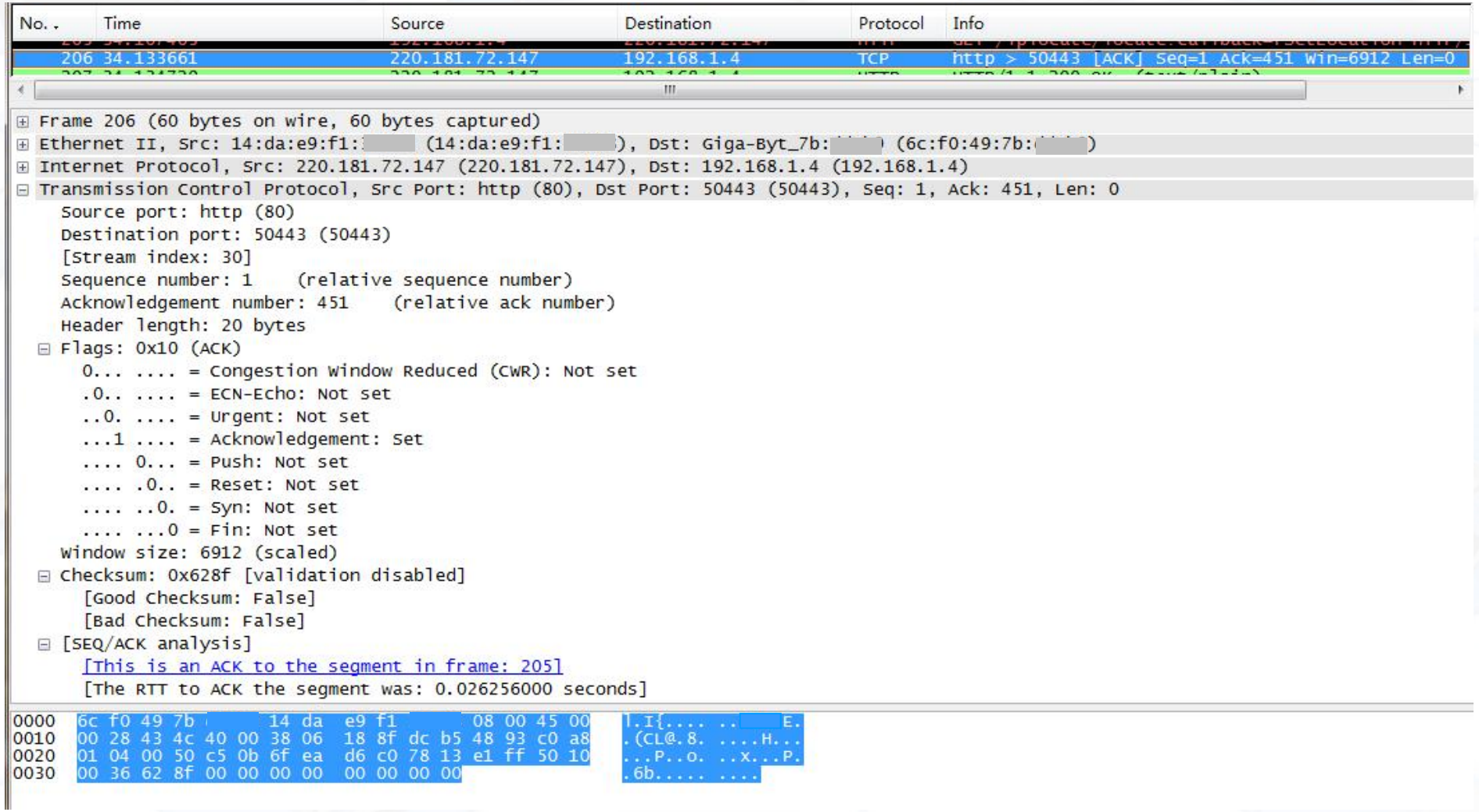
5. TCP的传输策略

建立在TCP协议上的传输可以分为两类

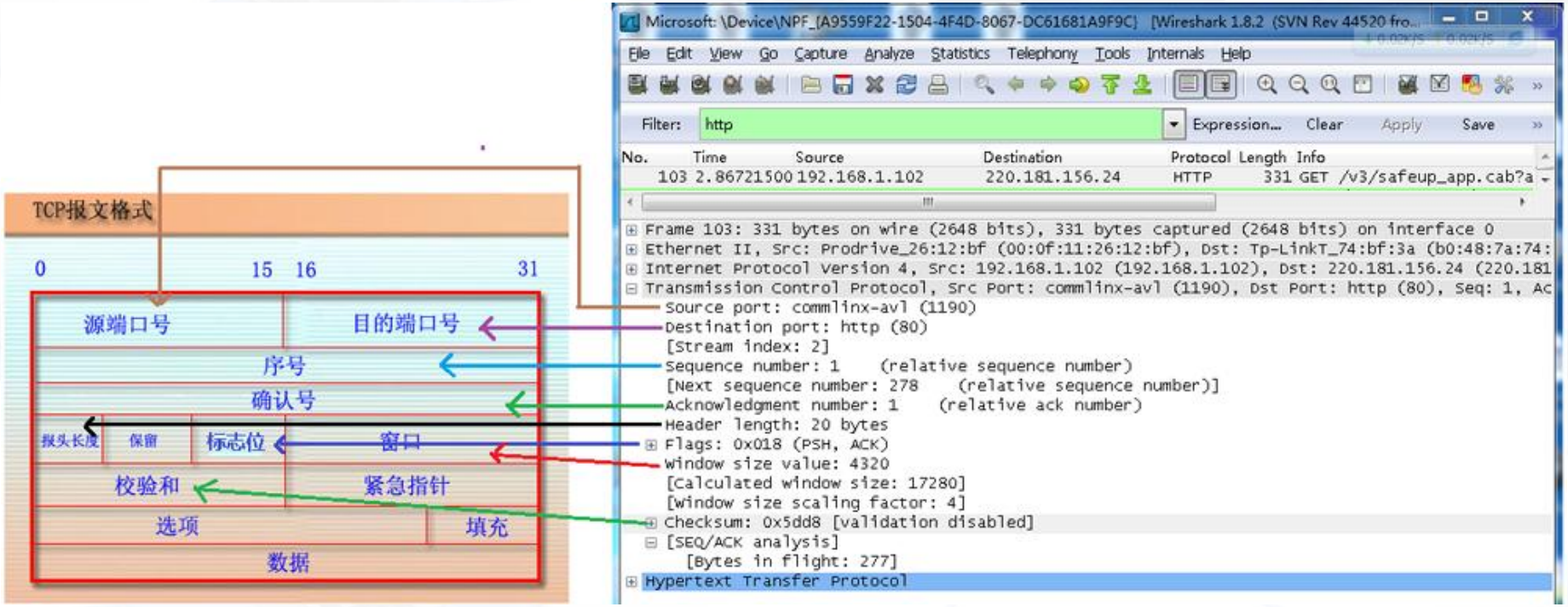
- TCP的交互数据流
  - 捎带ACK
  - Nagle算法
- TCP的成块数据流
  - 传输数据时确认的问题
  - 滑动窗口
  - 数据拥塞控制



