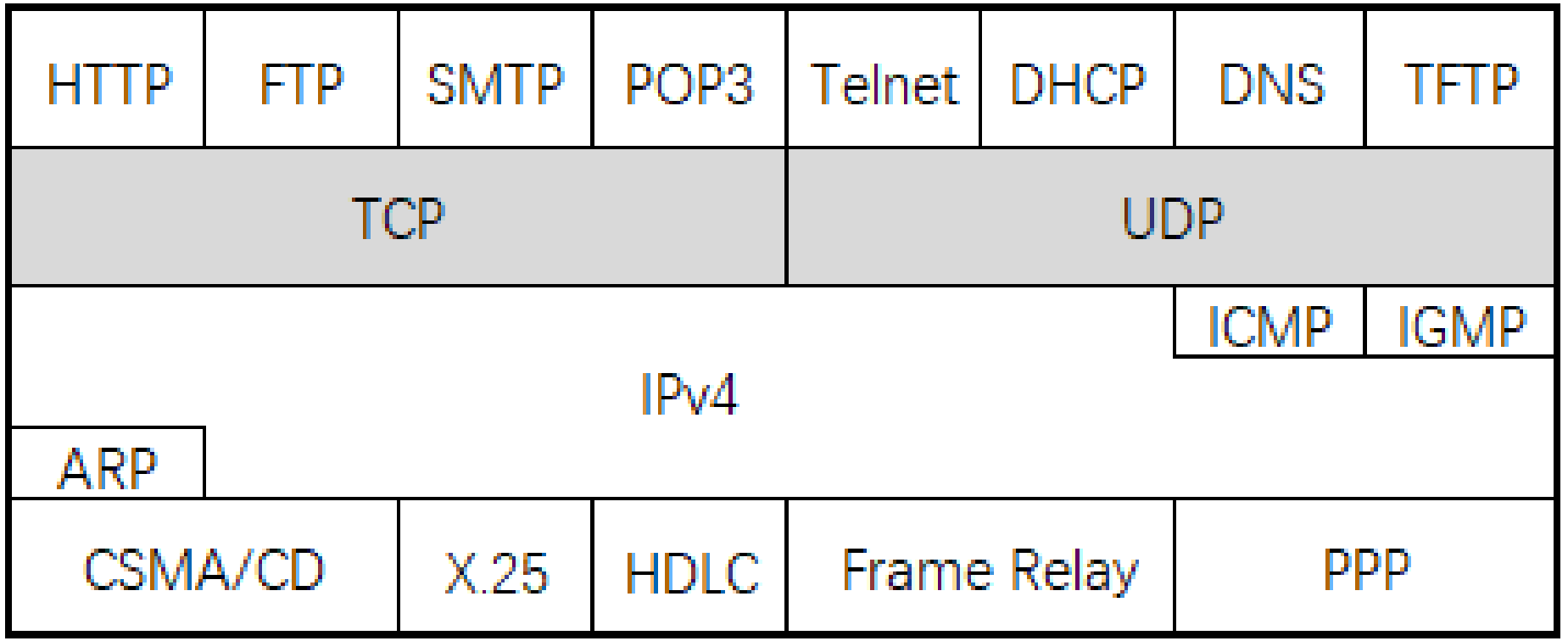
第08章 传输层

讲师：韩立刚
51CTO学院金牌讲师
51CTO学院教学顾问
微软最有价值专家MVP
河北师大软件学院讲师
河北地质大学客座教授
计算机图书作者

高校计算机网络

本章重点讲传输层两个

TCP/IPv4协议组



本章内容

- 8.1传输层的两个协议
- 8.2用户数据报协议UDP
- 8.3传输控制协议TCP
- 8.4可靠传输
- 8.5流量控制
- 8.6拥塞控制
- 8.7 TCP连接管理

8.1传输层的两个协议

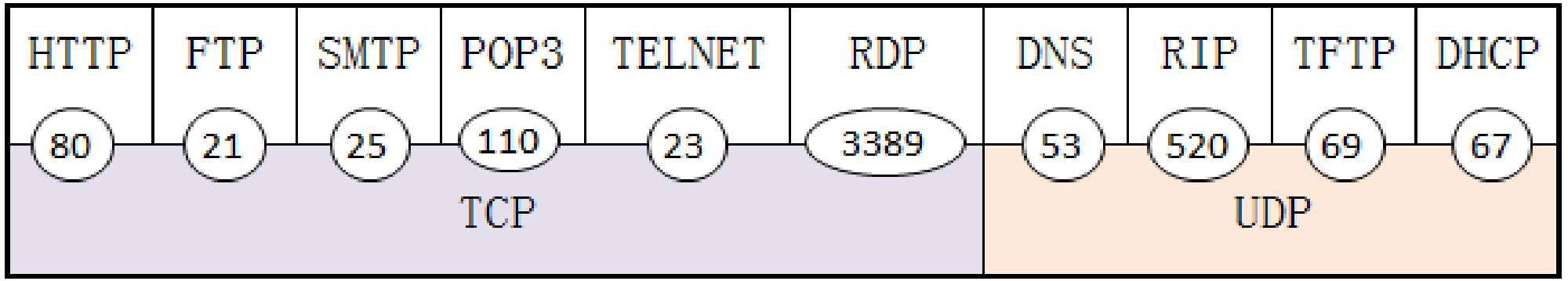
- 8.1.1 TCP和UDP协议的应用场景
- 8.1.2 传输层协议和应用层协议之间的关系
- 8.1.3 服务和端口之间的关系
- 8.1.4 实战：服务器端口冲突造成服务器启动失败
- 8.1.5 实战：更改服务使用的默认端口
- 8.1.6 端口和安全的关系
- 8.1.7 实战：Windows防火墙和TCP/IP筛选实现网络安全

8.1.1 TCP和UDP协议的应用场景

- 网络中的计算机通信无外乎有以下两种情况：
  - 1.要发送的内容多，需要将发送的内容分成多个数据包发送。
  - 2.要发送的内容少，一个数据包就能发送全部内容。
- 针对这两种情况，在传输层有两个协议，TCP（Transmission Control Protocol 即传输控制协议）和UDP（User Datagram Protocol即用户数据报协议）。

8.1.2传输层协议和应用层协议之间的关系1

- 应用层协议很多，传输层就两个协议，如何使用传输层两个协议标识应用层协议呢？
- 传输层协议加一个端口号来标识一个应用层协议，展示了传输层协议和应用层协议之间的关系。

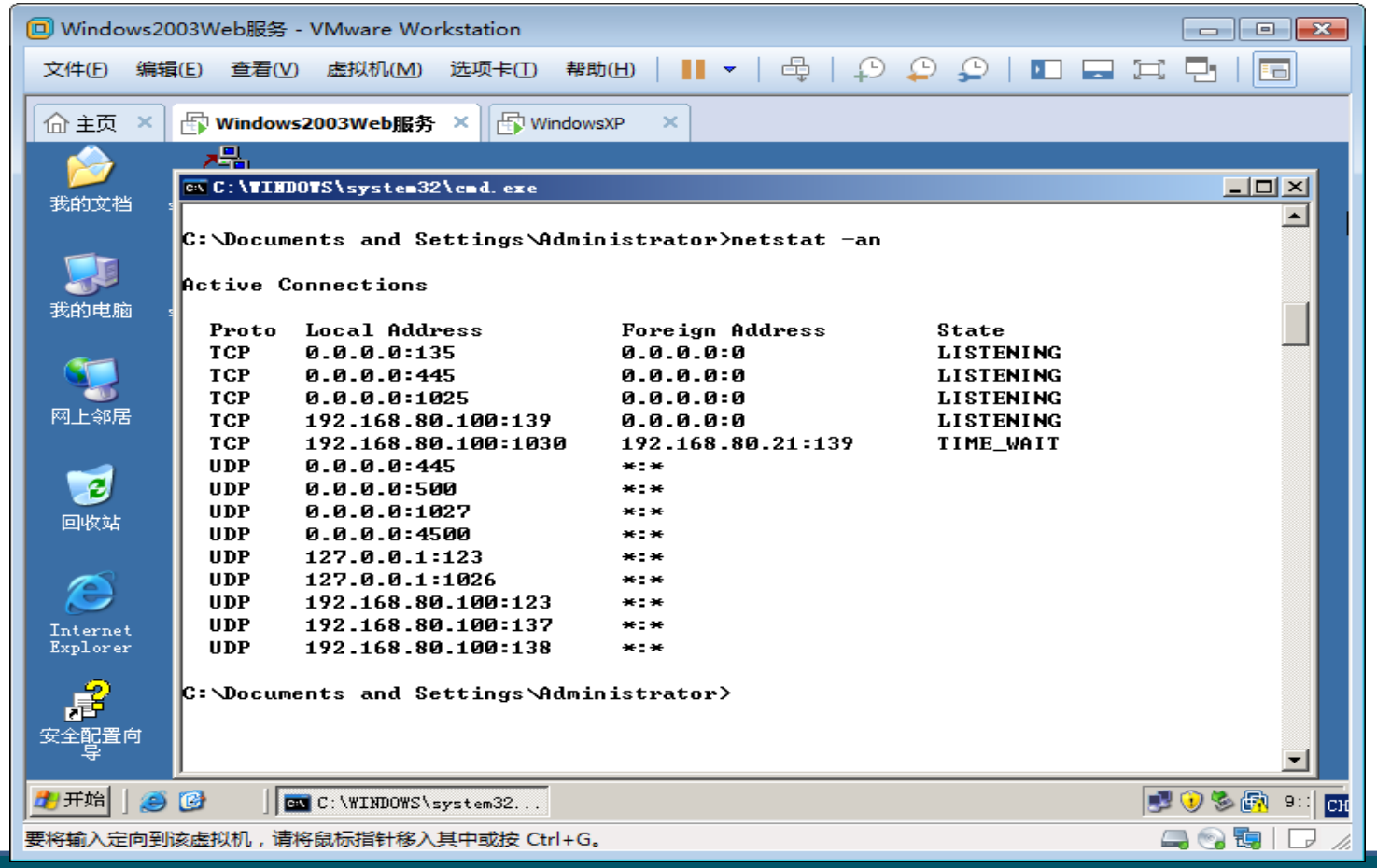


8.1.2传输层协议和应用层协议之间的关系2

- 些常见的应用层协议和传输层协议，以及它们之间的关系。
  - HTTP默认使用TCP的80端口标识。
  - FTP默认使用TCP的21端口标识。
  - SMTP默认使用TCP的25端口标识。
  - POP3默认使用TCP的110端口。
  - HTTPS默认使用TCP的443端口。
  - DNS使用UDP的53端口。
  - 远程桌面协议（RDP）默认使用TCP的3389端口。
  - Telnet使用TCP的23端口。
  - Windows访问共享资源使用TCP的445端口。
  - 微软SQL数据库默认使用TCP的1433端口。
  - mySQL数据库默认使用TCP的3306端口。

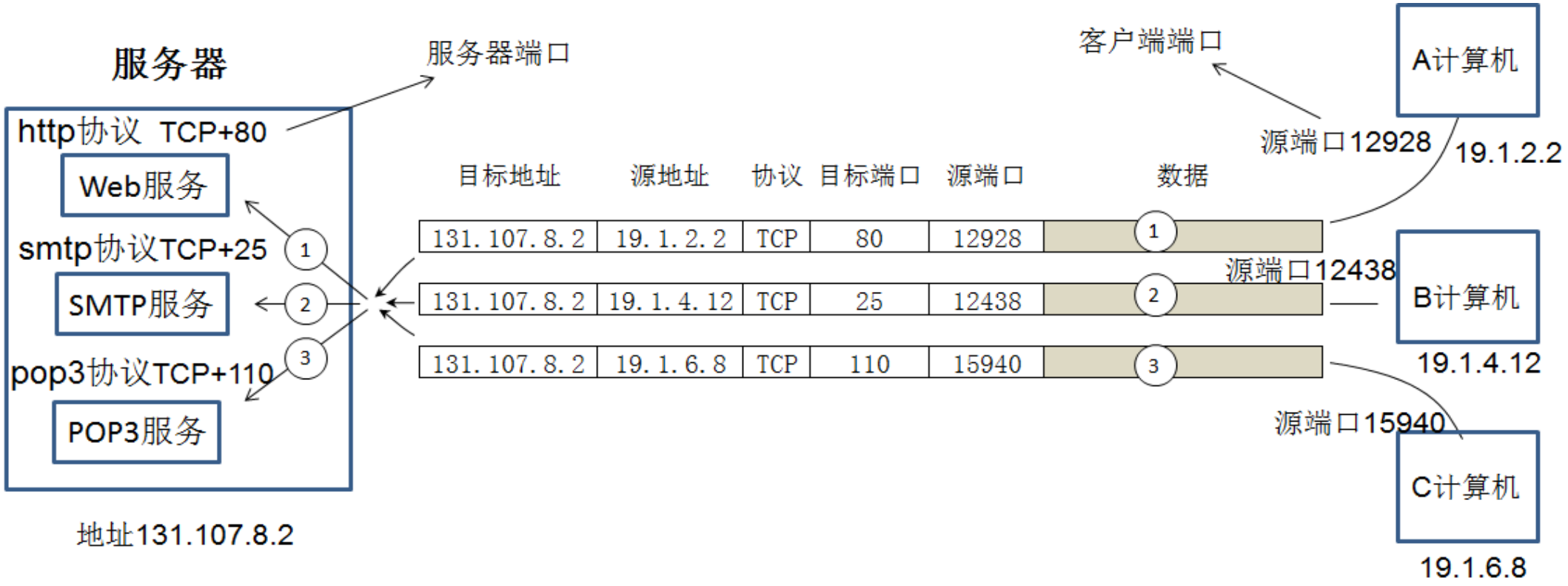
8.1.3服务和端口之间的关系

- Windows和Linux操作系统有些服务为本地计算机提供服务，有些服务为网络中的计算机提供服务。
- 为网络中计算机提供服务的服务，一旦启动就会使用TCP或UDP的某个端口侦听客户端的请求。



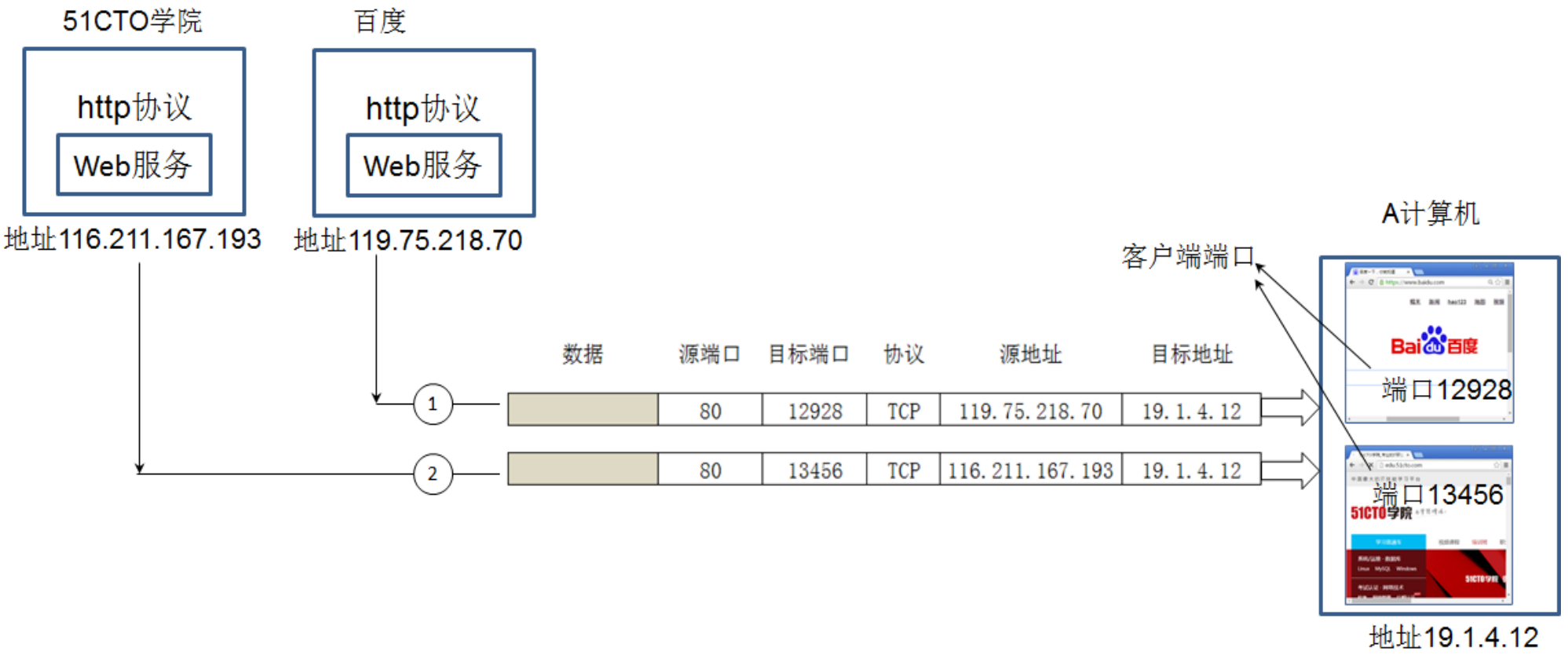
服务和端口的关系

- 客户端通过IP地址定位服务器，通过协议和端口号定位服务器上的服务。



客户端端口的作用

- 客户端软件可以同时访问多个服务器，客户端会为每个出去的流量分配一个唯一的源端口。

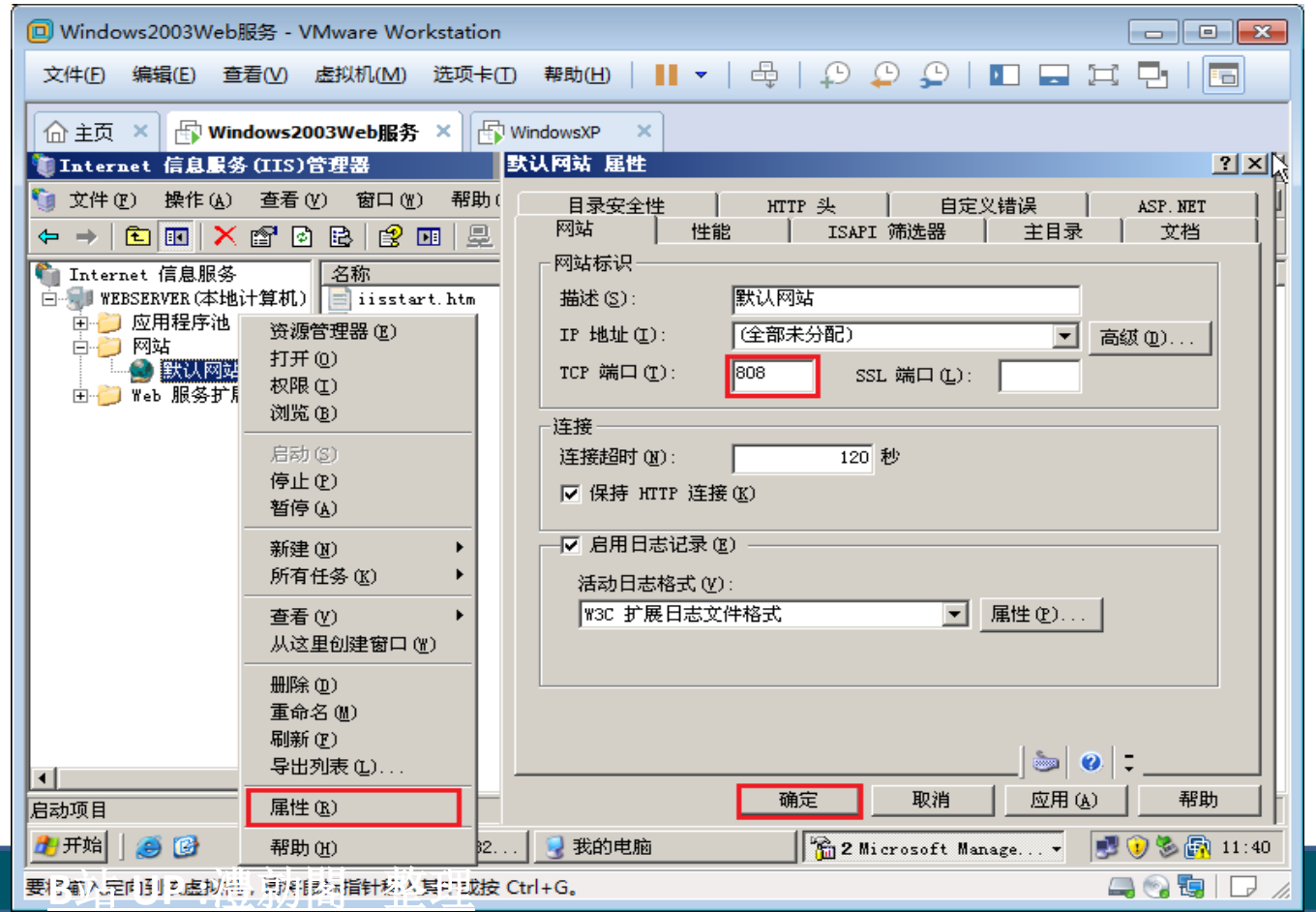


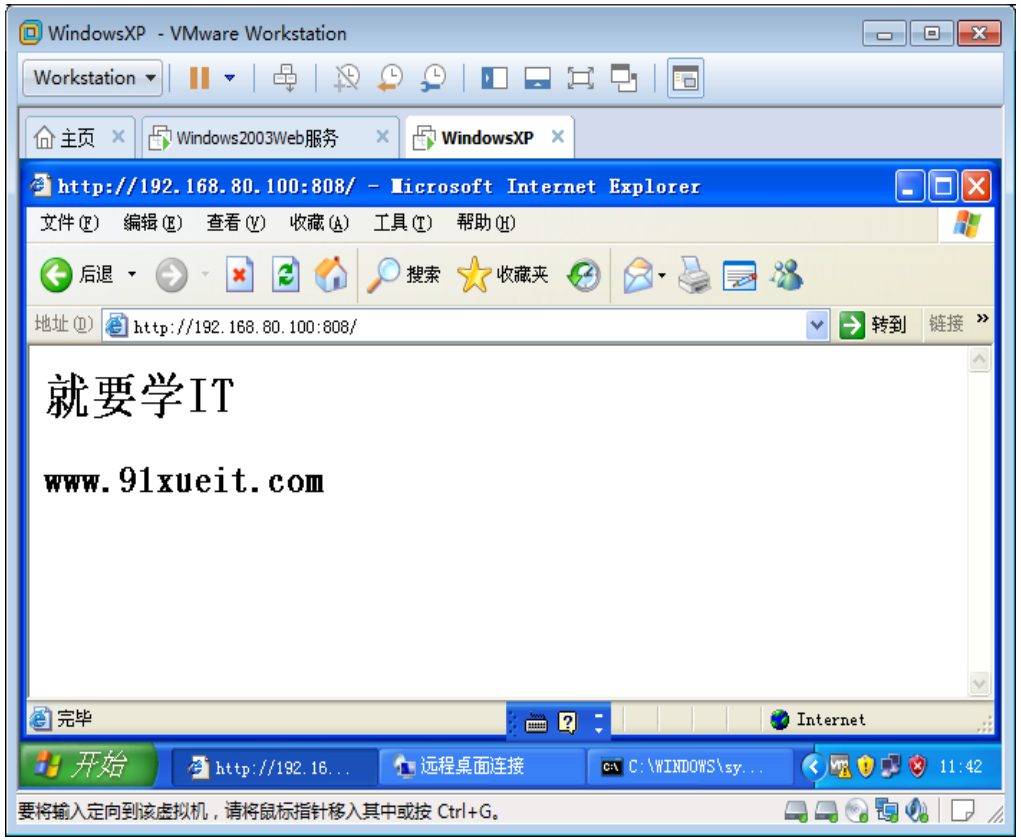
8.1.4实战：服务器端口冲突造成服务启动失败

■服务器上的服务侦听的端口不能冲突，否则将会造成服务启动失败。

8.1.5实战：更改服务使用的默认端口

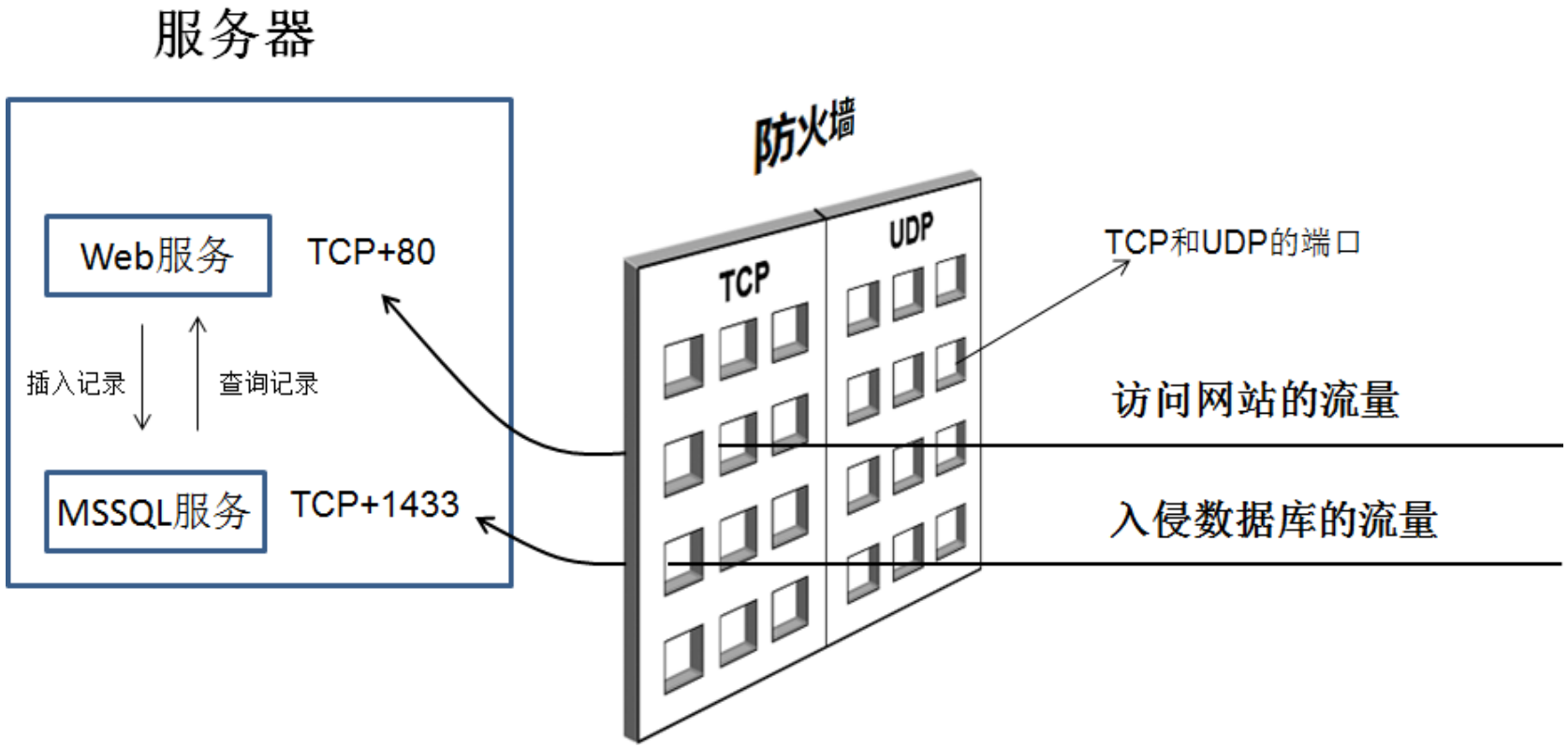
- 应用层协议也可以不使用默认端口和客户端通信。
- 如果不适用默认端口通信，客户度需要指明使用的端口。





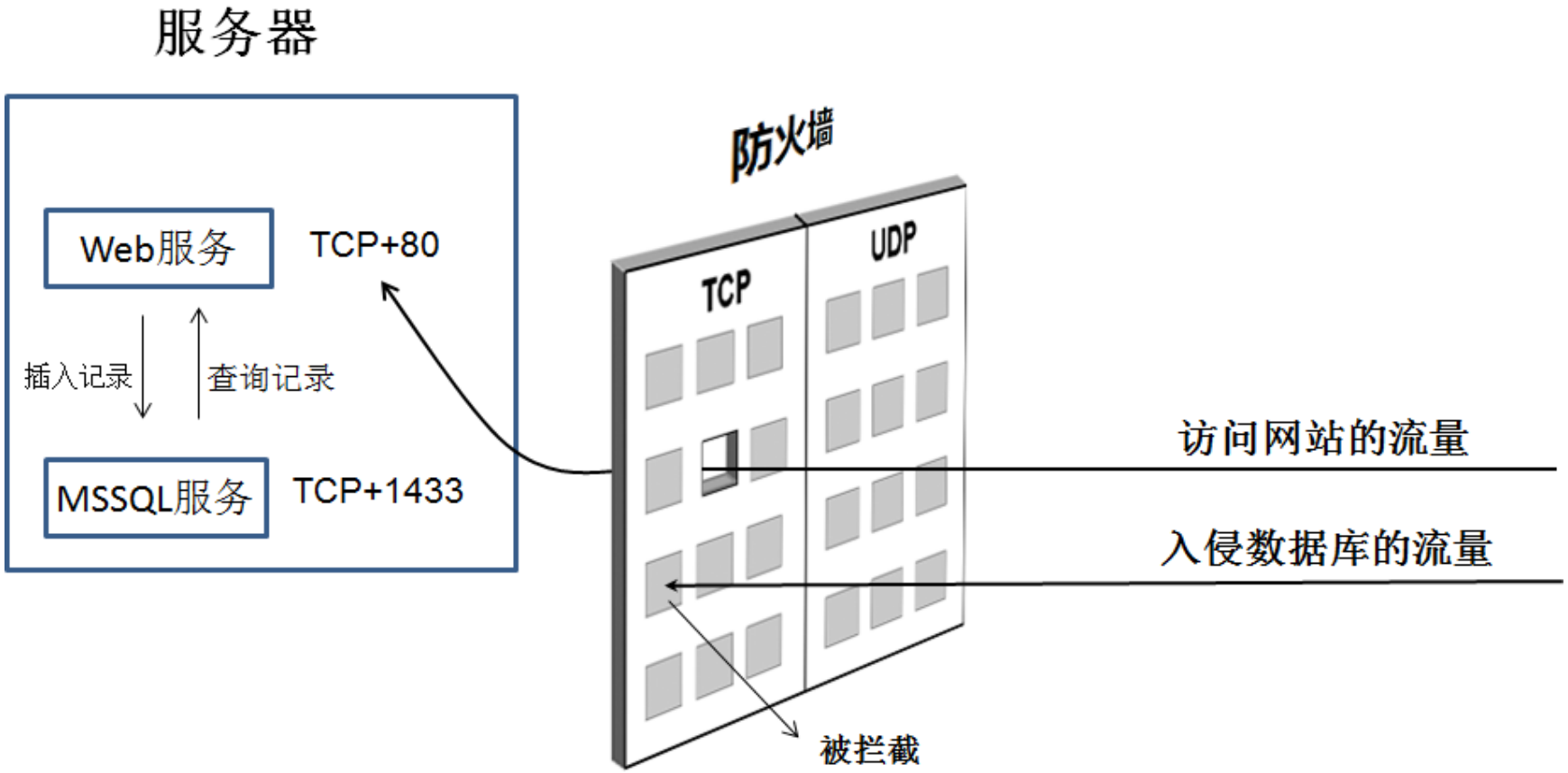
8.1.6端口和网络安全的关系1

■客户端和服务器之间的通信使用应用层协议，应用层协议使用传输层协议+端口标识，如果在网络设备封掉TCP或UDP的某个端口，就不能访问其对应的服务，就可以实现网络安全。