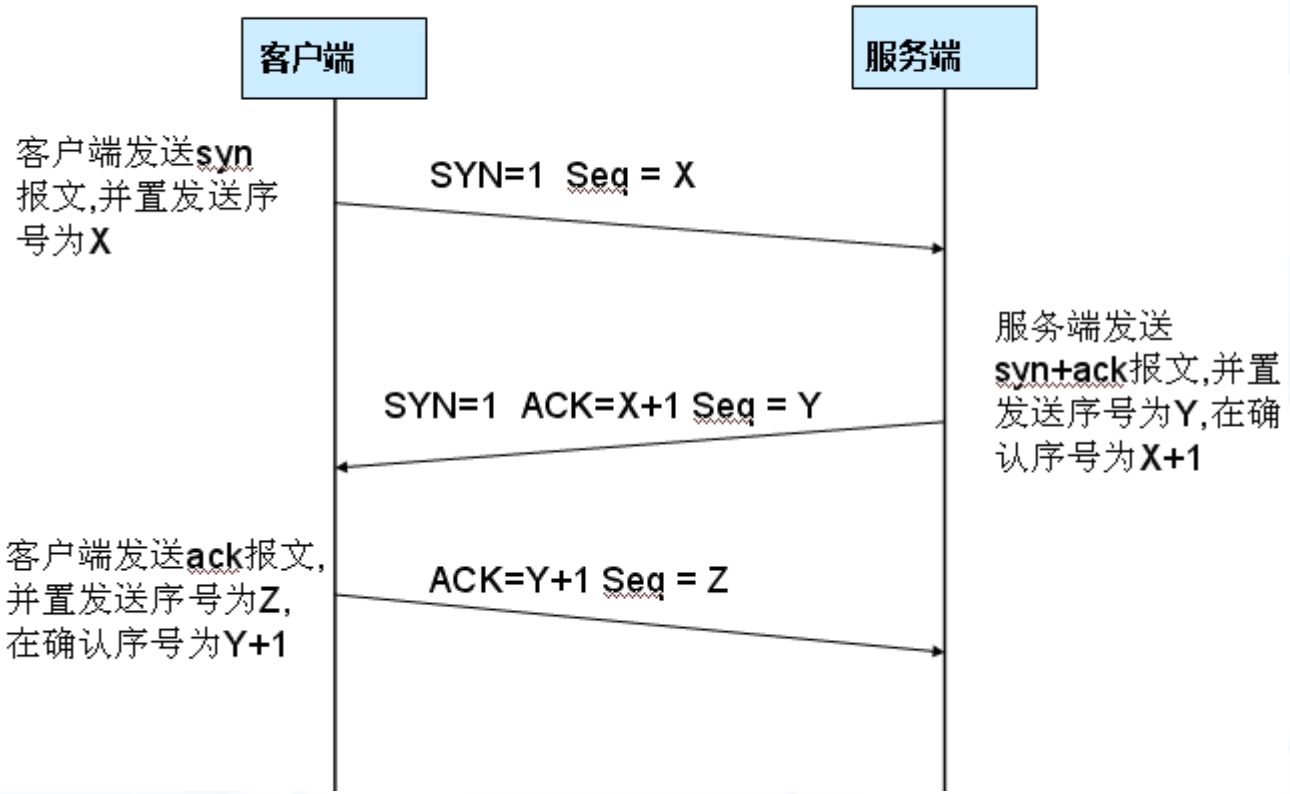
TCP的三次握手连接

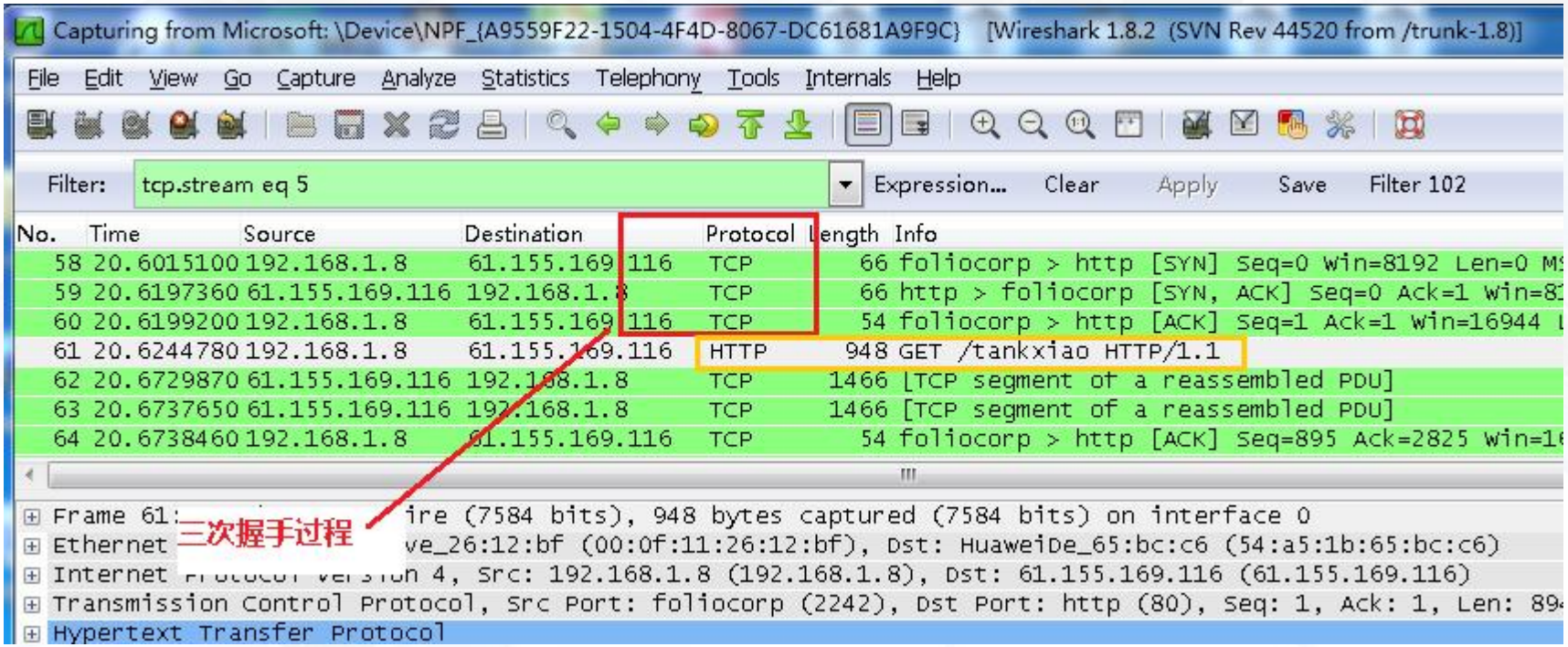

No. Time Source Destination Protocol Info
206 34.133661 220.181.72.147 192.168.1.4 TCP http > 50443 [ACK] Seq=1 Ack=451 Win=6912 Len=0
Frame 206 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 14:da:e9:f1: (14:da:e9:f1:), Dst: Giga-Byt_7b: (6c:f0:49:7b:)
Internet Protocol, Src: 220.181.72.147 (220.181.72.147), Dst: 192.168.1.4 (192.168.1.4)
Transmission Control Protocol, Src Port: http (80), Dst Port: 50443 (50443), Seq: 1, Ack: 451, Len: 0
Source port: http (80)
Destination port: 50443 (50443)
[Stream index: 30]
Sequence number: 1 (relative sequence number)
Acknowledgement number: 451 (relative ack number)
Header length: 20 bytes
Flags: 0x10 (ACK)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...1 .... = Acknowledgement: Set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
.... ..0. = Syn: Not set
....0 = Fin: Not set
Window size: 6912 (scaled)
Checksum: 0x628f [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]
[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 205]
[The RTT to ACK the segment was: 0.026256000 seconds]
0000 6c f0 49 7b 14 da e9 f1 08 00 45 00
0010 00 28 43 4c 40 00 38 06 18 8f dc b5 48 93 c0 a8
0020 01 04 00 50 c5 0b 6f ea d6 c0 78 13 e1 ff 50 10
0030 00 36 62 8f 00 00 00 00 00 00 00 00