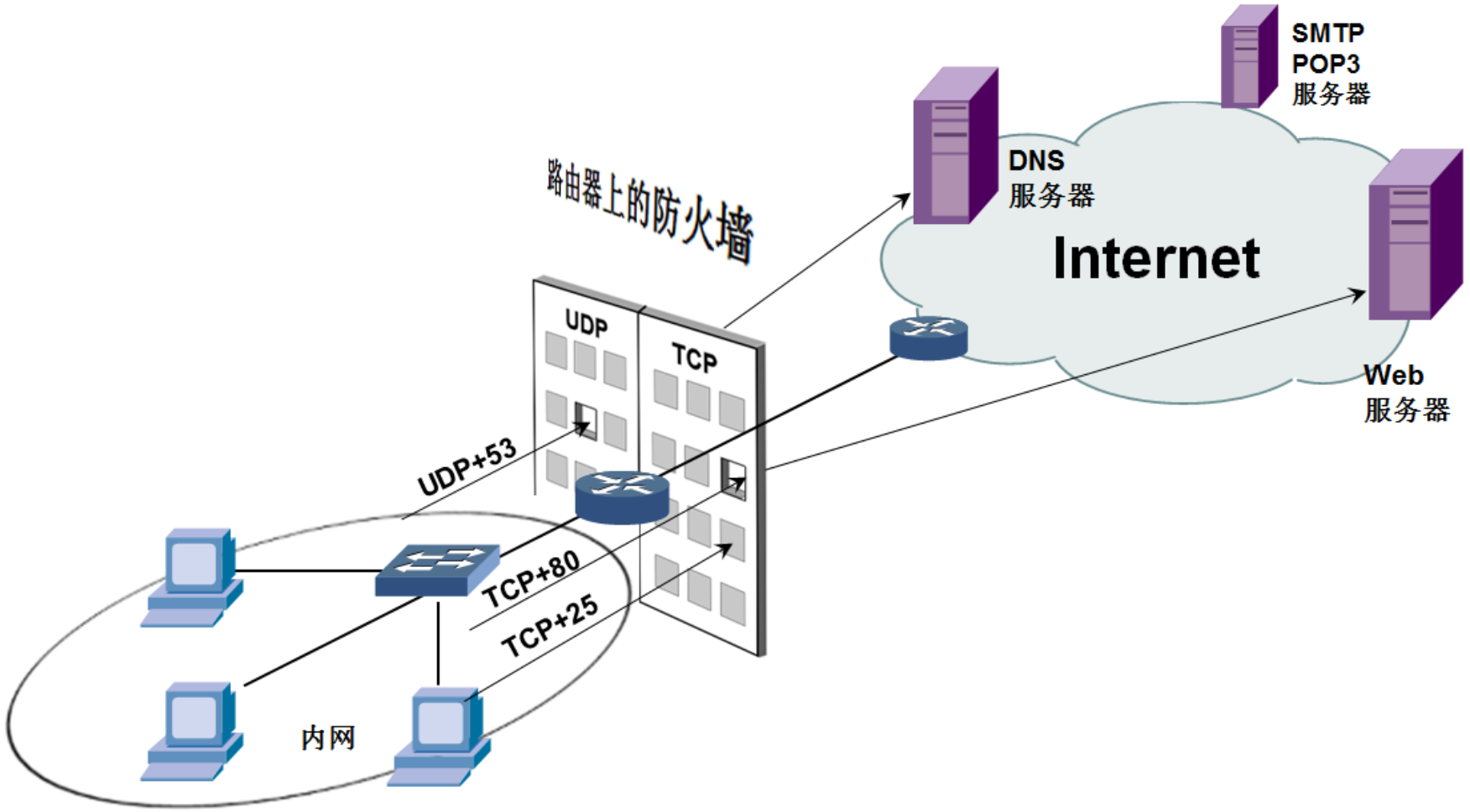
设置服务器网络安全

- 设置服务器网络安全，只开放必要的端口。
- 在Windows上可以通过设置TCP/IP筛选和Windows防火墙来实现。

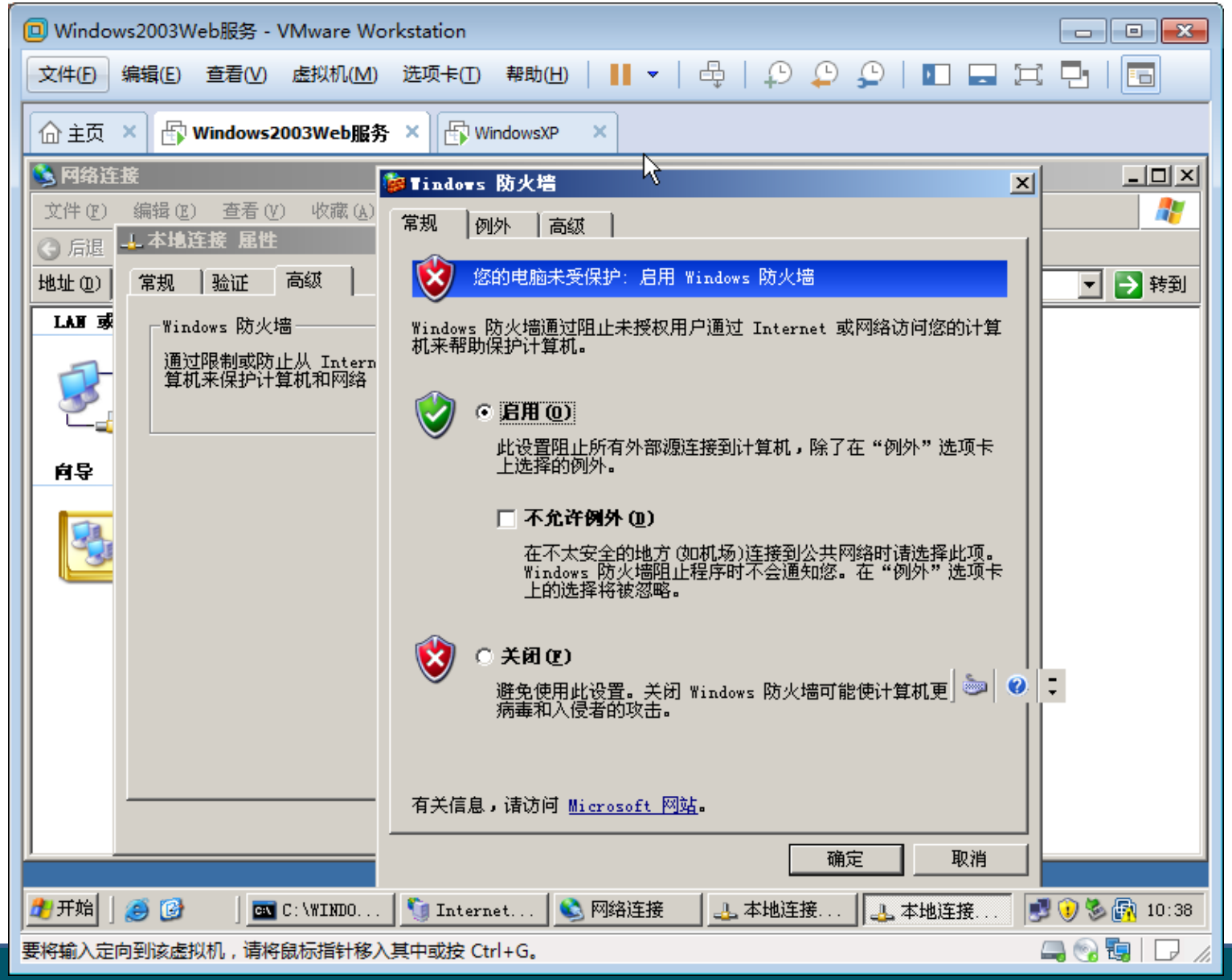


8.1.6端口和网络安全的关系2

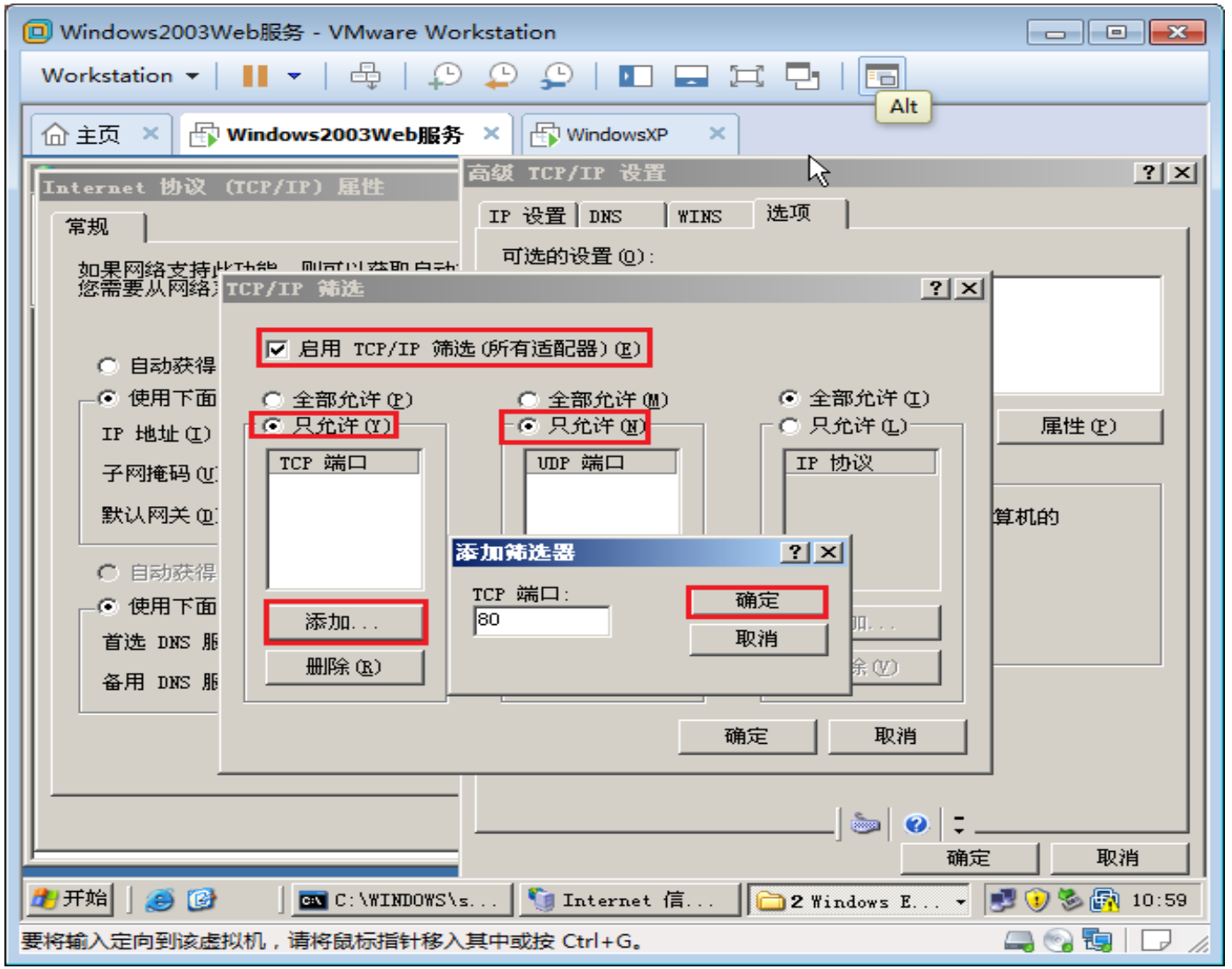
■在网络设备上控制端口



8.1.7实战：Windows防火墙和TCP/IP筛选实现网络安全1



8.1.7实战：Windows防火墙和TCP/IP筛选实现网络安全2

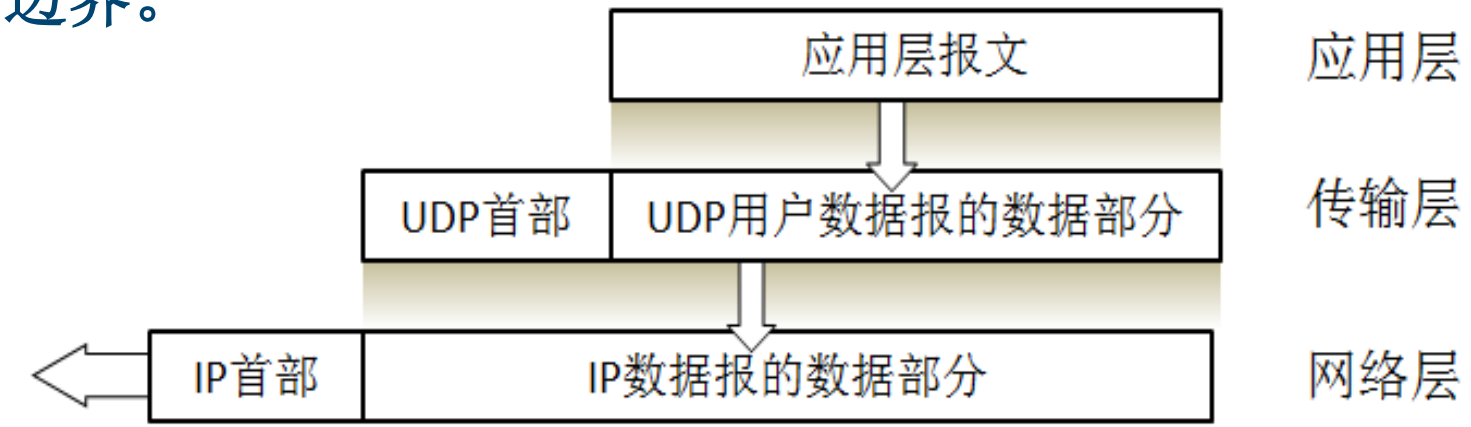


8.2用户数据报协议UDP

- 8.2.1UDP协议的特点
- 8.2.2UDP的首部格式

8.2.1UDP协议的特点1

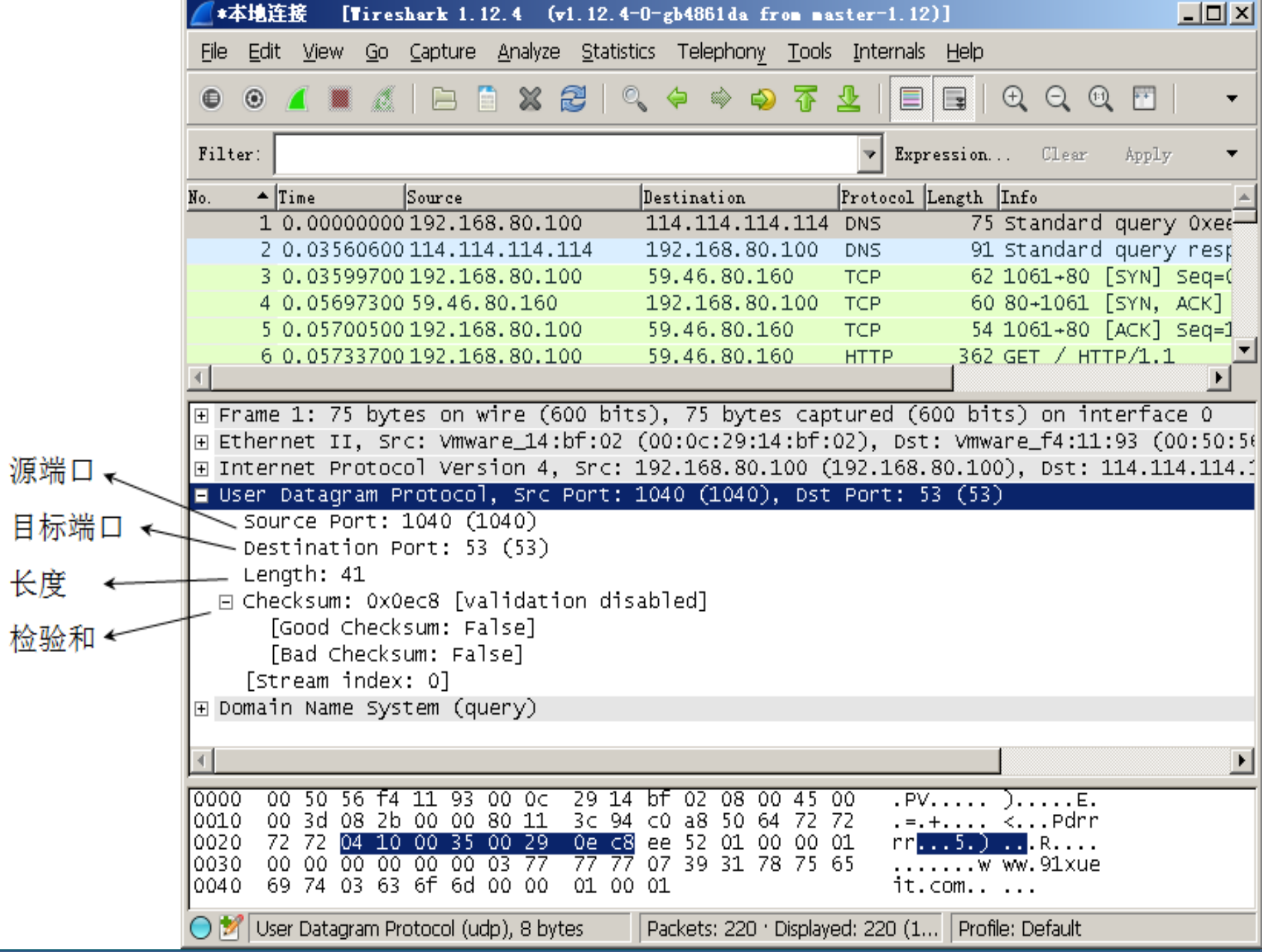
- （1）UDP是无连接的，即发送数据之前不需要建立连接（当然发送数据结束时也没有连接可释放），因此减少了开销和发送数据之前的时延。
- （2）UDP使用尽最大努力交付，即不保证可靠交付，因此主机不需要维持复杂的连接状态表（这里面有许多参数），通信的两端不用保持连接，因此节省系统资源。
- （3）UDP是面向报文的，发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给网络层。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。



8.2.1UDP协议的特点2

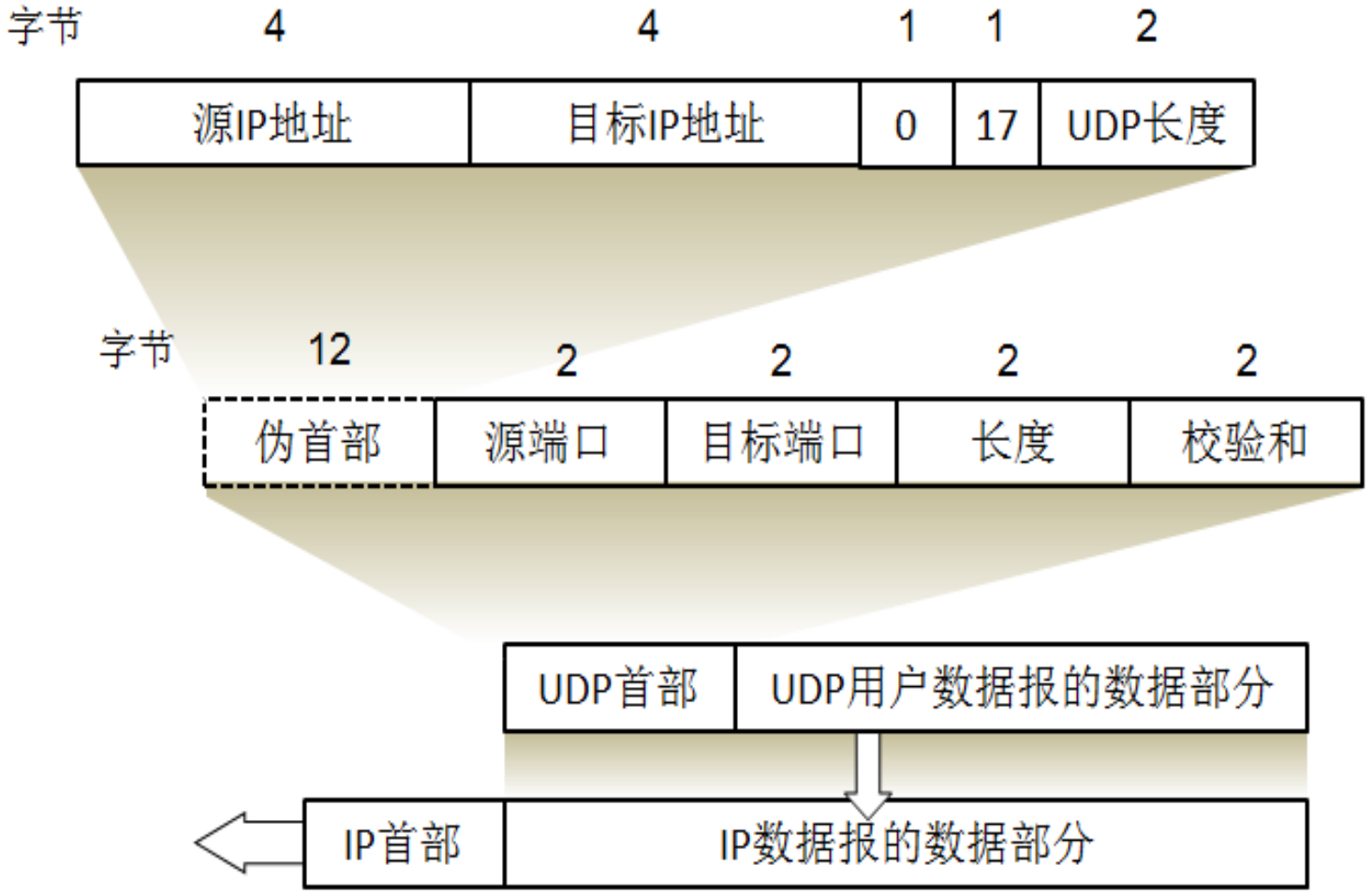
- （4）UDP没有拥塞控制，因此网络出现的拥塞不会使源主机的发送速率降低。这对某些实时应用是很重要的。
- （5）UDP支持一对一、一对多、多对一和多对多的交互通信。
- （6）UDP的首部开销小，只有8个字节，比TCP的20个字节的首部要短。

8.2.2 UDP 的首部格式1

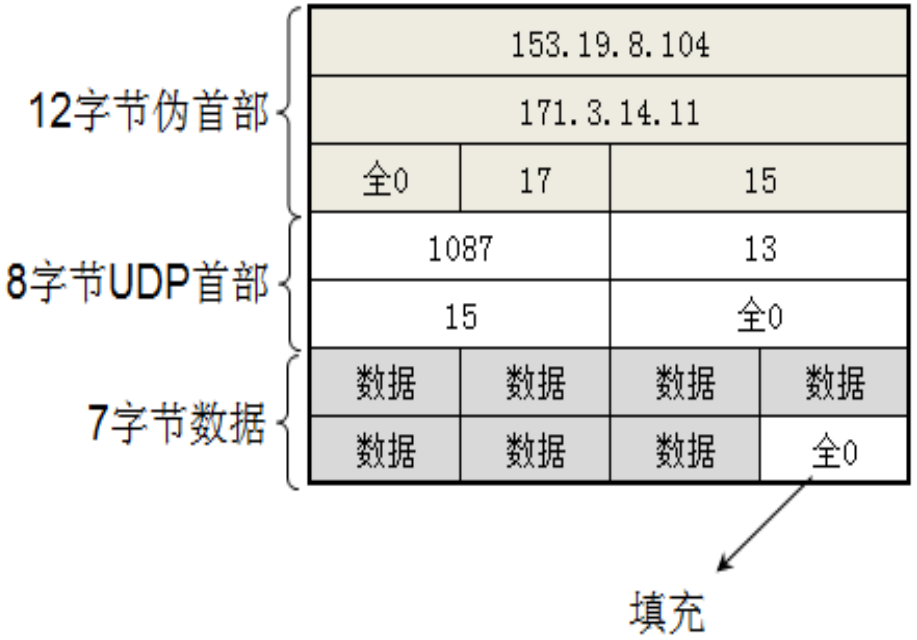


8.2.2UDP的首部格式2

- UDP的首部包括四个字段，源端口、目标端口、长度和校验和，每个字段的长度是两个字节。
- 伪首部包括：源地址、目的地址、UDP数据长度、协议类型（0x11），协议类型就一个字节，但需要补一个字节的0x0，构成12个字节。



计算UDP检验和的例子



10011001 00010011 → 153.19
00001000 01101000 → 8.104
10101011 00000011 → 171.3
00001110 00001011 → 14.11
00000000 00010001 → 0 和 17
00000000 00001111 → 15
00000100 00111111 → 1087
00000000 00001101 → 13
00000000 00001111 → 15
00000000 00000000 → 0（检验和）
01010100 01000101 → 数据
01010011 01010100 → 数据
01001001 01001110 → 数据
01000111 00000000 → 数据和 0（填充）

按二进制反码运算求和 10010110 11101101 → 求和得出的结果

将得出的结果求反码 01101001 00010010 → 检验和

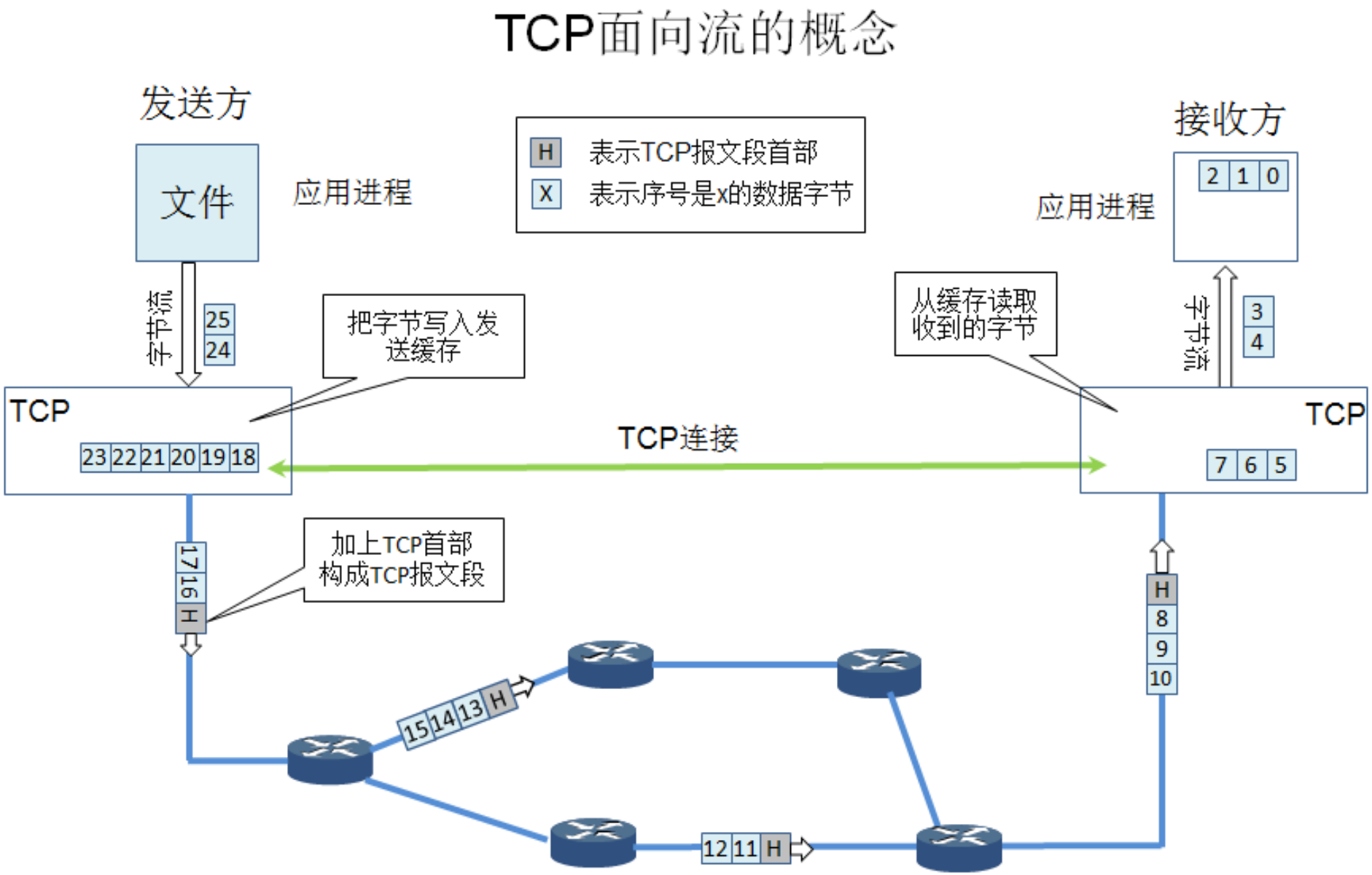
8.3传输控制协议TCP

- 8.3.1TCP协议的主要特点
- 8.3.2TCP报文的首部格式

8.3.1TCP协议的主要特点

- （1）TCP是面向连接的传输层协议。这就是说，应用程序在使用TCP协议之前，必须先建立TCP连接。在传送数据完毕后，必须释放已经建立的TCP连接。这就是说，应用进程之间的通信好像在“打电话”：通话前要先拨号建立连接，通话结束后要挂机释放连接。
- （2）每一条TCP连接只能有两个端点（endpoint），每一条TCP连接只能是点对点的（一对一）。
- （3）TCP提供可靠交付的服务。也就是说，通过TCP连接传送的数据，无差错、不丢失、不重复、且按序发送。
- （4）TCP提供全双工通信。TCP允许通信双方的应用进程在任何时候都能发送数据。TCP连接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据。在发送时，应用程序把数据传送给TCP的缓存后，就可以做自己的事，而TCP在合适的时候把数据发送出去。在接收时，TCP把收到的数据放入缓存，上层的应用进程在合适的时候读取缓存中的数据。
- （5）面向字节流。TCP中的“流”（steam）指的是流入到进程或从进程流出的字节序列。

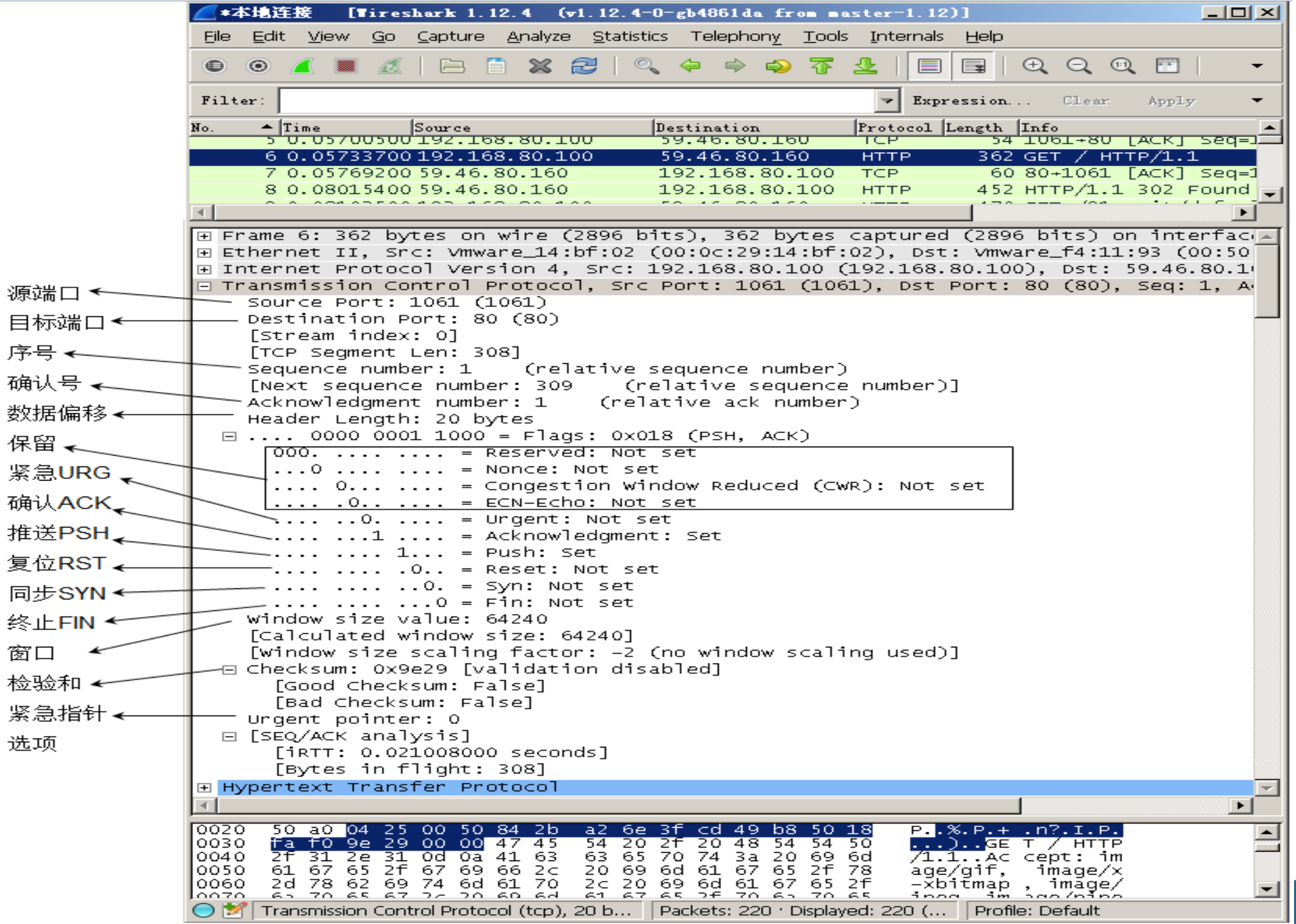
面向字节流


TCP面向流的概念
发送方
接收方
H 表示TCP报文段首部
X 表示序号是x的数据字节
文件
应用进程
应用进程
2 1 0
25
24
把字节写入发送缓存
从缓存读取收到的字节
3
4
TCP
TCP
23 22 21 20 19 18
TCP连接
7 6 5
17 16 H
加上TCP首部构成TCP报文段
H 8 9 10
15 14 13 H
12 11 H

8.3.2TCP报文的首部格式1

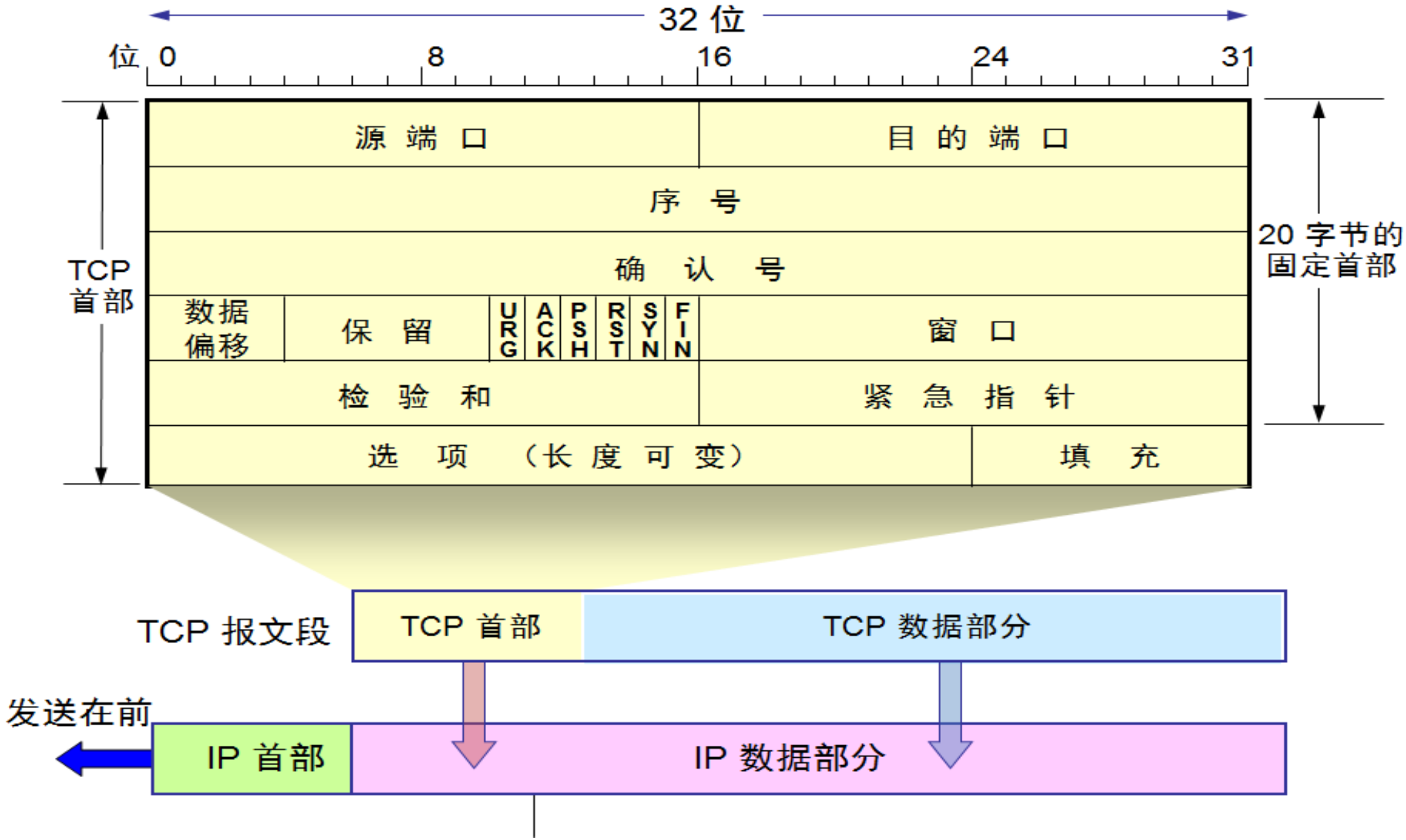
- TCP协议是能够实现数据分段传输、可靠传输、流量控制、网络拥塞避免等功能，因此TCP报文的首部要比UDP报文首部字段要多，并且首部长度不固定。

传输层首部



源端口
目标端口
序号
确认号
数据偏移
保留
紧急URG
确认ACK
推送PSH
复位RST
同步SYN
终止FIN
窗口
检验和
紧急指针
选项

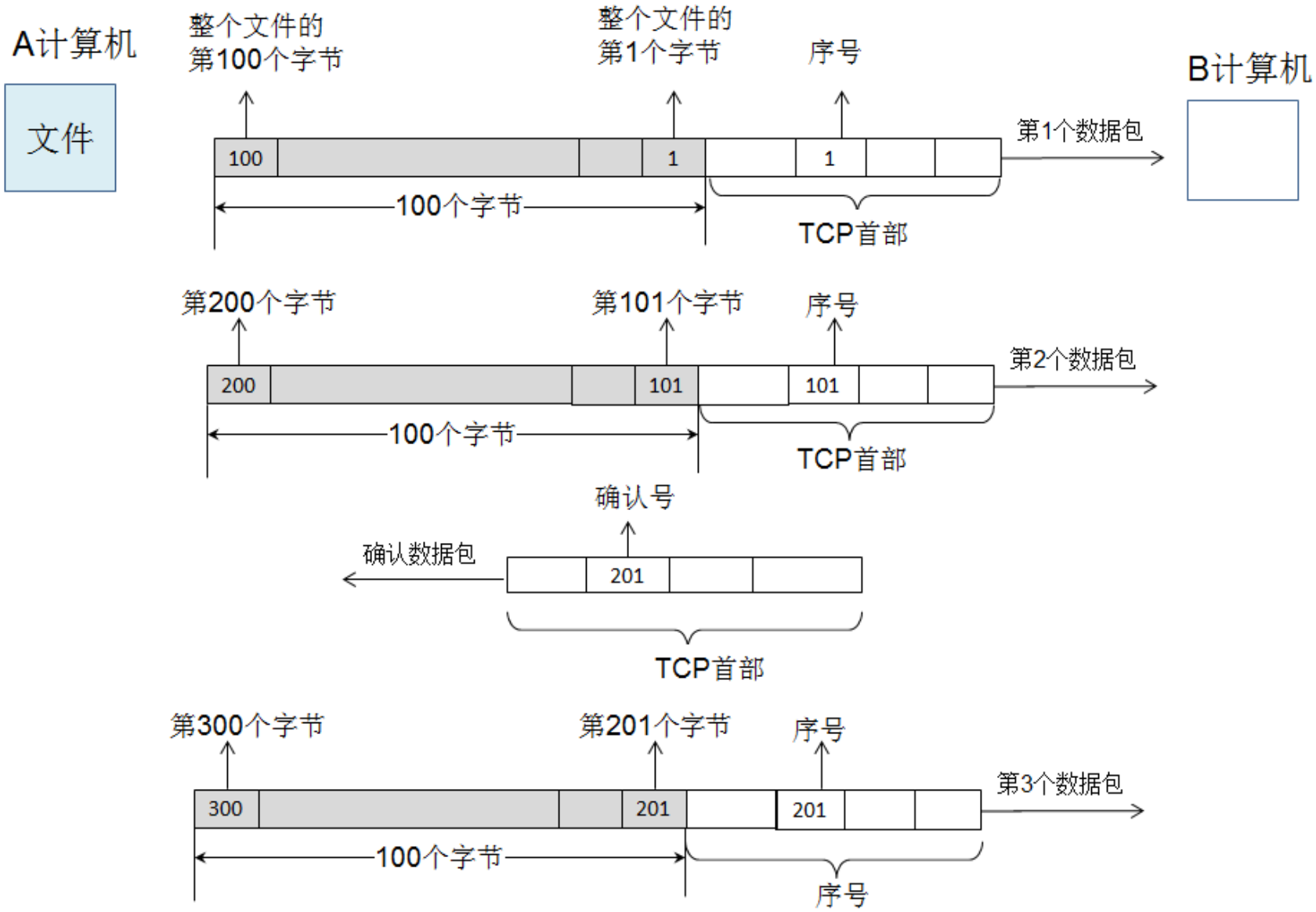
8.3.2TCP报文的首部格式2



8.3.2TCP报文的首部格式3

- （1）源端口和目的端口各占2个字节，分别写入源端口号和目的端口号。和前面图所示的UDP的分用相似，TCP的分用功能也是通过端口实现的。
- （2）序号占4字节。序号范围是[0，2^{32}-1]，共2^{32}（即4 294 967 296）个序号。序号增加到2^{32}-1后，下一个序号就又回到0。TCP是面向字节流的。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

序号 字段的意义


A计算机
文件
整个文件的
第100个字节
整个文件的
第1个字节
序号
B计算机
100
1
1
第1个数据包
100个字节
TCP首部
第200个字节
第101个字节
序号
200
101
101
第2个数据包
100个字节
TCP首部
确认号
确认数据包
201
TCP首部
第300个字节
第201个字节
序号
300
201
201
第3个数据包
100个字节
序号

8.3.2TCP报文的首部格式4

- （3）确认号 占4字节，是期望收到对方下一个报文段的第一个数据字节的序号。
  - TCP协议能够实现可靠传输，接收方收到几个数据包后，就会给发送方一个确认数据包，告诉发送方下一个数据包该发第多少个字节了。
  - 若确认号是N，则表明：到序号N-1为止的所有数据都已正确收到。
- （4）数据偏移 占4位，它指出TCP报文段的数据起始处距离TCP报文段的起始处有多远。这个字段实际上是指出TCP报文段的首部长度。由于首部中还有长度不确定的选项字段，因此数据偏移字段是必要的。但请注意，“数据偏移”的单位为4字节，由于4位二进制数能够表示的最大十进制数字是15，因此数据偏移的最大值是60字节，这也是TCP首部的最大长度，这也就意味着选项长度不能超过40字节。

8.3.2TCP报文的首部格式5

- （6）保留 占6位，保留为今后使用，但目前应置为0。
- （7）紧急URG（URGent） 当URG=l时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快传送（相当于高优先级的数据），而不要按原来的排队顺序来传送。
- 确认ACK（ACKnowlegment） 仅当ACK=1时确认号字段才有效。当ACK=0时，确认号无效。TCP规定，在连接建立后所有传送的报文段都必须把ACK置1。
- （9）推送PSH（PuSH） 当两个应用进程进行交互式的通信时，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。