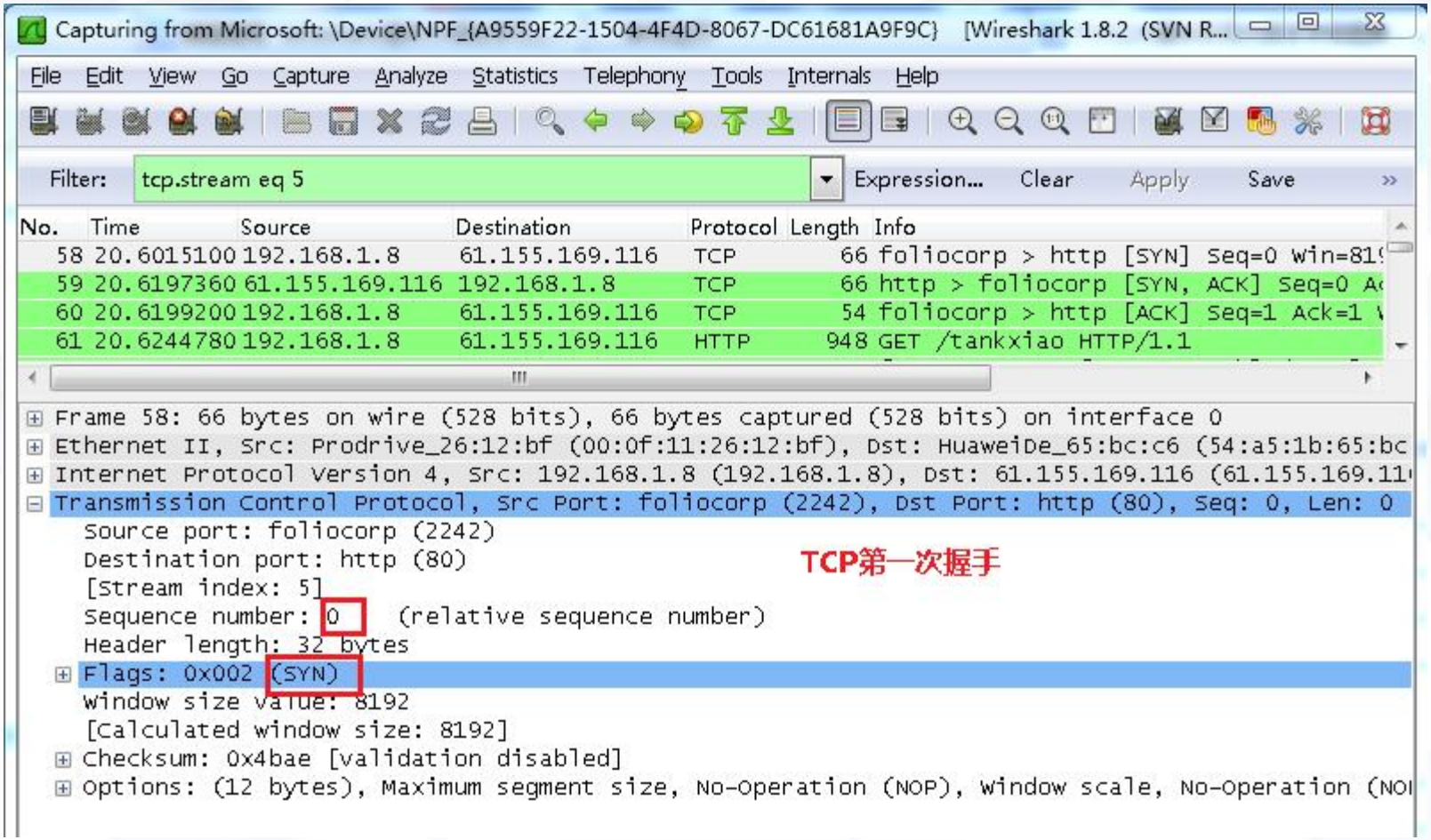
TCP报文实例

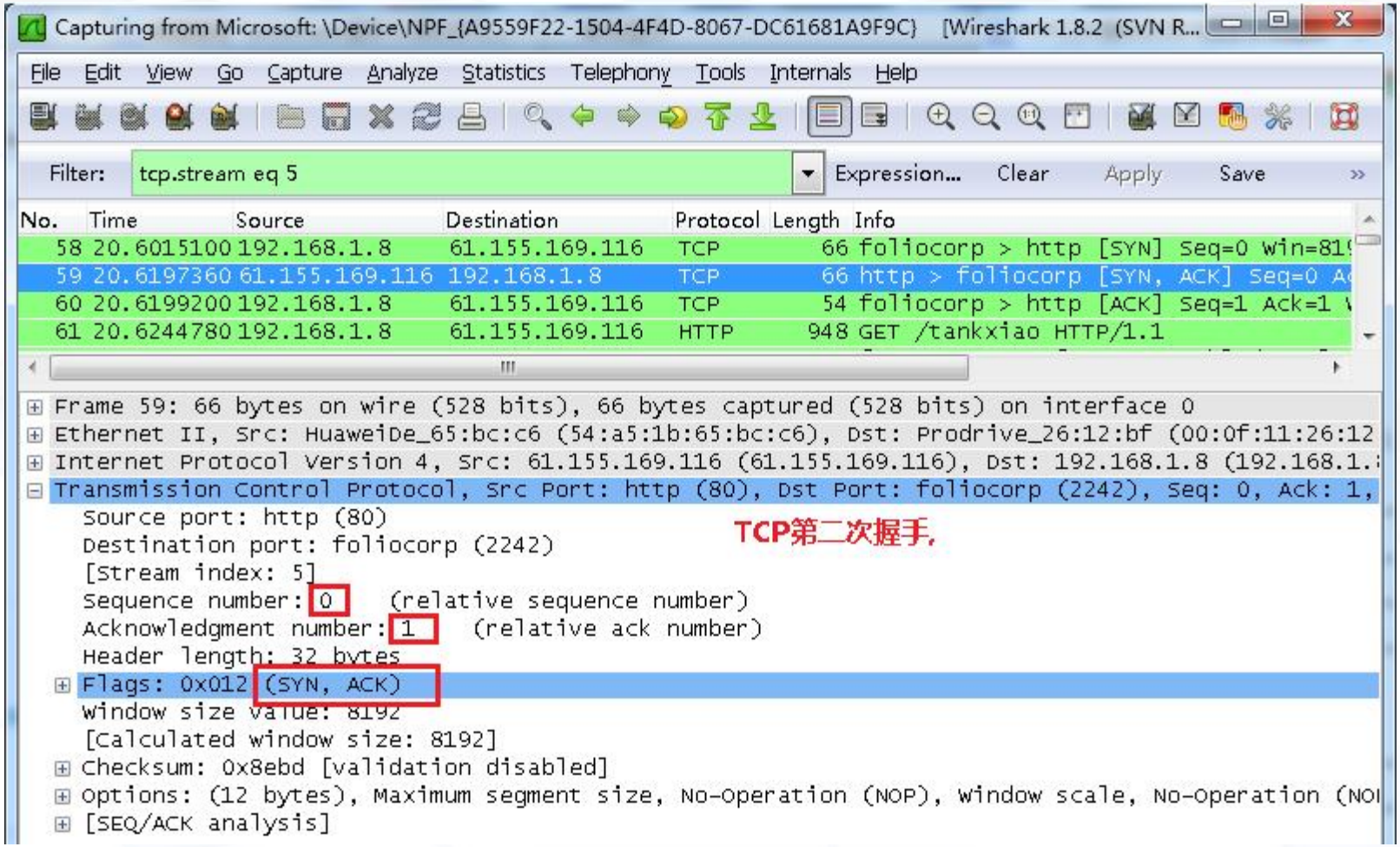

TCP报文格式
0
15
16
31
源端口号
目的端口号
序号
确认号
报头长度
保留
标志位
窗口
校验和
紧急指针
选项
填充
数据
Microsoft: \Device\NPF_{A9559F22-1504-4F4D-8067-DC61681A9F9C} [Wireshark 1.8.2 (SVN Rev 44520 fro...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
103 2.86721500 192.168.1.102 220.181.156.24 HTTP 331 GET /v3/safeup_app.cab?a
Frame 103: 331 bytes on wire (2648 bits), 331 bytes captured (2648 bits) on interface 0
Ethernet II, Src: Prodrive_26:12:bf (00:0f:11:26:12:bf), Dst: Tp-LinkT_74:bf:3a (b0:48:7a:74:
Internet Protocol Version 4, Src: 192.168.1.102 (192.168.1.102), Dst: 220.181.156.24 (220.181
Transmission Control Protocol, Src Port: commlinx-avl (1190), Dst Port: http (80), Seq: 1, Ac
Source port: commlinx-avl (1190)
Destination port: http (80)
[Stream index: 2]
Sequence number: 1 (relative sequence number)
[Next sequence number: 278 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x018 (PSH, ACK)
Window size value: 4320
[Calculated window size: 17280]
[Window size scaling factor: 4]
Checksum: 0x5dd8 [validation disabled]
[SEQ/ACK analysis]
[Bytes in flight: 277]
Hypertext Transfer Protocol

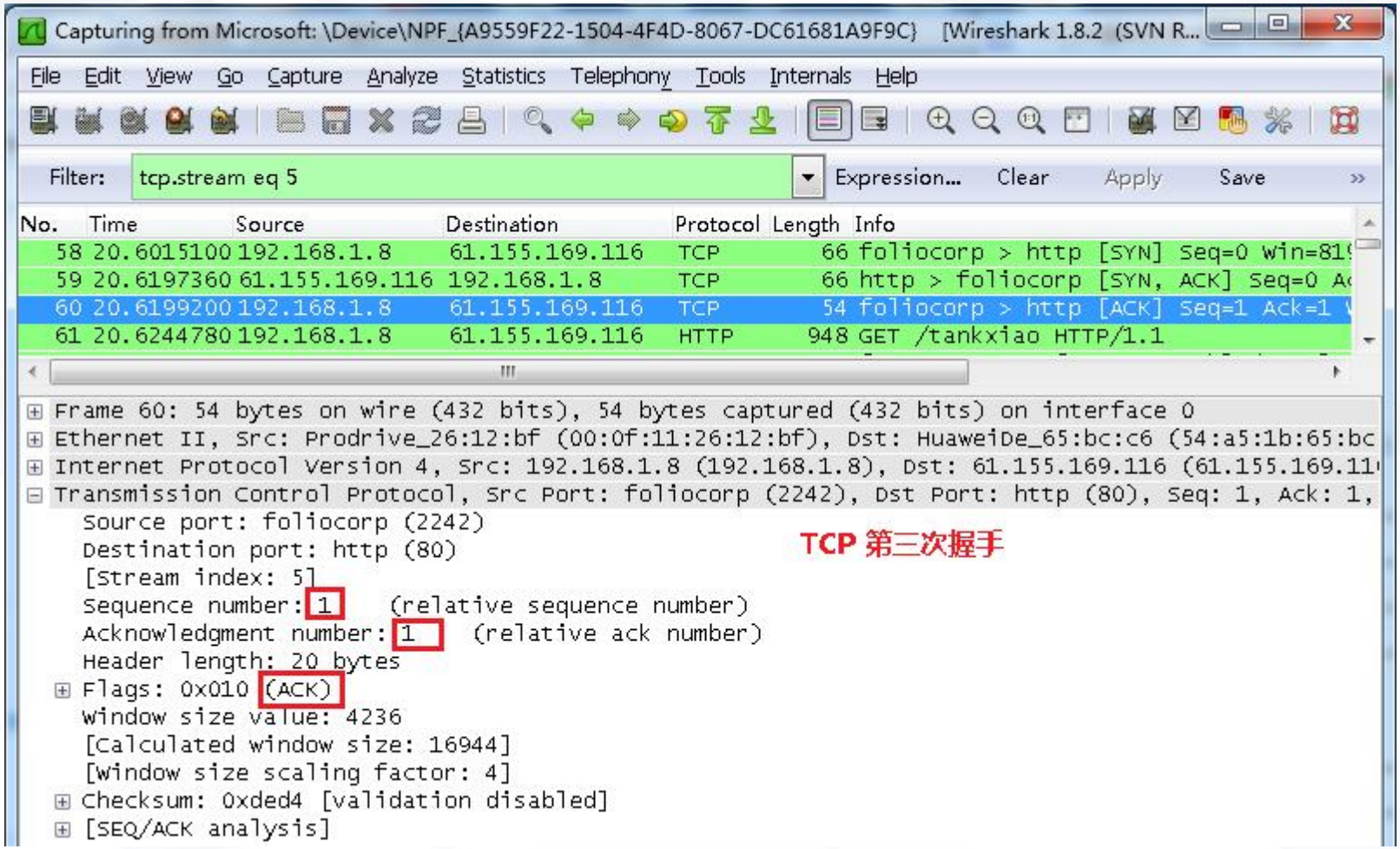
TCP 三次握手

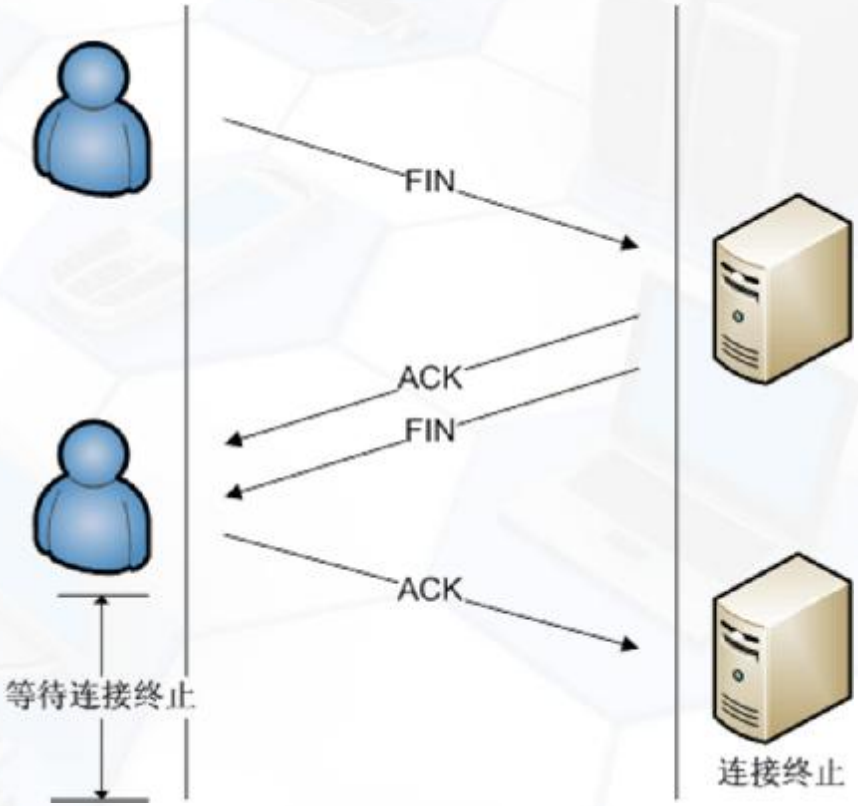

客户端
服务端
客户端发送syn报文,并置发送序号为X
SYN=1 Seq = X
服务端发送syn+ack报文,并置发送序号为Y,在确认序号为X+1
SYN=1 ACK=X+1 Seq = Y
客户端发送ack报文,并置发送序号为Z,在确认序号为Y+1
ACK=Y+1 Seq = Z


Capturing from Microsoft: \Device\NPF_{A9559F22-1504-4F4D-8067-DC61681A9F9C} [Wireshark 1.8.2 (SVN Rev 44520 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 5
Expression... Clear Apply Save Filter 102
No. Time Source Destination Protocol Length Info
58 20.6015100 192.168.1.8 61.155.169.116 TCP 66 foliocorp > http [SYN] Seq=0 Win=8192 Len=0
59 20.6197360 61.155.169.116 192.168.1.8 TCP 66 http > foliocorp [SYN, ACK] Seq=0 Ack=1 Win=8
60 20.6199200 192.168.1.8 61.155.169.116 TCP 54 foliocorp > http [ACK] Seq=1 Ack=1 Win=16944
61 20.6244780 192.168.1.8 61.155.169.116 HTTP 948 GET /tankxiao HTTP/1.1
62 20.6729870 61.155.169.116 192.168.1.8 TCP 1466 [TCP segment of a reassembled PDU]
63 20.6737650 61.155.169.116 192.168.1.8 TCP 1466 [TCP segment of a reassembled PDU]
64 20.6738460 192.168.1.8 61.155.169.116 TCP 54 foliocorp > http [ACK] Seq=895 Ack=2825 Win=16
三次握手过程
Frame 61: ire (7584 bits), 948 bytes captured (7584 bits) on interface 0
Ethernet ve_26:12:bf (00:0f:11:26:12:bf), Dst: HuaweiDe_65:bc:c6 (54:a5:1b:65:bc:c6)
Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 61.155.169.116 (61.155.169.116)
Transmission Control Protocol, Src Port: foliocorp (2242), Dst Port: http (80), Seq: 1, Ack: 1, Len: 89
Hypertext Transfer Protocol


Capturing from Microsoft: \Device\NPF_{A9559F22-1504-4F4D-8067-DC61681A9F9C} [Wireshark 1.8.2 (SVN R...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 5
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
58 20.6015100 192.168.1.8 61.155.169.116 TCP 66 foliocorp > http [SYN] Seq=0 Win=81
59 20.6197360 61.155.169.116 192.168.1.8 TCP 66 http > foliocorp [SYN, ACK] Seq=0 A
60 20.6199200 192.168.1.8 61.155.169.116 TCP 54 foliocorp > http [ACK] Seq=1 Ack=1
61 20.6244780 192.168.1.8 61.155.169.116 HTTP 948 GET /tankxiao HTTP/1.1
Frame 58: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Prodrive_26:12:bf (00:0f:11:26:12:bf), Dst: HuaweiDe_65:bc:c6 (54:a5:1b:65:bc
Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 61.155.169.116 (61.155.169.11
Transmission Control Protocol, Src Port: foliocorp (2242), Dst Port: http (80), Seq: 0, Len: 0
Source port: foliocorp (2242)
Destination port: http (80)
[Stream index: 5]
Sequence number: 0 (relative sequence number)
Header length: 32 bytes
Flags: 0x002 (SYN)
Window size value: 8192
[Calculated window size: 8192]
Checksum: 0x4bae [validation disabled]
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOI
TCP第一次握手


Capturing from Microsoft: \Device\NPF_{A9559F22-1504-4F4D-8067-DC61681A9F9C} [Wireshark 1.8.2 (SVN R...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 5
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
58 20.6015100 192.168.1.8 61.155.169.116 TCP 66 foliocorp > http [SYN] Seq=0 Win=81
59 20.6197360 61.155.169.116 192.168.1.8 TCP 66 http > foliocorp [SYN, ACK] Seq=0 A
60 20.6199200 192.168.1.8 61.155.169.116 TCP 54 foliocorp > http [ACK] Seq=1 Ack=1
61 20.6244780 192.168.1.8 61.155.169.116 HTTP 948 GET /tankxiao HTTP/1.1
Frame 59: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: HuaweiDe_65:bc:c6 (54:a5:1b:65:bc:c6), Dst: Prodrive_26:12:bf (00:0f:11:26:12
Internet Protocol Version 4, Src: 61.155.169.116 (61.155.169.116), Dst: 192.168.1.8 (192.168.1.
Transmission Control Protocol, Src Port: http (80), Dst Port: foliocorp (2242), Seq: 0, Ack: 1,
Source port: http (80)
Destination port: foliocorp (2242)
[Stream index: 5]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 1 (relative ack number)
Header length: 32 bytes
Flags: 0x012 (SYN, ACK)
Window size value: 8192
[Calculated window size: 8192]
Checksum: 0x8ebd [validation disabled]
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NO
[SEQ/ACK analysis]
TCP第二次握手,