8.3.2TCP报文的首部格式6

- （10）复位RST（ReSeT） 当RST=l时，表明TCP连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。
- （11）同步SYN（SYNchronization） 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，则应在响应的报文段中使SYN=1和ACK=1。因此，SYN置为1就表示这是一个连接请求或连接接受报文。
- （12）终止FIN（FINish意思是“完”、“终”） 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，并要求释放传输连接。

8.3.2TCP报文的首部格式7

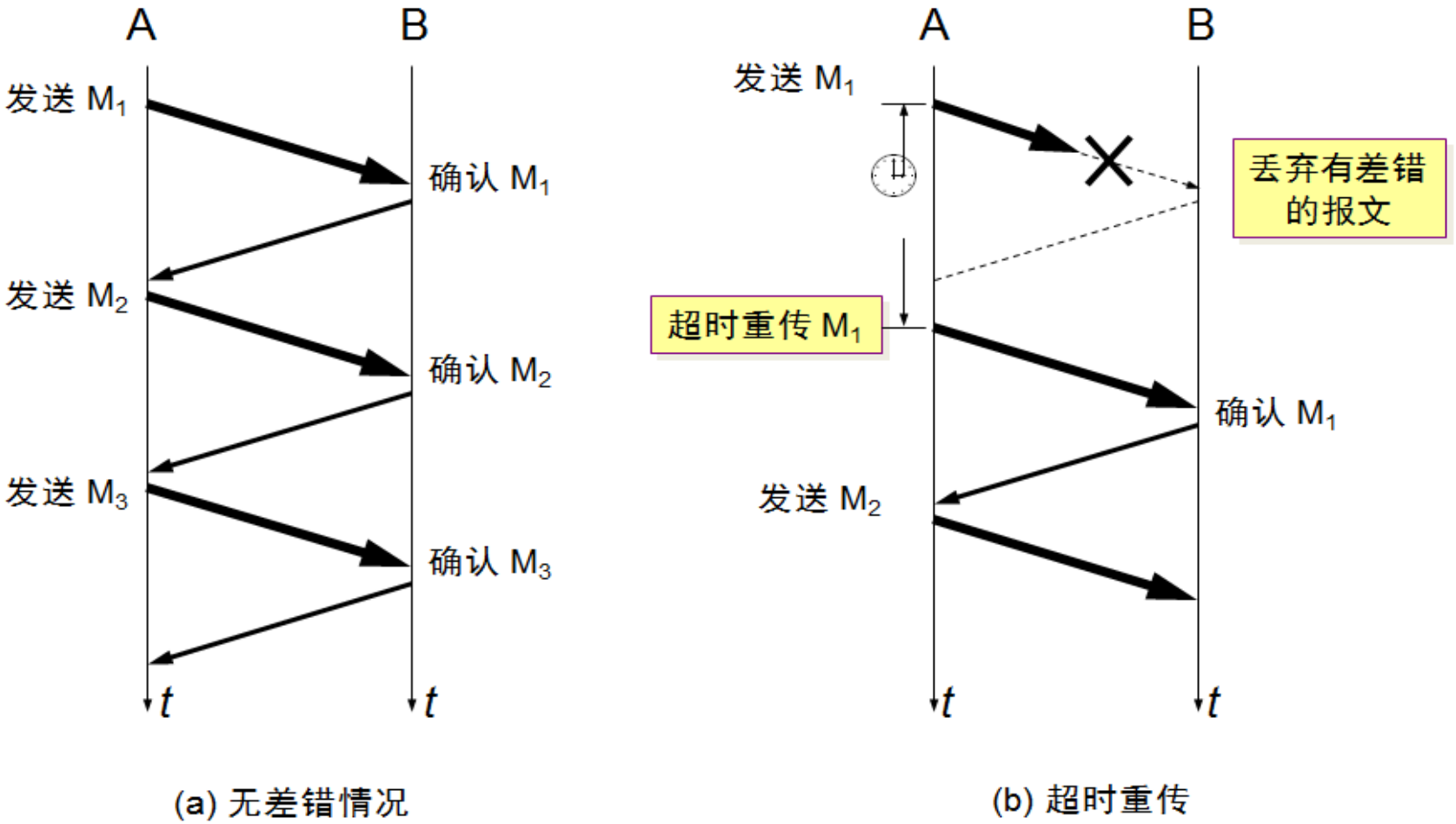
- （13）窗口 占2字节。窗口值是[0，2^{16}-1]之间的整数。TCP协议有流量控制功能，窗口值告诉对方：从本报文段首部中的确认号算起，接收方目前允许对方发送的数据量（单位是字节）。
- （14）检验和 占2字节。检验和字段检验的范围包括首部和数据这两部分。和UDP用户数据报一样，在计算检验和时，要在TCP报文段的前面加上12字节的伪首部。
- （15）紧急指针 占2字节。紧急指针仅在URG=1时才有意义，它指出本报文段中的紧急数据的字节数（紧急数据结束后就是普通数据）。因此紧急指针指出了紧急数据的末尾在报文段中的位置。
- （16）选项 长度可变，最长可达40个字节。当没有使用选项时，TCP的首部长度是20字节。TCP最初只规定了一种选项，即最大报文段长度MSS（Maximum Segment Size）。

8.4可靠传输

- 8.4.1TCP可靠传输的实现-停止等待协议
- 8.4.2连续ARQ协议和滑动窗口协议-改进的停止等待协议
- 8.4.3以字节为单位的滑动窗口技术详解
- 8.4.4改进的确认-选择确认（SACK）
- 8.4.5超时重传的时间调整

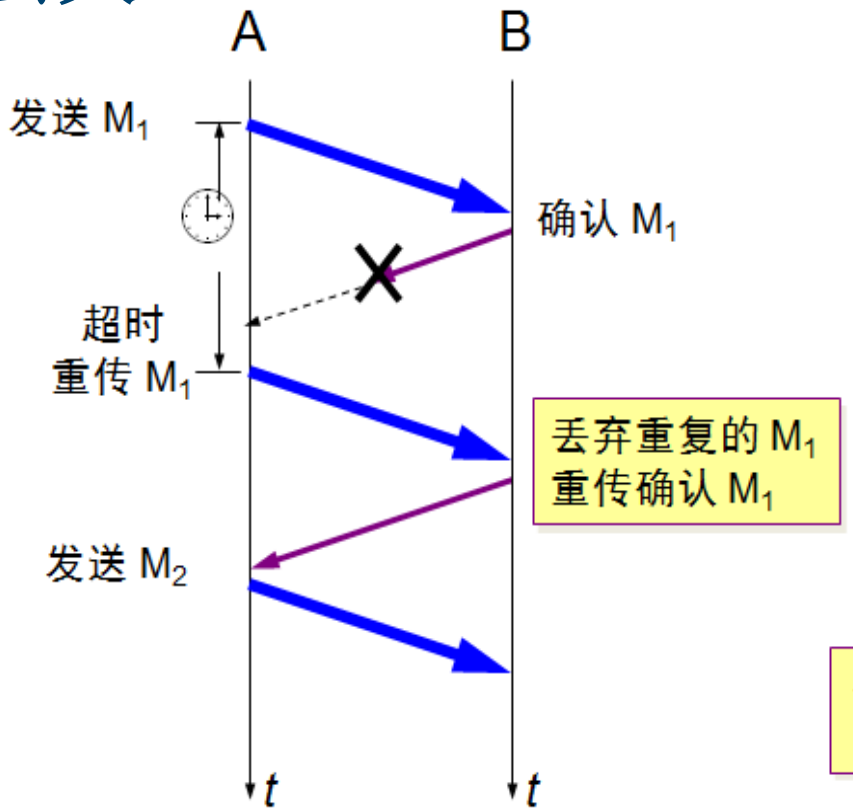
8.4.1TCP可靠传输的实现-停止等待协议1

■无差错情况

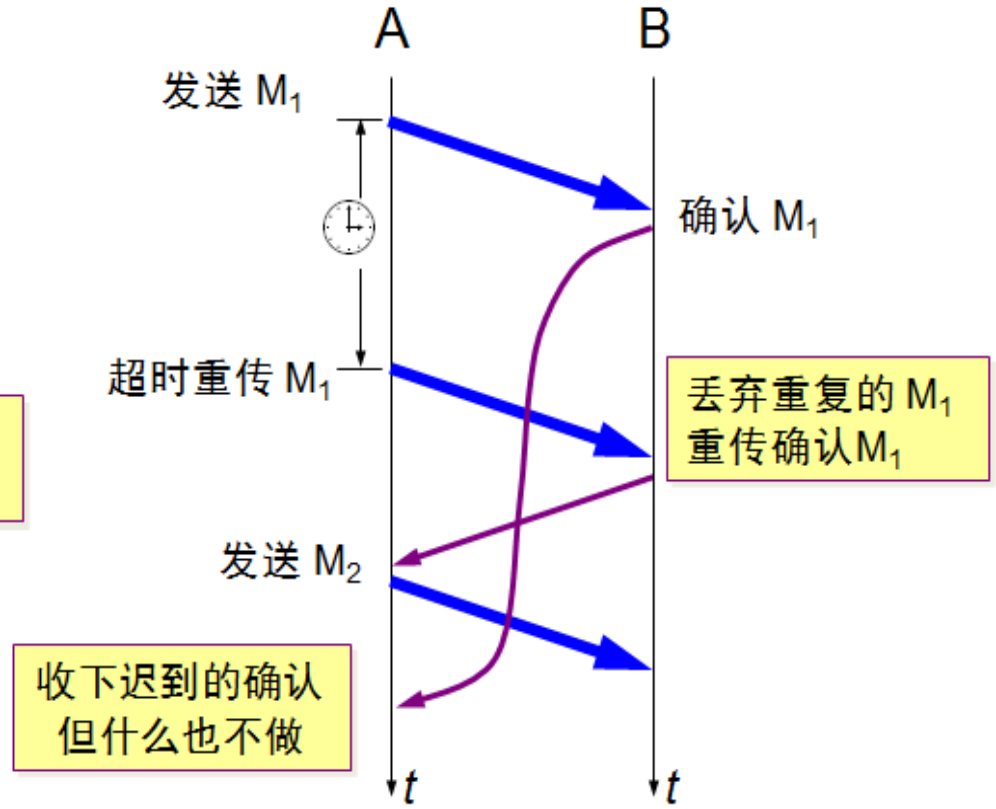


8.4.1TCP可靠传输的实现-停止等待协议2

■出现差错或丢失

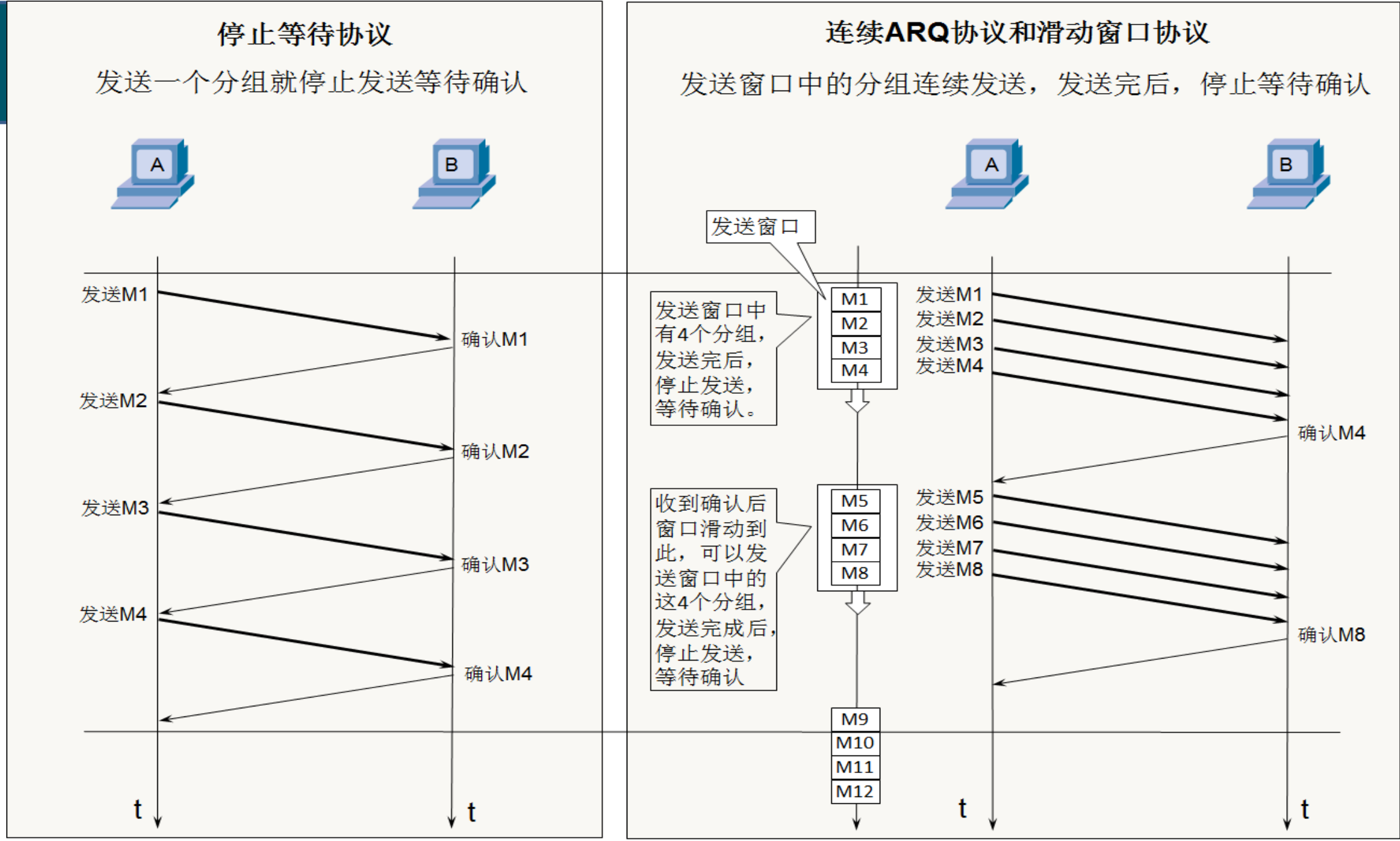


(a) 确认丢失



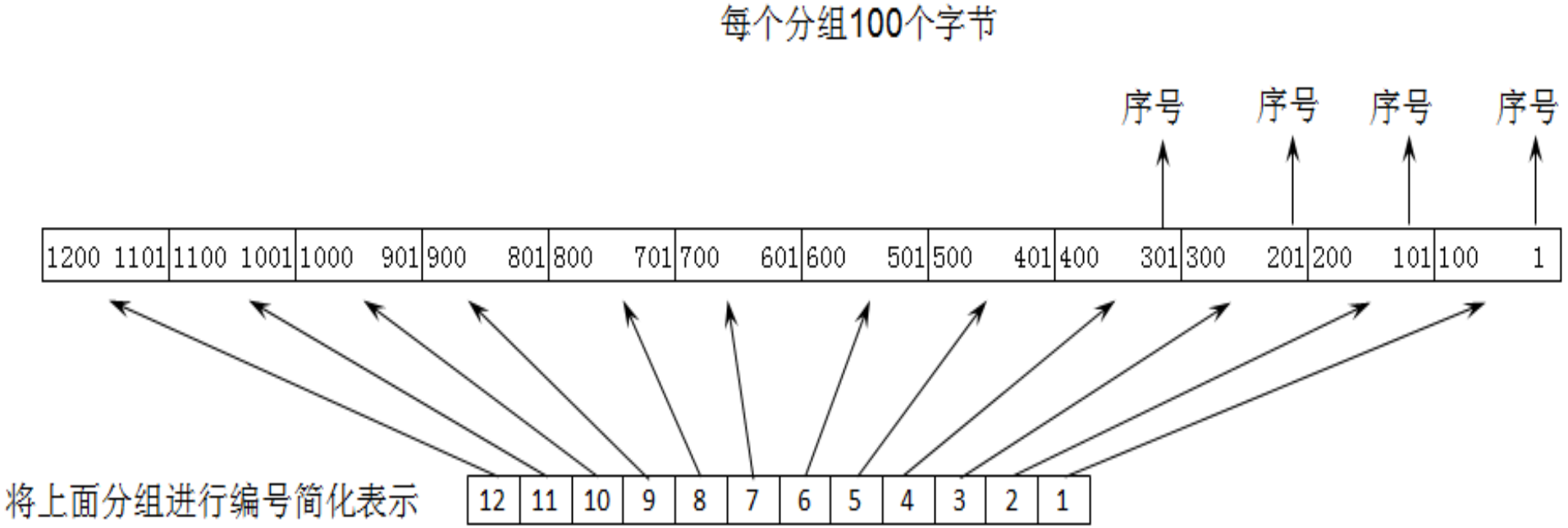
(b) 确认迟到

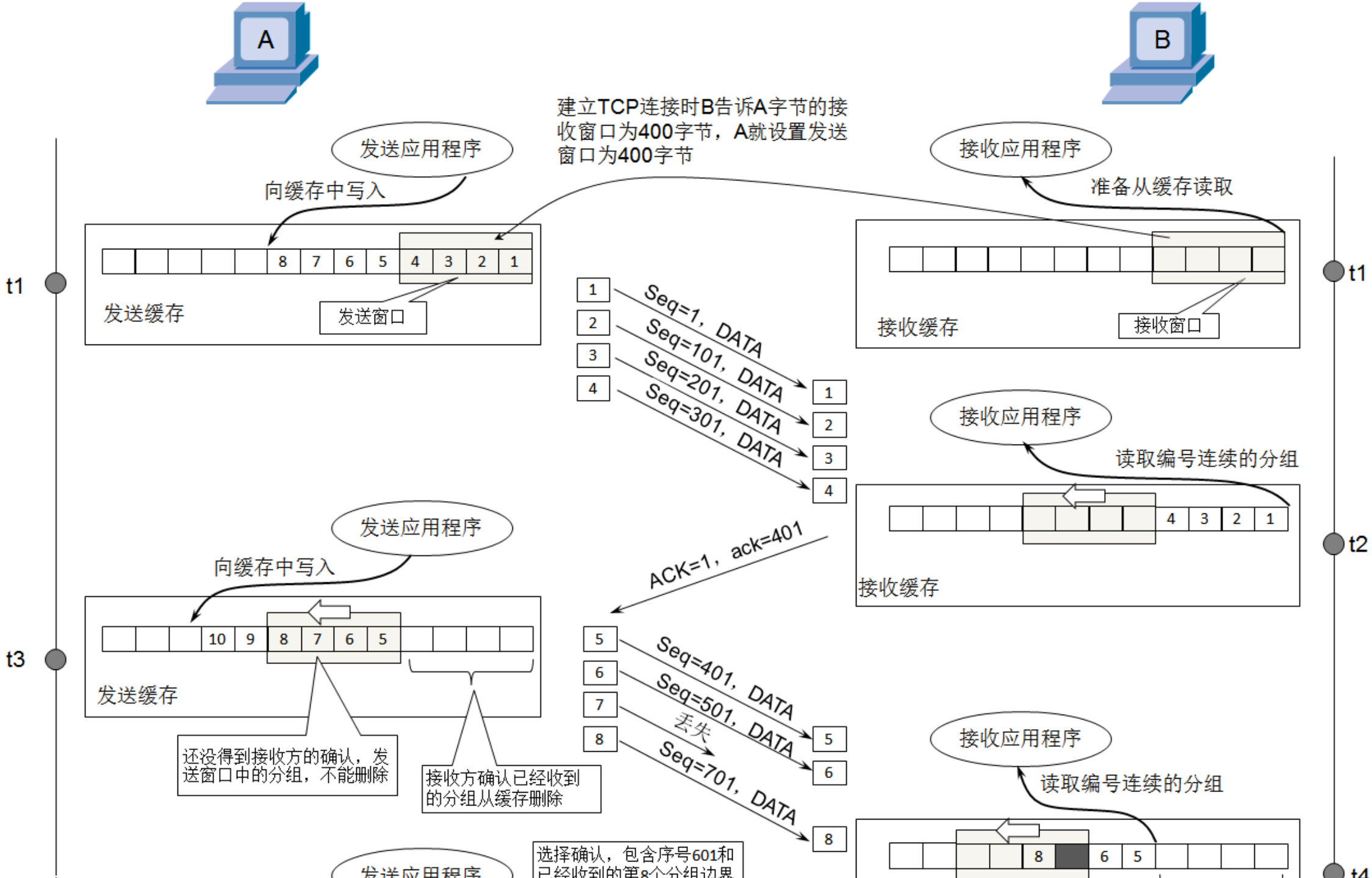
连续ARQ协议和滑动窗口协议-改进的停止等待协议



8.4.3以字节为单位的滑动窗口技术详解

- 滑动窗口是面向字节流的，为了方便大家记住每个分组的序号，下面的讲解每一个分组就假设100个字节，为了方便画图表示，将分组进行编号简化表示，如图所示，不过你要记住，每一