Capturing from Microsoft: \Device\NPF_{A9559F22-1504-4F4D-8067-DC61681A9F9C} [Wireshark 1.8.2 (SVN R...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 5
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
58 20.6015100 192.168.1.8 61.155.169.116 TCP 66 foliocorp > http [SYN] Seq=0 Win=81
59 20.6197360 61.155.169.116 192.168.1.8 TCP 66 http > foliocorp [SYN, ACK] Seq=0 A
60 20.6199200 192.168.1.8 61.155.169.116 TCP 54 foliocorp > http [ACK] Seq=1 Ack=1
61 20.6244780 192.168.1.8 61.155.169.116 HTTP 948 GET /tankxiao HTTP/1.1
Frame 60: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: Prodrive_26:12:bf (00:0f:11:26:12:bf), Dst: HuaweiDe_65:bc:c6 (54:a5:1b:65:bc
Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 61.155.169.116 (61.155.169.11
Transmission Control Protocol, Src Port: foliocorp (2242), Dst Port: http (80), Seq: 1, Ack: 1,
Source port: foliocorp (2242)
Destination port: http (80)
[Stream index: 5]
Sequence number: 1 (relative sequence number)
Acknowledgment number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x010 (ACK)
Window size value: 4236
[Calculated window size: 16944]
[Window size scaling factor: 4]
Checksum: 0xded4 [validation disabled]
[SEQ/ACK analysis]
TCP 第三次握手



TCP的三次握手释放

6. TCP的拥塞控制

§ 拥塞发生的原因是“需求”大于“供给”

§ 主机如何知道拥塞

- ü 收到ICMP的源抑制报文
- ü 因报文丢失引起的超时

§ TCP的拥塞控制由以下4个核心部分组成

- ü 慢启动（Slow Start）
- ü 拥塞避免（Congestion Avoidance）
- ü 快速重传（Fast Retransmit）
- ü 快速恢复（Fast Recovery）

拥塞控制相关的参数

§拥塞窗口（Congestion Widnow）：拥塞控制的关键参数，它描述发送方根据自己估计的网络拥塞情况设置一次最多能发送的数据包数量

§接收方窗口（Receiver Window）：又叫通知窗口，是接收方根据目前自己的接收缓存大小给发送方预设的最新窗口大小

§发送窗口：发送方每次实际发送数据的窗口大小，其上限值是接收窗口和拥塞窗口的较小值

§慢启动阈值：拥塞控制中慢启动阶段和拥塞避免阶段的分界点，初始值通常设置为65535字节

§往返时延（RTT）：一个TCP数据包从发送方发送到接收方，发送方收到接收方确认的时间间隔

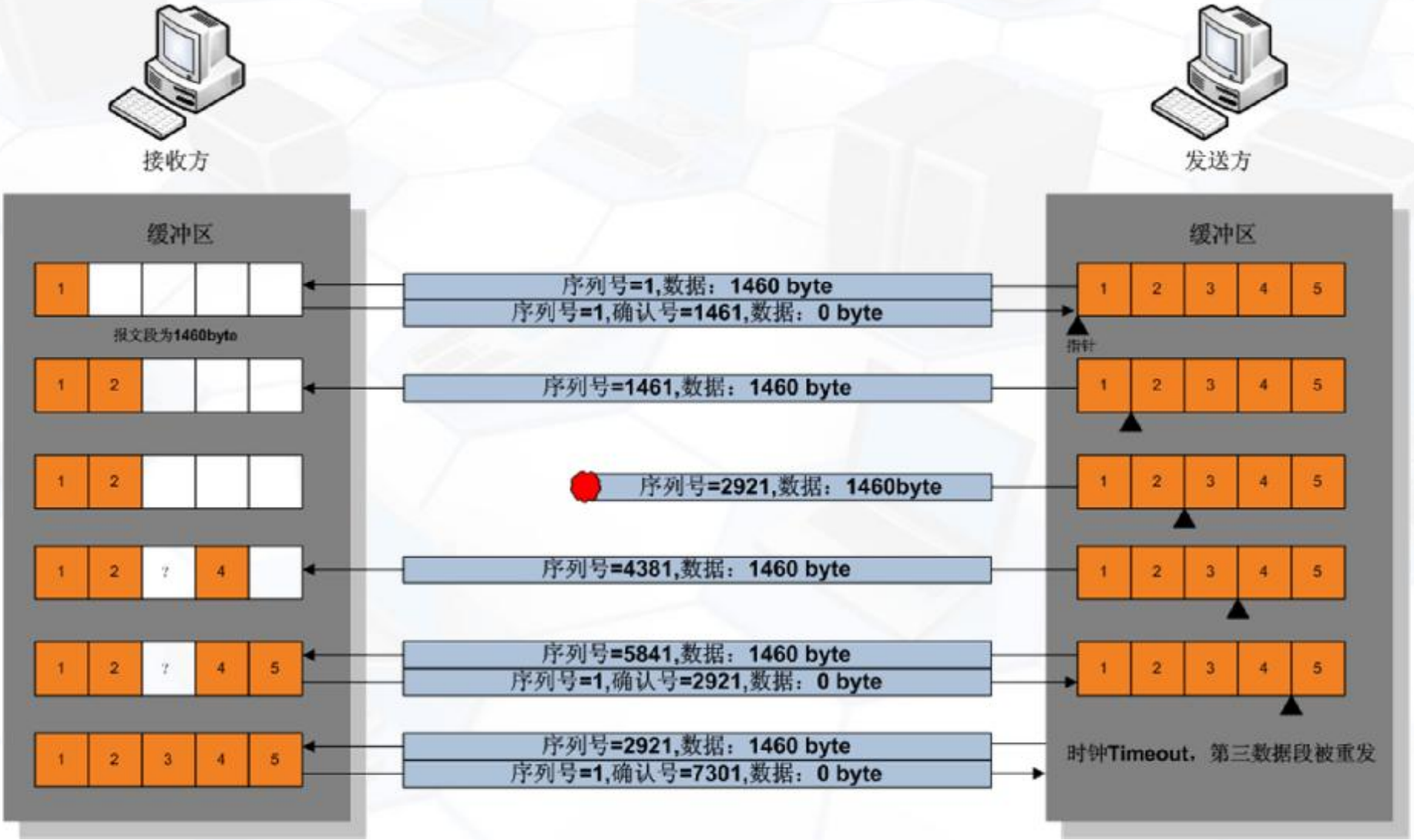
拥塞控制策略

§ 慢启动与拥塞避免

- ü 在不清楚环境的情况下向网络传送数据，要求TCP缓慢地探测网络以确定可用流量。拥塞窗口被初始化为一个数据包大小，发送方按发送窗口大小发送数据，每收到一个ACK确认，拥塞窗口就增加一个数据包发送量。显然，拥塞窗口的增长随RTT呈指数级增长
- ü 若拥塞窗口≤慢启动阈值，TCP重新进入慢启动过程
- ü 若拥塞窗口>慢启动阈值，TCP执行拥塞避免算法。拥塞避免是指在拥塞避免阶段使拥塞窗口按线性规律增长，使网络不容易出现拥塞

§ 快速重传和快速恢复

快速重传和快速恢复规定，不必等到超过，只要发送方收到3个或3个以上重复ACK时，就确定数据报已经被丢失，并重传数据报，同时将慢启动阈值设置为当前拥塞窗口的一半（这和慢启动过程相同），但是拥塞窗口不再设置为1，而是慢启动阈值+n × MSS。如果发送窗口还允许发送报文段，就按拥塞避免算法继续发报送文段。如果收到了确认新的报文的ACK，就将拥塞窗口缩小到慢启动阈值。也就是说，采用快速恢复过程时，慢启动算法只在TCP建立连接时才使用


接收方
发送方
缓冲区
缓冲区
1
报文段为1460byte
序列号=1,数据：1460 byte
序列号=1,确认号=1461,数据：0 byte
指针
序列号=1461,数据：1460 byte
序列号=2921,数据：1460byte
序列号=4381,数据：1460 byte
序列号=5841,数据：1460 byte
序列号=1,确认号=2921,数据：0 byte
序列号=2921,数据：1460 byte
序列号=1,确认号=7301,数据：0 byte
时钟Timeout，第三数据段被重发

http://histrory.visualland.net/view.php?path=content/TCP/Simu/sm_1_Reliable_Transmit.show&type=show

http://histrory.visualland.net/tcp_swnd.html

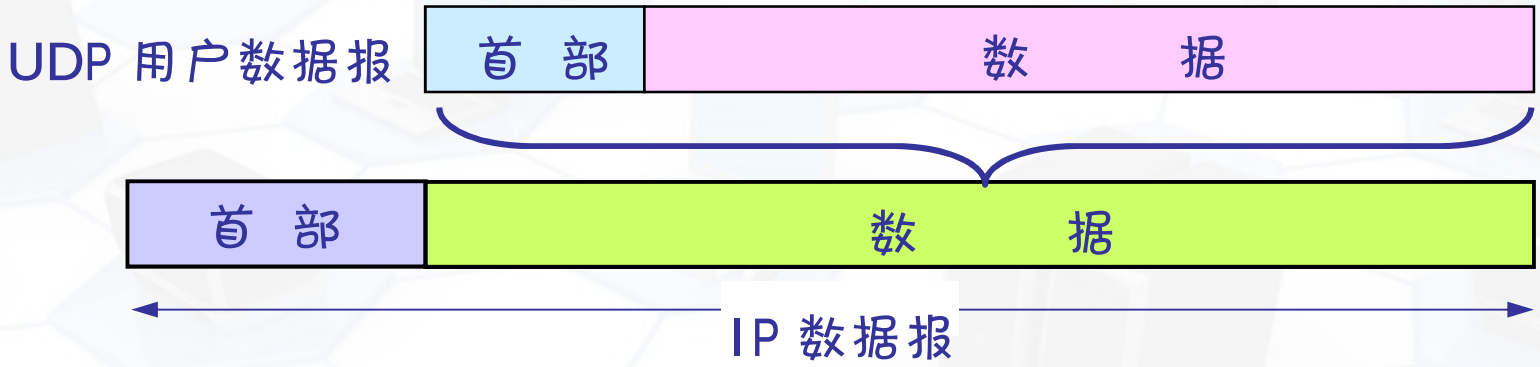
http://histrory.visualland.net/view.php

http://histrory.visualland.net/tcp_fast_retransmit.html

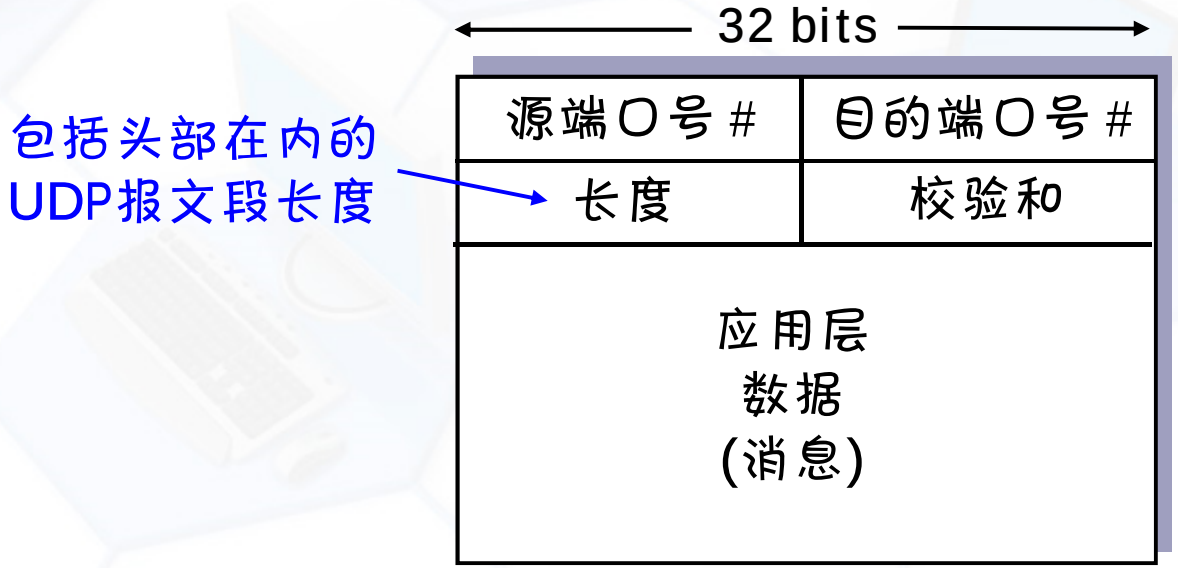
传输层

四、 用户数据报协议UDP

- § UDP提供的是不可靠的、无连接的数据传输服务
- § 使用UDP为传输层协议的网络应用其可靠性的问题需要由使用UDP的应用程序来解决
- § 常用在基于流的多媒体应用
  - ü容忍丢包
  - ü速率敏感
- § UDP的其他应用
  - üDNS
  - üSNMP
- § 基于UDP的可靠传输: 在应用层增加可靠性
  - ü 在应用层建立错误恢复机制



UDP 的封装

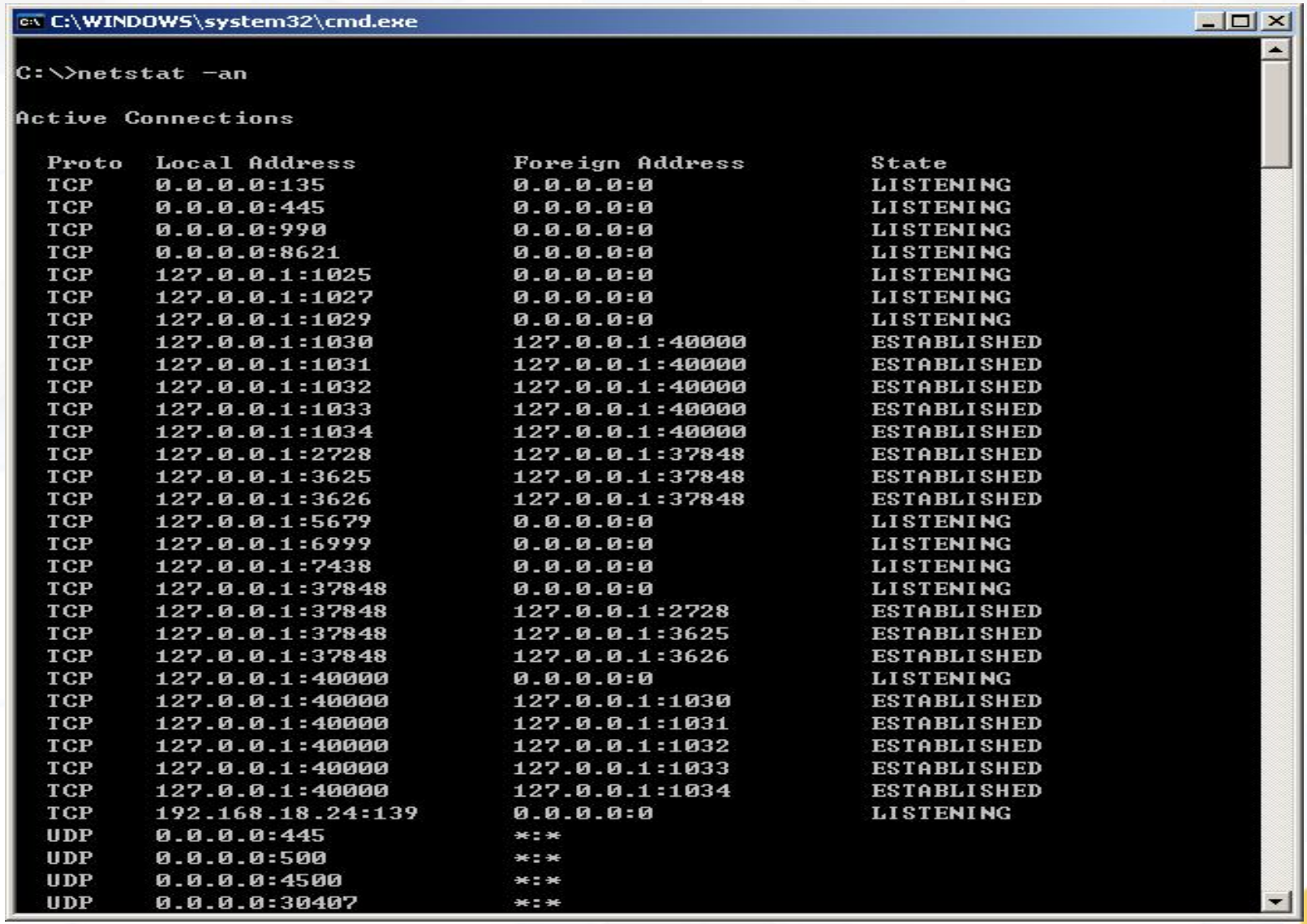


UDP 的数据格式

UDP与TCP相比较：

- 两者在如何实现信息的可靠传递方面有所不同
- TCP包含了可靠性保证机制，而UDP不提供数据传送的保证机制
- TCP在实际执行的过程中会占用大量的系统开销，严重地影响到传输速度。而UDP不考虑可靠性，将安全和排序等功能移交给上层应用来完成，极大地减少了时间，使速度得到了保证