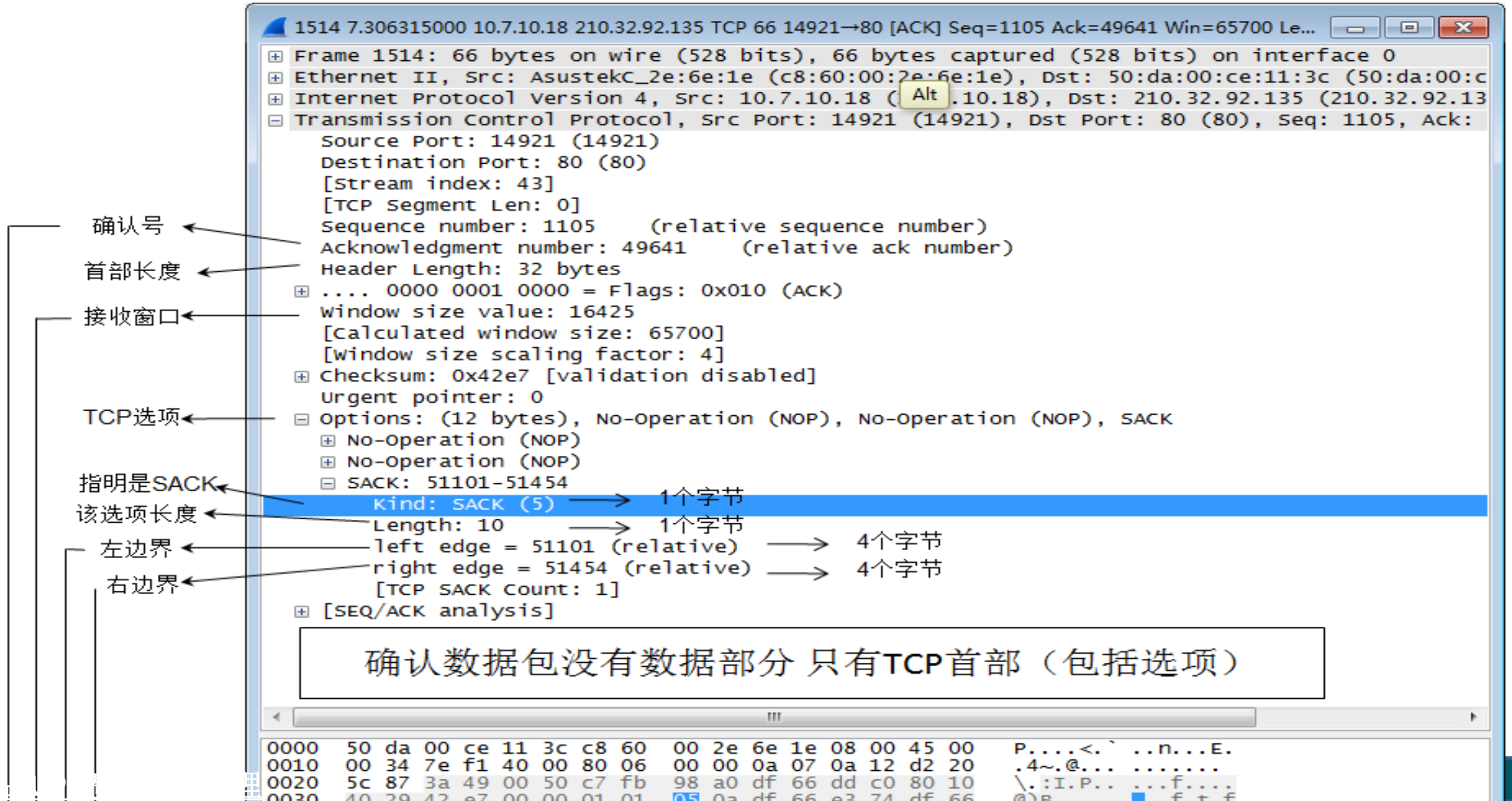



A
B
建立TCP连接时B告诉A字节的接收窗口为400字节，A就设置发送窗口为400字节
发送应用程序
向缓存中写入
接收应用程序
准备从缓存读取
t1
8 7 6 5 4 3 2 1
发送缓存
发送窗口
接收缓存
接收窗口
t1
1
2
3
4
Seq=1，DATA
Seq=101，DATA
Seq=201，DATA
Seq=301，DATA
1
2
3
4
接收应用程序
读取编号连续的分组
4 3 2 1
接收缓存
t2
发送应用程序
向缓存中写入
ACK=1，ack=401
10 9 8 7 6 5
发送缓存
t3
还没得到接收方的确认，发送窗口中的分组，不能删除
接收方确认已经收到的分组从缓存删除
5
6
7
8
Seq=401，DATA
Seq=501，DATA
丢失
Seq=701，DATA
5
6
8
接收应用程序
读取编号连续的分组
8 6 5
选择确认，包含序号601和
t4

8.4.4改进的确认-选择确认（SACK）

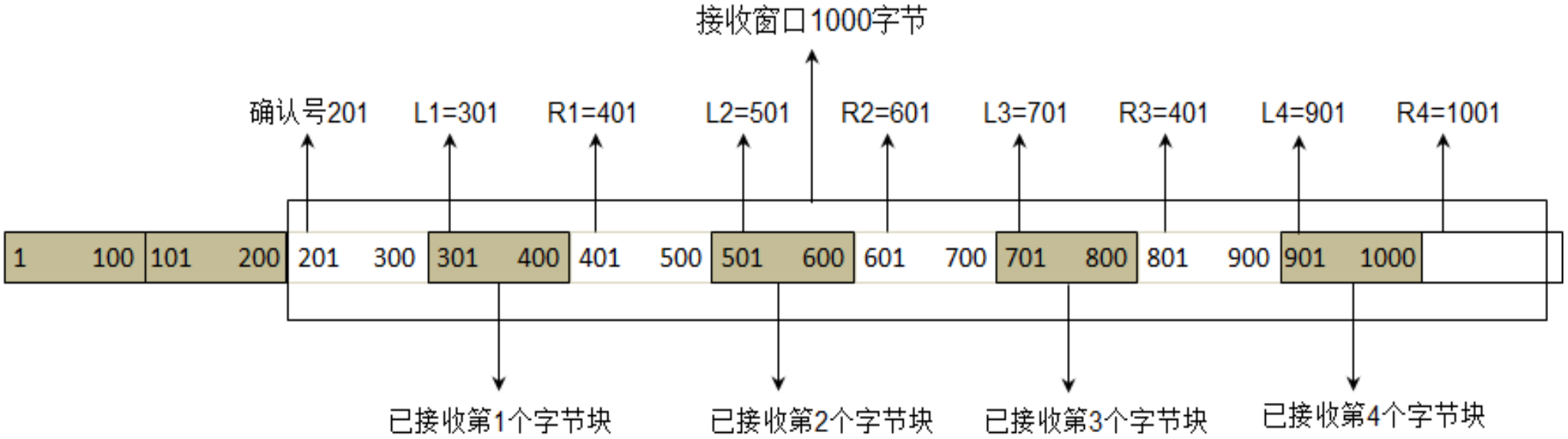
- TCP通信时，如果发送序列中间某个数据包丢失，TCP会通过重传最后确认的分组后续的分组，这样原先已经正确传输的分组也可能重复发送，降低了TCP性能。为改善这种情况，发展出SACK（Selective Acknowledgment，选择确认）技术，使TCP只重新发送丢失的包，不用发送后续所有的分组，而且提供相应机制使接收方能告诉发送方哪些数据丢失，哪些数据已经提前收到等。

8.4.4改进的确认-选择确认（SACK）



选择性确认最多表示4个边界

- 由于TCP首部选项最长40个字节，而指明一个边界需要用掉4个字节（因为序号有32位，需要使用4个字节表示），因此在TCP选项中一次最多只能指明4个字节块的边界信息。这是因为4个字节块有8个边界，一个边界占用4个字节，占用32个字节，另外还需要2个字节，一个字节用来指明是SACK选项，另一个字节指明这个选项占多少字节。



8.4.5超时重传的时间调整

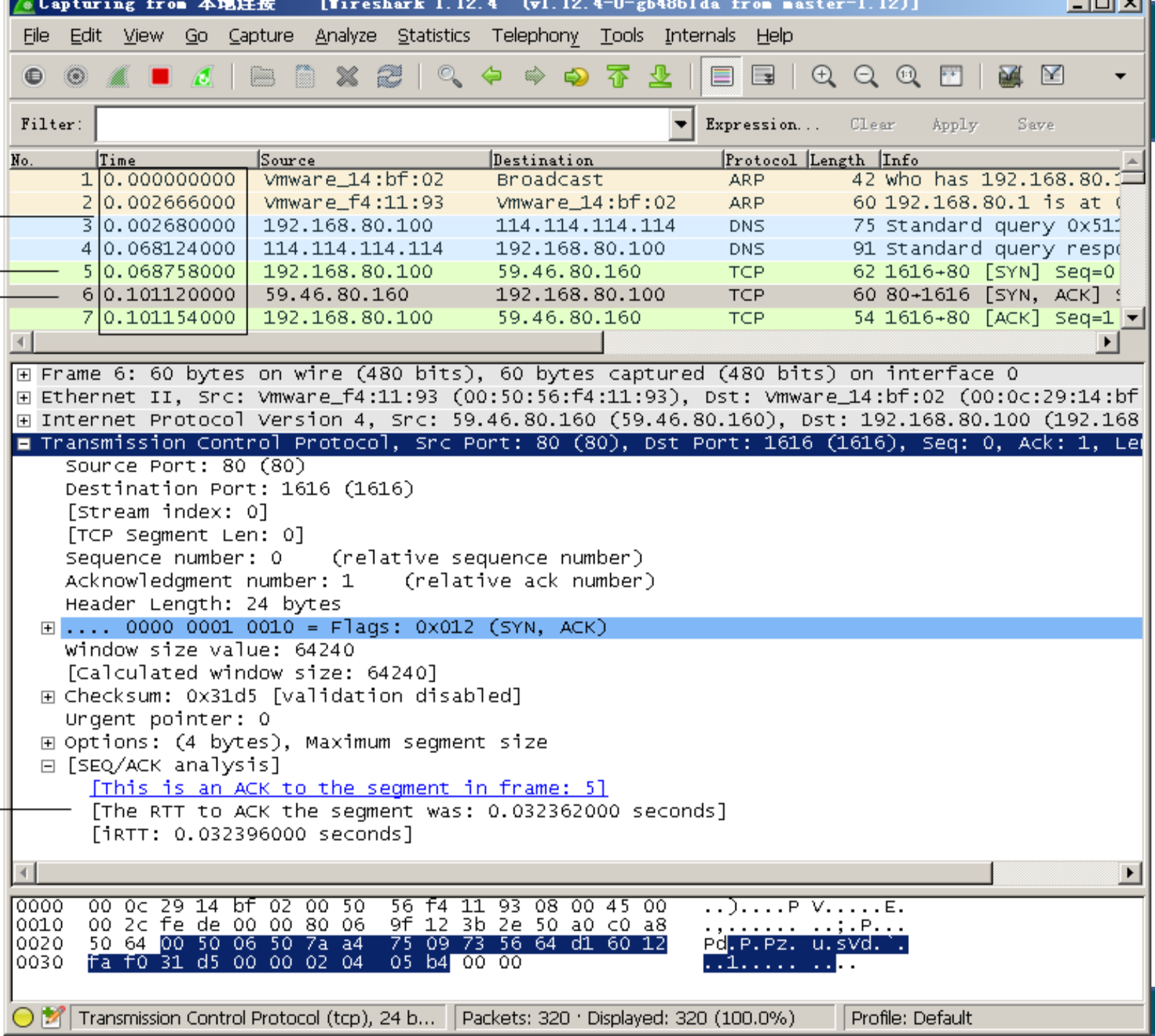
- TCP的发送方在规定的时间内没有收到确认就要重传已发送的报文段。这种重传的概念是很简单的，但重传时间的选择却是TCP最复杂的问题之一。
- 传输层的超时计时器的超时重传时间究竟应设置为多大呢？TCP往返传输时间（RTT） 的测量可以采用两种方法：
- TCP Timestamp选项
  - TCP时间戳选项可以用来精确的测量RTT。发送方在发送报文段时把当前时钟的时间值放入时间戳字段，接收方在确认该报文段时把时间戳字段值复制到时间戳回送回答字段。因此，发送方在收到确认报文后，可以准确地计算出RTT来。RTT=当前时间-数据包中Timestamp选项的回显时间。
- 重传队列中数据包的TCP控制块
  - 在TCP发送窗口中保存着发送而未被确认的数据包，数据包skb中的TCP控制块包含着一个变量， tcp_skb_cb→when，记录了该数据包的第一次发送时间，当收到该数据包确认，就可以计算RTT，RTT=当前时间-when。这就意味着发送端收到一个确认，就能计算新的RTT。

抓包计时

建立TCP连接请求

建立TCP连接响应

第5个包的确认计算出的RTT



RTT的调整

- RTT是随着网络状态动态的变化的，TCP保留了RTT的一个加权平均往返时间RTTs（这又称为平滑的往返时间，S表示Smoothed。因为进行的是加权平均，因此得出的结果更加平滑）。每当第一次测量到RTT样本时，RTTs值就取为所测量到的RTT样本值。但以后每测量到一个新的RTT样本，就按下式重新计算一次RTTs:

  新的RTTs＝（1-α）×（旧的RTTs）+α×（新的RTT样本）

- 在上式中，0≤α＜l。若α很接近于零，表示新的RTTs值和旧的RTT:值相比变化不大，而对新的RTF样本影响不大，RTT值更新较慢）。若选择α接近于1，则表示新的RTTs值受新的RTT样本的影响较大（RTT值更新较快），RFC2988推荐的α值为1/8，即0.125。用这种方法得出的加权平均往返时间RTTs就比测量出的RTT值更加平滑。

超时计时器设置的超时重传时间RTO

- 超时计时器设置的超时重传时间RTO（ RetransmissionTime-Out）应略大于上面得出的加权平均往返时间RTTs。RFC2988建议使用下式计算RTO：

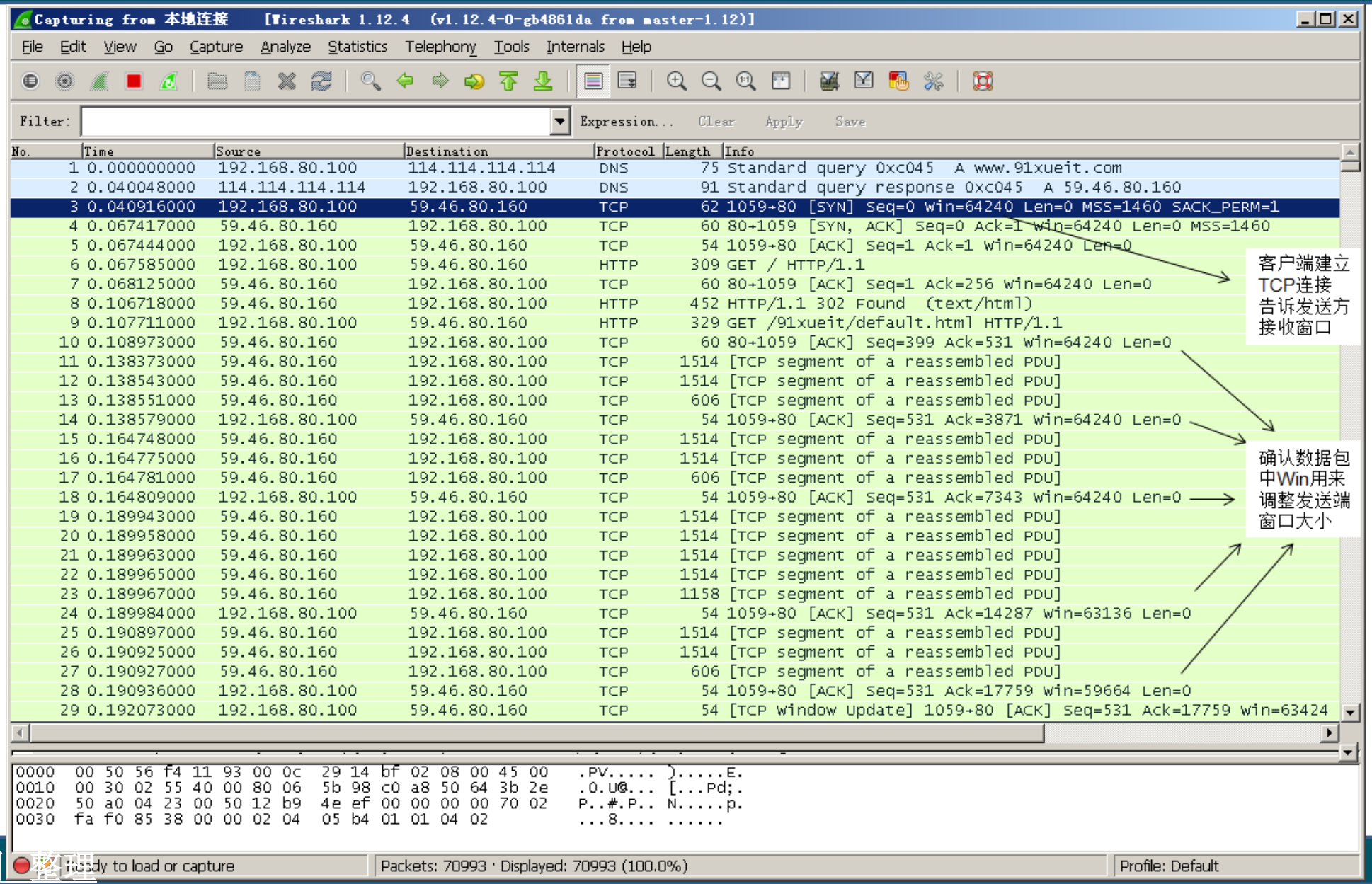
$$RTO=RTTs+4\times RTT_D$$

- 而RTT_D是RTT的偏差的加权平均值，它与RTTs和新的RTT样本之差有关。RFC2988建议这样计算RTT_D。当第一次测量时，RTT_D值取为测量到的RTT样本值的一半。在以后的测量中，则使用下式计算加权平均的RTT_D。