应用实例：查看本机当前开放端口



ü Netstat 程序

显示活动的 TCP 连接、计算机侦听的端口、以太网统计信息、IP 路由表、IPv4 统计信息（对于IP、ICMP、TCP 和 UDP 协议）以及 IPv6 统计信息（对于 IPv6、ICMPv6、通过 IPv6 的 TCP以及通过 IPv6 的 UDP 协议）。使用时如果不带参数，netstat 显示活动的 TCP 连接

ü 语法

netstat [-a] [-e] [-n] [-o] [-p Protocol] [-r] [-s]
[Interval]

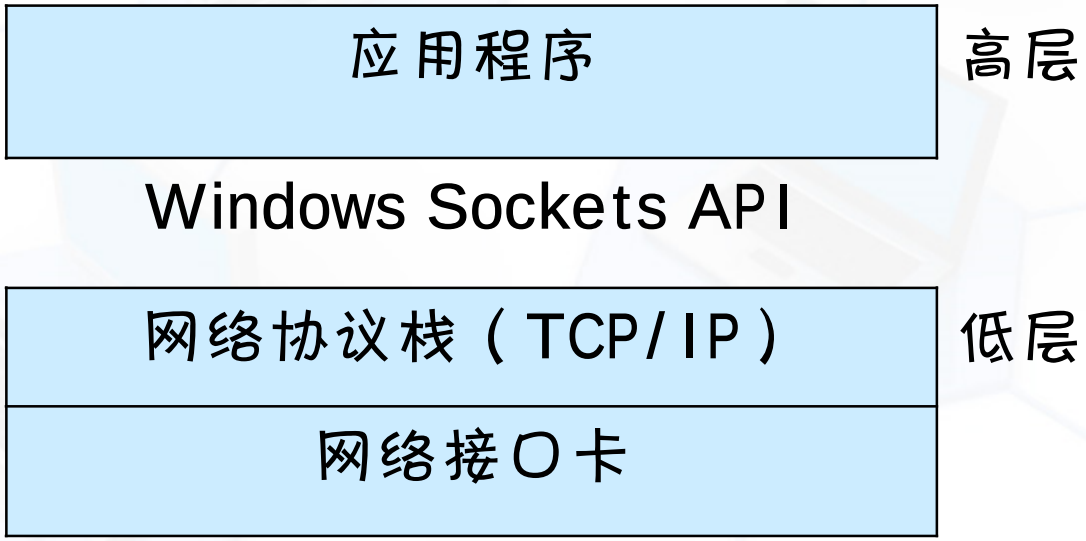
五、 套接字编程

1. 套接字接口

套接字API是用于构造网络应用程序的编程接口，它定义了应用程序与操作系统中传输协议进行交互时使用的一组操作，如图所示：

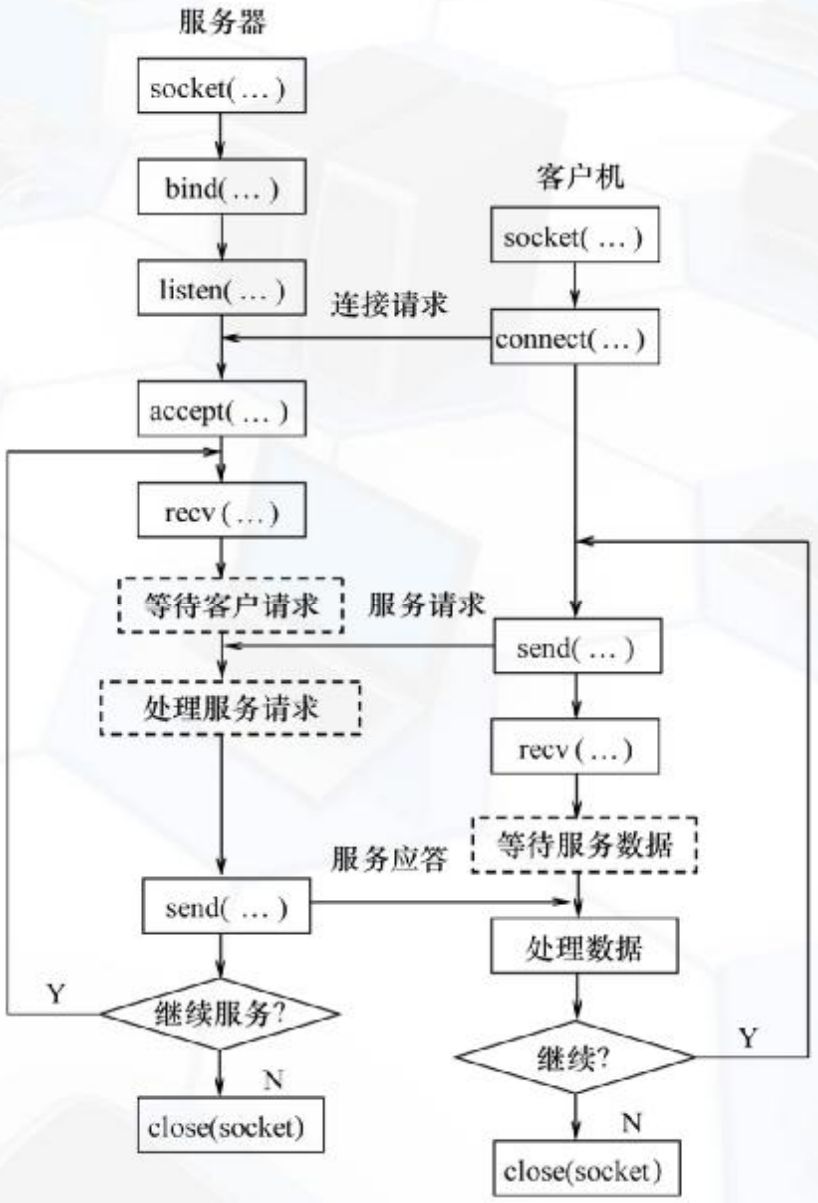
Windows Sockets规范

- 一套开放的、支持多种协议的Windows 下的网络编程接口，现在已成为Windows 网络编程的事实上的标准



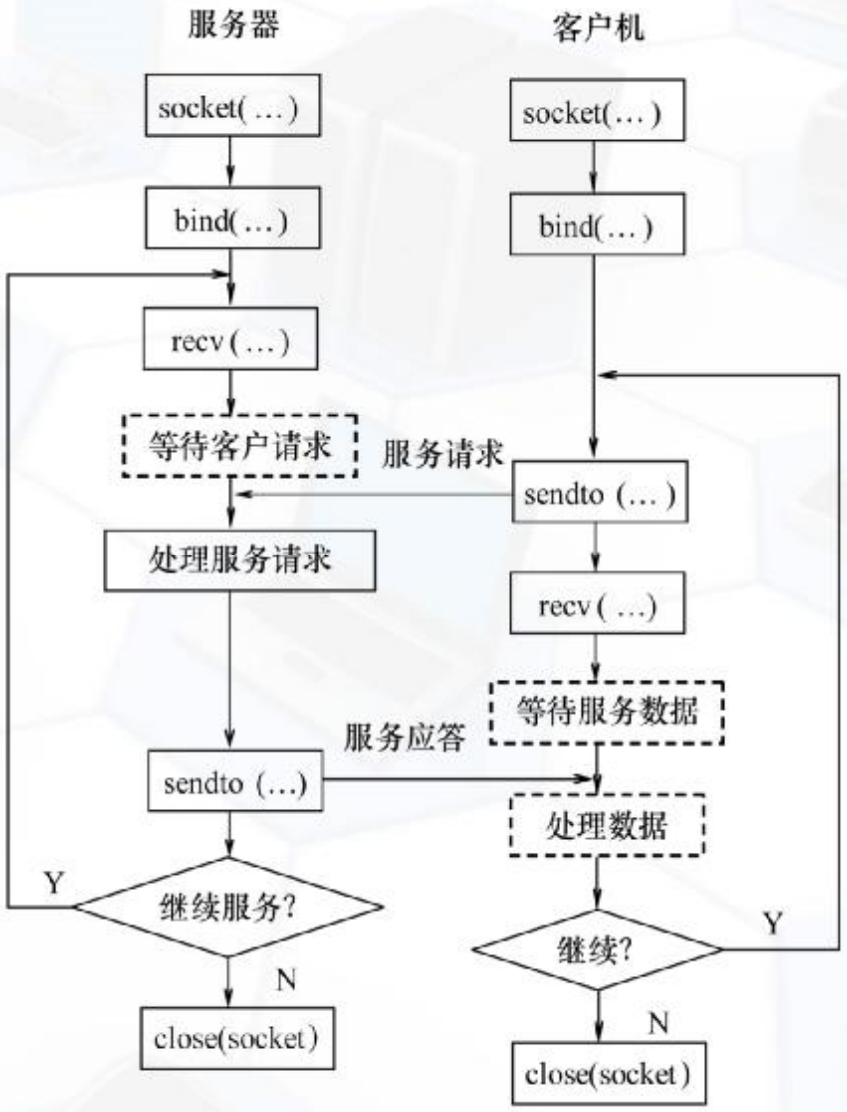
2. 面向连接的通信

面向连接套接口应用程序的工作过程如图:



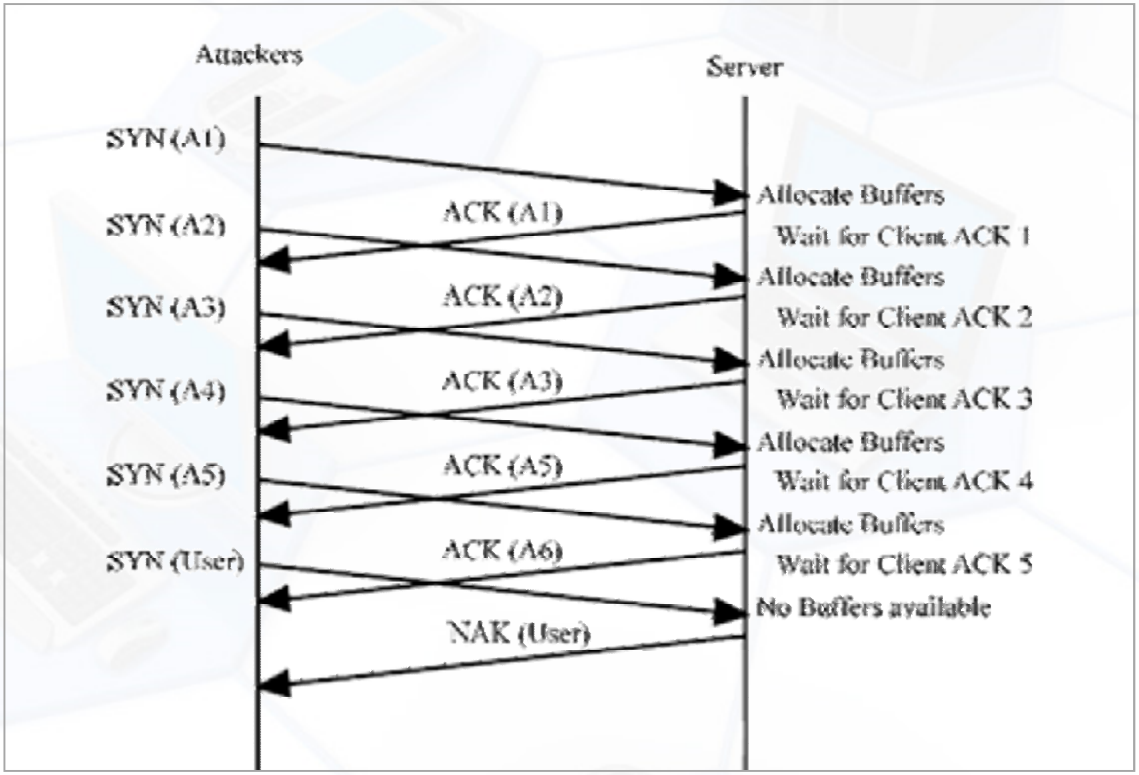
3. 无连接的通信

无连接套接口应用程序的工作过程如图：



六、 传输层安全

1. SYN洪泛攻击



2. DoS攻击和DDoS攻击

- 死亡之ping (ping of death)

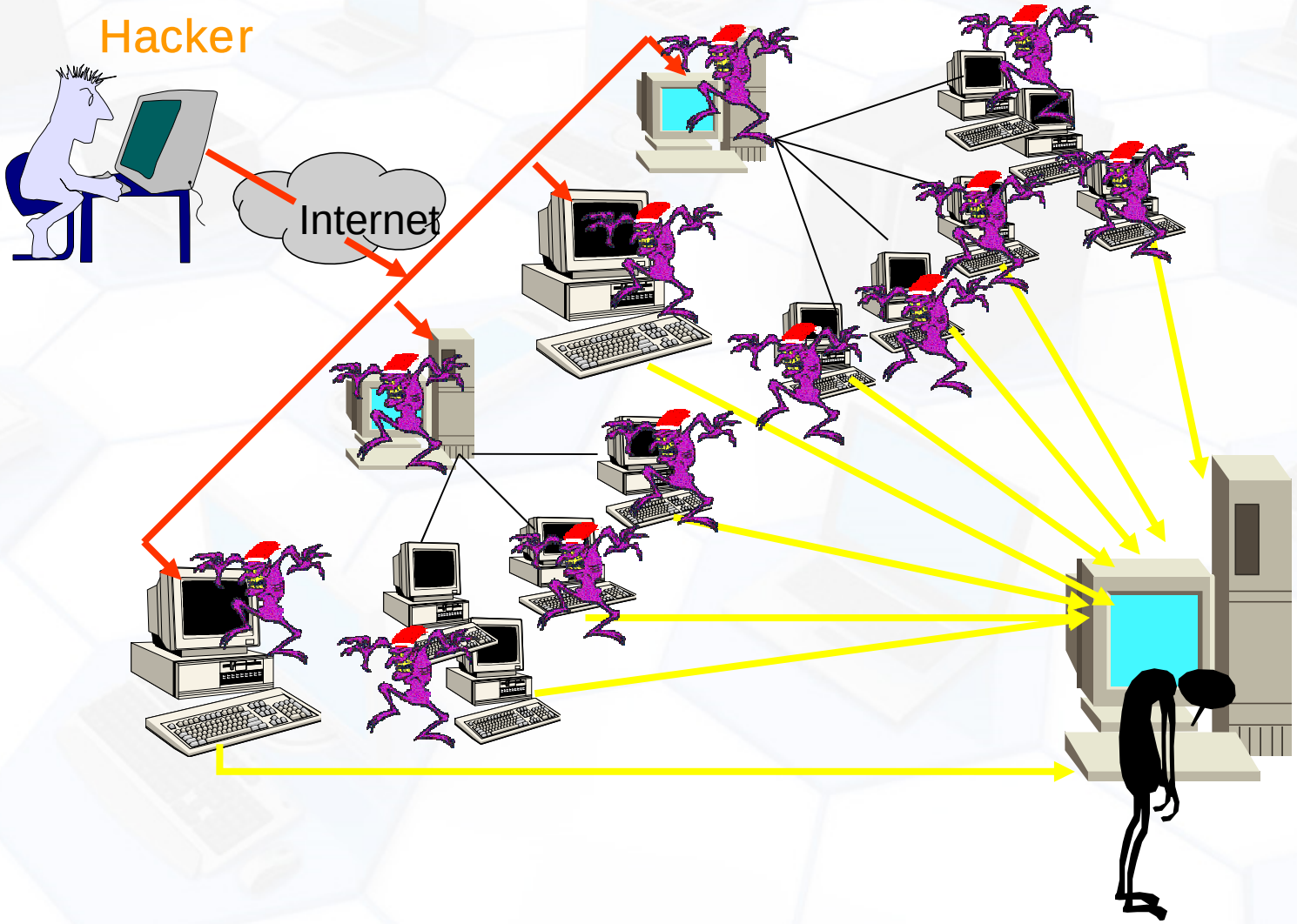
在TCP/IP中对包的最大尺寸都有严格限制规定，ICMP包大小为64KB，且在对包的标题头进行读取之后，要根据该标题头里包含的信息来为有效载荷生成缓冲区。"Ping of Death"就是故意产生畸形的测试包，声称自己的尺寸超过ICMP上限，使未采取保护措施的网络系统出现内存分配错误，导致TCP/IP协议栈崩溃，最终接收方宕机

- 泪滴攻击

利用在TCP/IP协议栈实现中，信任IP碎片中包的标题头所含的信息来实现自己的攻击。IP分段含有指示该分段所包含的是原包的哪一段的信息，某些TCP/IP协议栈（例如NT在servicepack4以前）在收到含有重叠偏移的伪造分段时将崩溃

u Land攻击（Land Attack）

在Land攻击中，黑客利用一个特别打造的SYN包--它的原地址和目标地址都被设置成某一个服务地址，并以此进行攻击。此举将导致接收服务器向它自己的地址发送SYN-ACK消息，结果这个地址又发回ACK消息并创建一个空连接，每一个这样的连接都将保留直到超时，在Land攻击下，许多UNIX将崩溃，NT变得极其缓慢（大约持续五分钟）



Distributed Denial of Service示意