  新的RTT_D=（1-β）×（旧的RTT_D）+ β×｜RTTs-新的RTT样本｜

- 这里β是个小于1的系数，它的推荐值是1/4，即0.25。

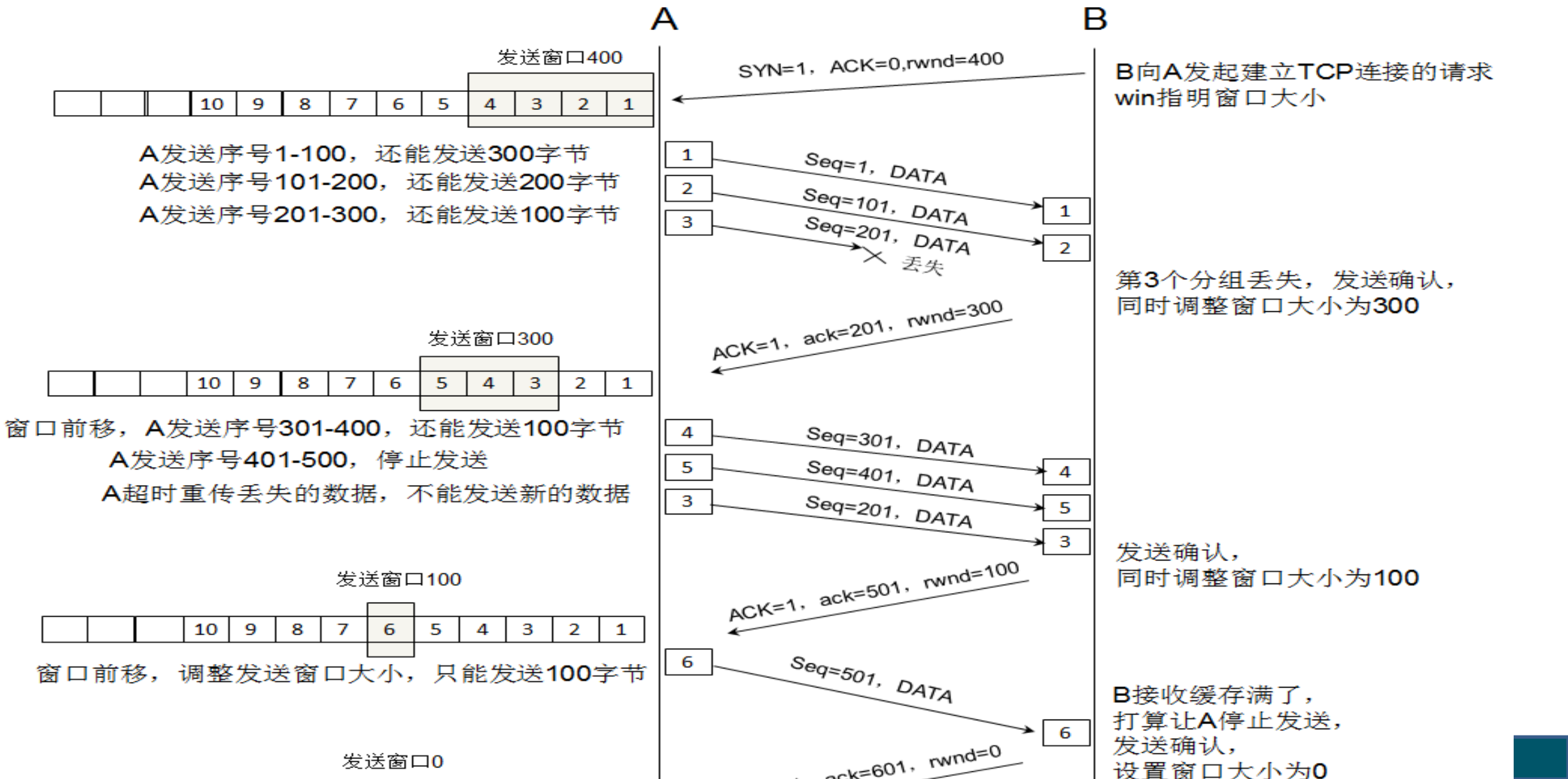
8.5流量控制1



客户端建立TCP连接告诉发送方接收窗口

确认数据包中Win用来调整发送端窗口大小

8.5流量控制2

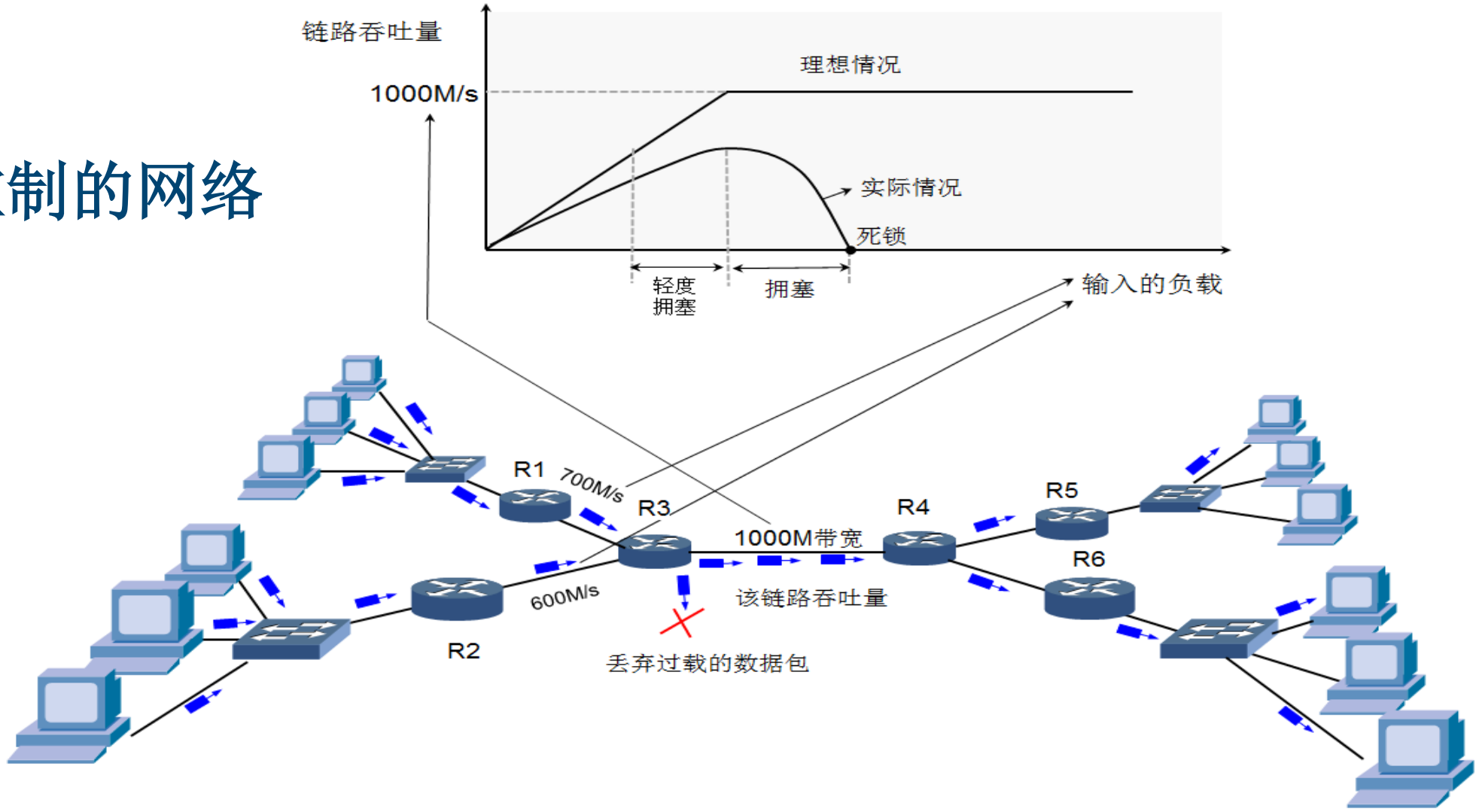

A
B
发送窗口400
10 9 8 7 6 5 4 3 2 1
SYN=1，ACK=0,rwnd=400
B向A发起建立TCP连接的请求
win指明窗口大小
A发送序号1-100，还能发送300字节
A发送序号101-200，还能发送200字节
A发送序号201-300，还能发送100字节
1
2
3
Seq=1，DATA
Seq=101，DATA
Seq=201，DATA
丢失
1
2
第3个分组丢失，发送确认，
同时调整窗口大小为300
发送窗口300
10 9 8 7 6 5 4 3 2 1
ACK=1，ack=201，rwnd=300
窗口前移，A发送序号301-400，还能发送100字节
A发送序号401-500，停止发送
A超时重传丢失的数据，不能发送新的数据
4
5
3
Seq=301，DATA
Seq=401，DATA
Seq=201，DATA
4
5
3
发送窗口100
10 9 8 7 6 5 4 3 2 1
发送确认，
同时调整窗口大小为100
ACK=1，ack=501，rwnd=100
窗口前移，调整发送窗口大小，只能发送100字节
6
Seq=501，DATA
6
B接收缓存满了，
打算让A停止发送，
发送确认，
设置窗口大小为0
发送窗口0
ack=601，rwnd=0

8.6拥塞控制

- 8.6.1拥塞控制的原理
- 8.6.2拥塞控制方法-慢开始和拥塞避免
- 8.6.3拥塞控制方法-快重传和快恢复
- 8.6.4发送窗口的上限

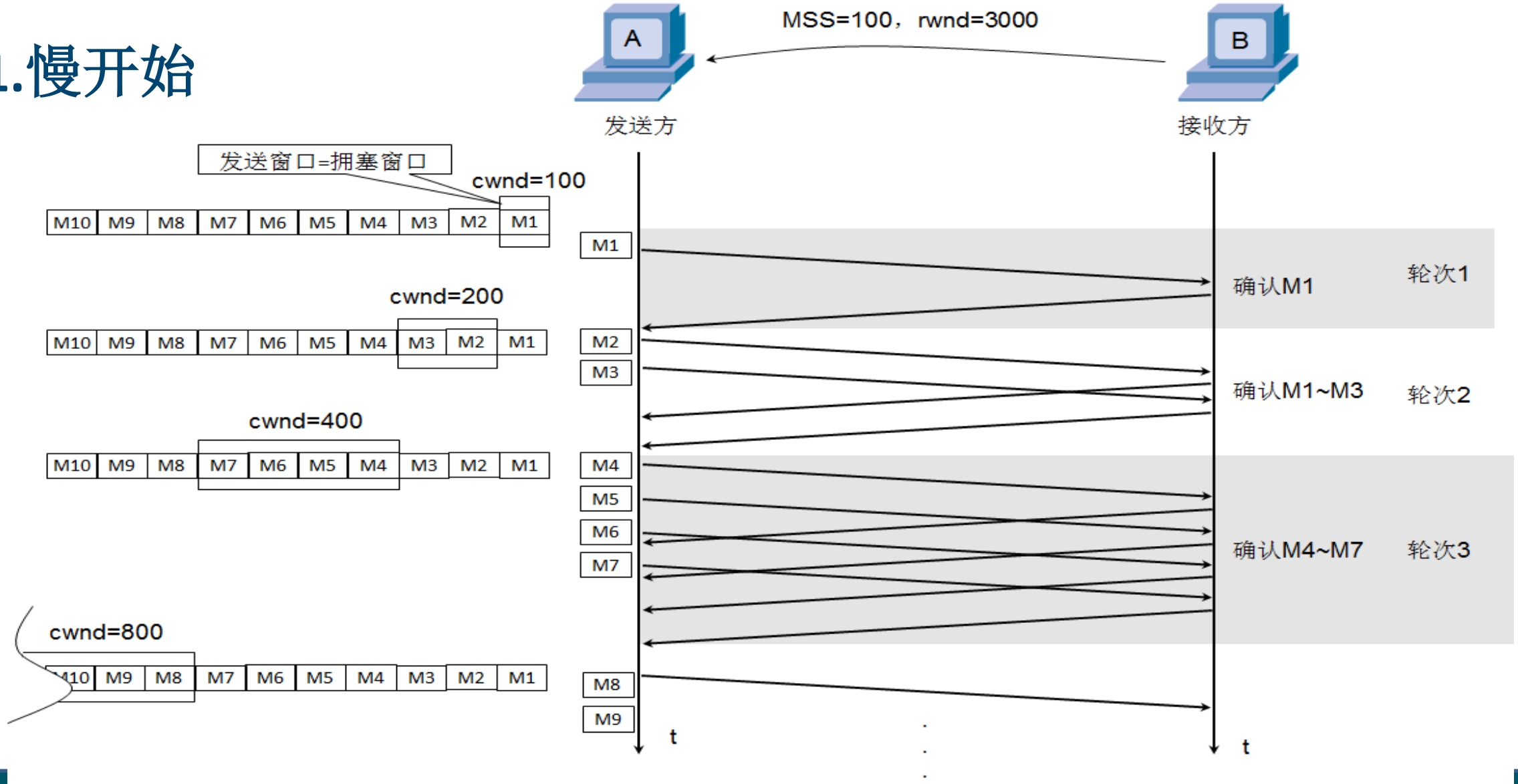
8.6.1拥塞控制的原理

■有拥塞控制的网络



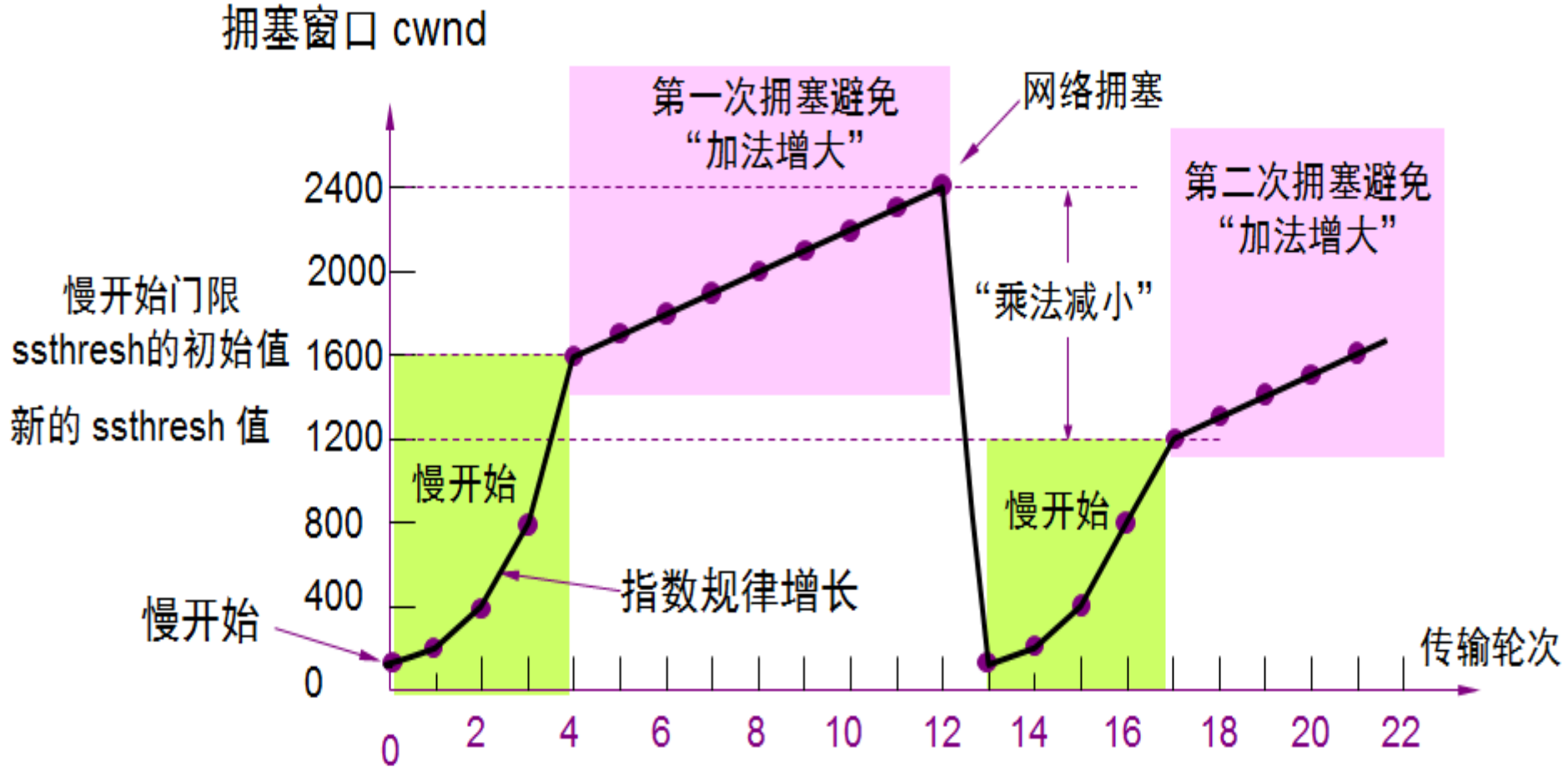
8.6.2拥塞控制方法-慢开始和拥塞避免1

■1.慢开始



8.6.2拥塞控制方法-慢开始和拥塞避免2

■2.拥塞避免算法

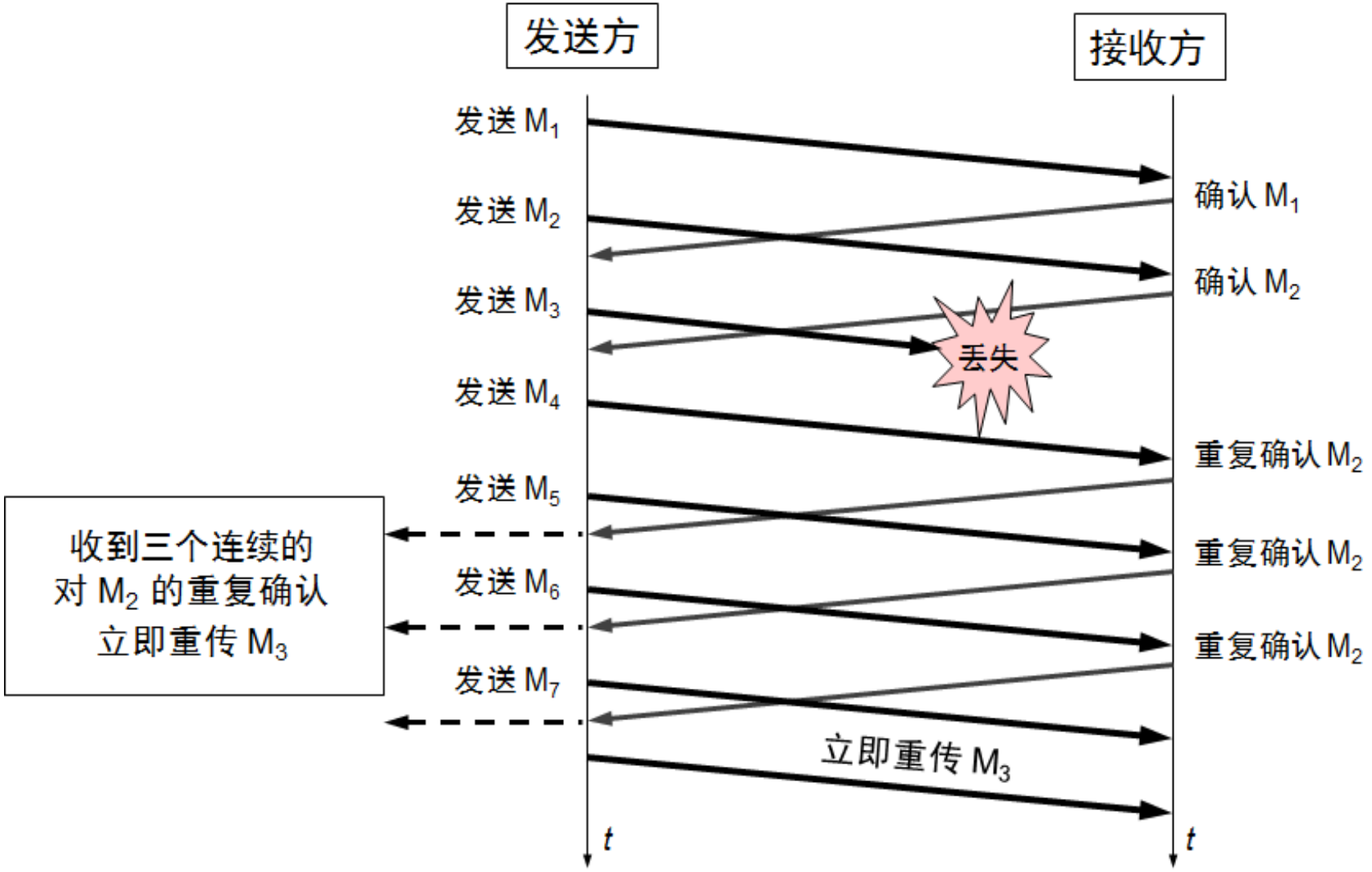


8.6.3拥塞控制方法-快重传和快恢复

■快重传

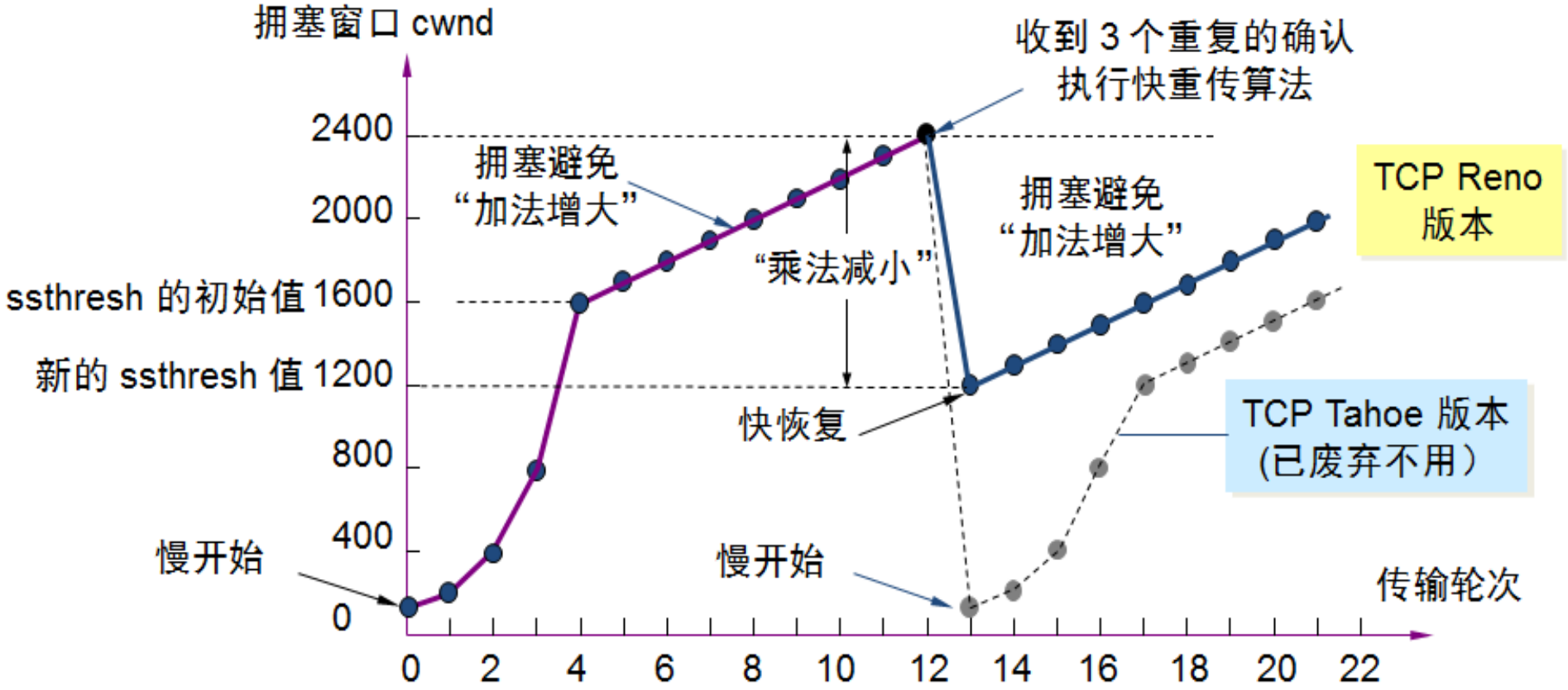
- 快重传算法首先要求接收方每收到一个失序的分组后就立即发出重复确认（为的是使发送方及早知道有分组没有到达对方）而不要等待自己发送数据时才进行捎带确认。
- 快重传算法规定，发送方只要一连收到三个重复确认就应当立即重传对方尚未收到的报文段M3，而不必继续等待为M3设置的重传计时器到期。

快重传


发送方
接收方
发送 M_1
发送 M_2
发送 M_3
发送 M_4
发送 M_5
发送 M_6
发送 M_7
确认 M_1
确认 M_2
丢失
重复确认 M_2
重复确认 M_2
重复确认 M_2
收到三个连续的
对 M_2 的重复确认
立即重传 M_3
立即重传 M_3
t
t

8.6.3拥塞控制方法-快重传和快恢复

■与快重传配合使用的还有快恢复算法



8.6.4发送窗口的上限

- 如果把本节所讨论的拥塞控制和接收方对发送方的流量控制一起考虑，那么很显然，发送方的窗口的上限值应当取为接收方窗口rwnd和拥塞窗口cwnd这两个变量中较小的一个，也就是说：

  发送方窗口的上限值 = Min [rwnd，cwnd]

- 当rwnd<cwnd时，是接收方的接收能力限制发送方窗口的最大值。
- 反之，当cwnd<rwnd时，则是网络的拥塞限制发送方窗口的最大值。
- 也就是说，rwnd和cwnd中较小的一个控制发送方发送数据的速率。

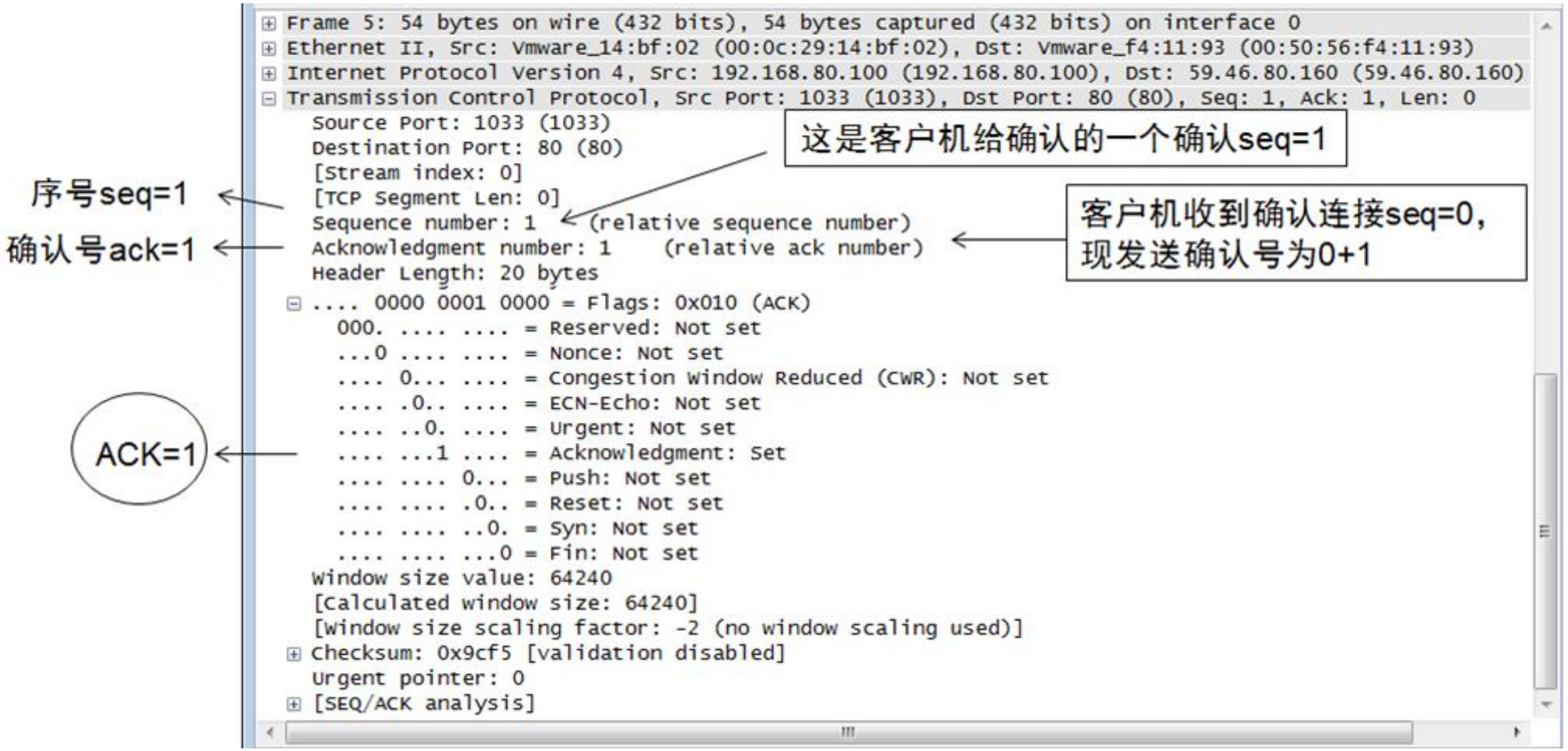
8.7 TCP连接管理

- 8.7.1 TCP的连接建立
- 8.7.2TCP连接释放
- 8.7.3实战： 查看TCP释放连接的数据包
- 8.7.4实战： SYN攻击

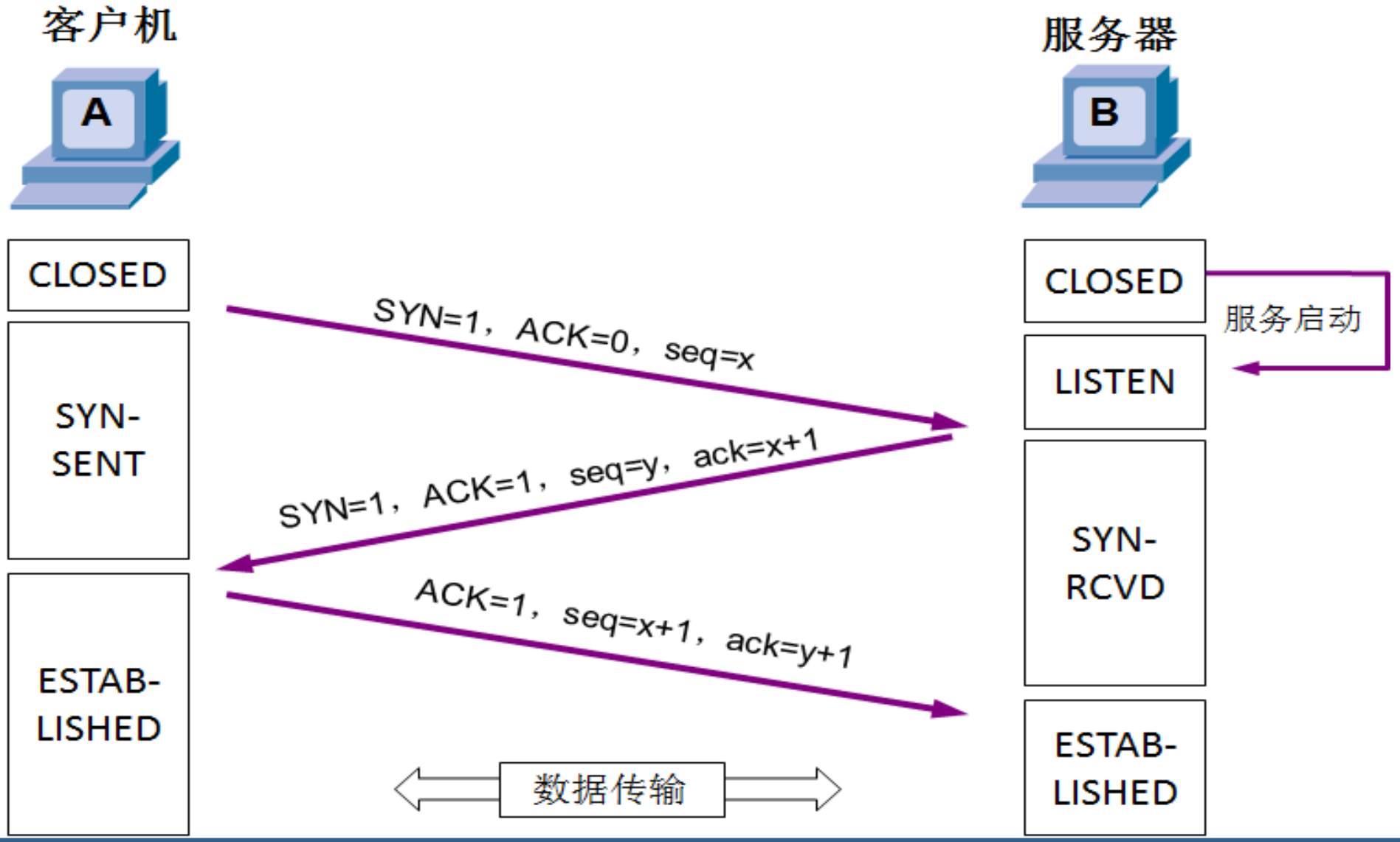
8.7.1 TCP建立连接-请求建立TCP连接的数据包

8.7.1 TCP建立连接- TCP连接确认数据包

8.7.1 TCP建立连接-确认的确认



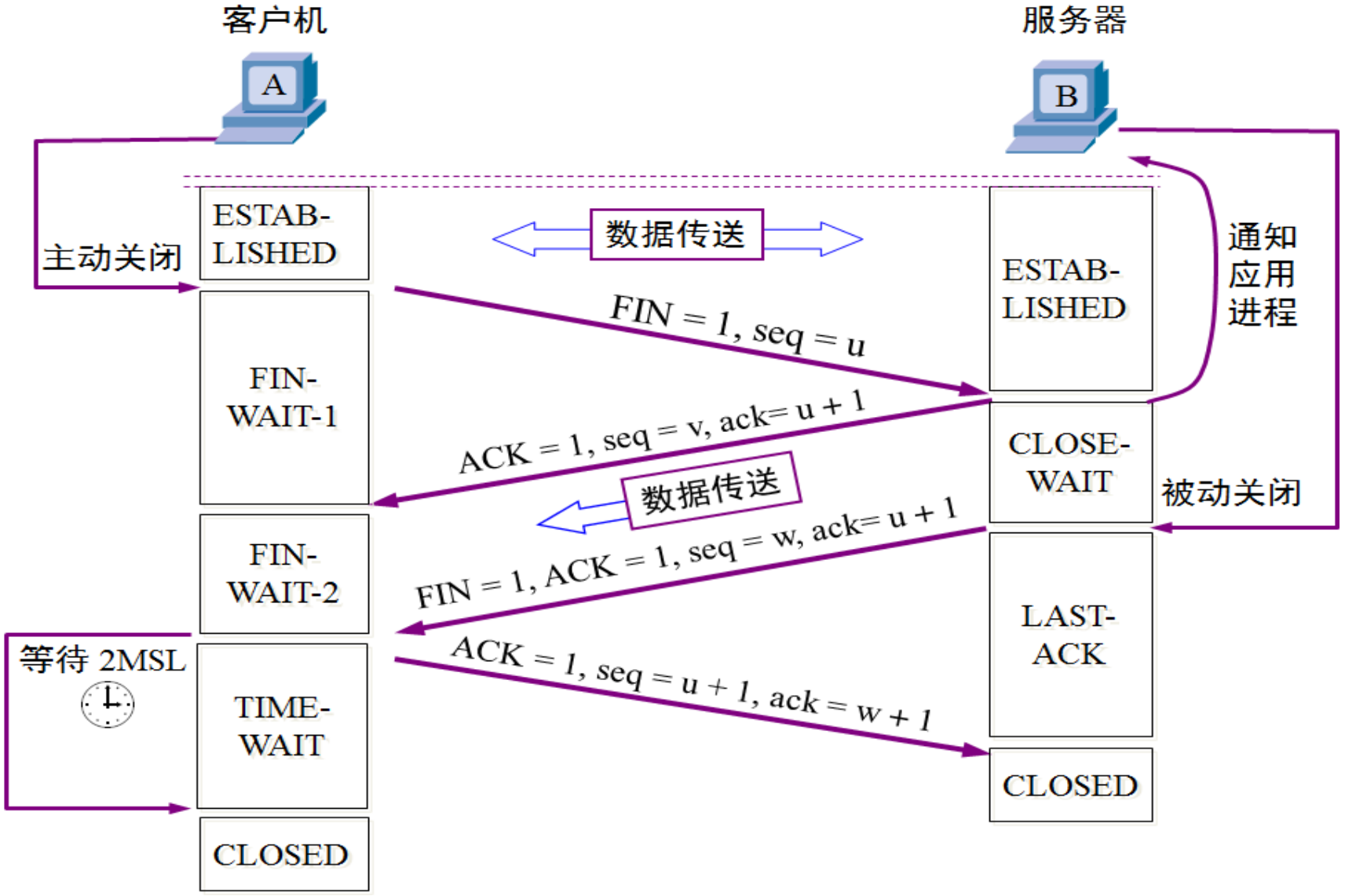
8.7.1 TCP的连接建立的过程



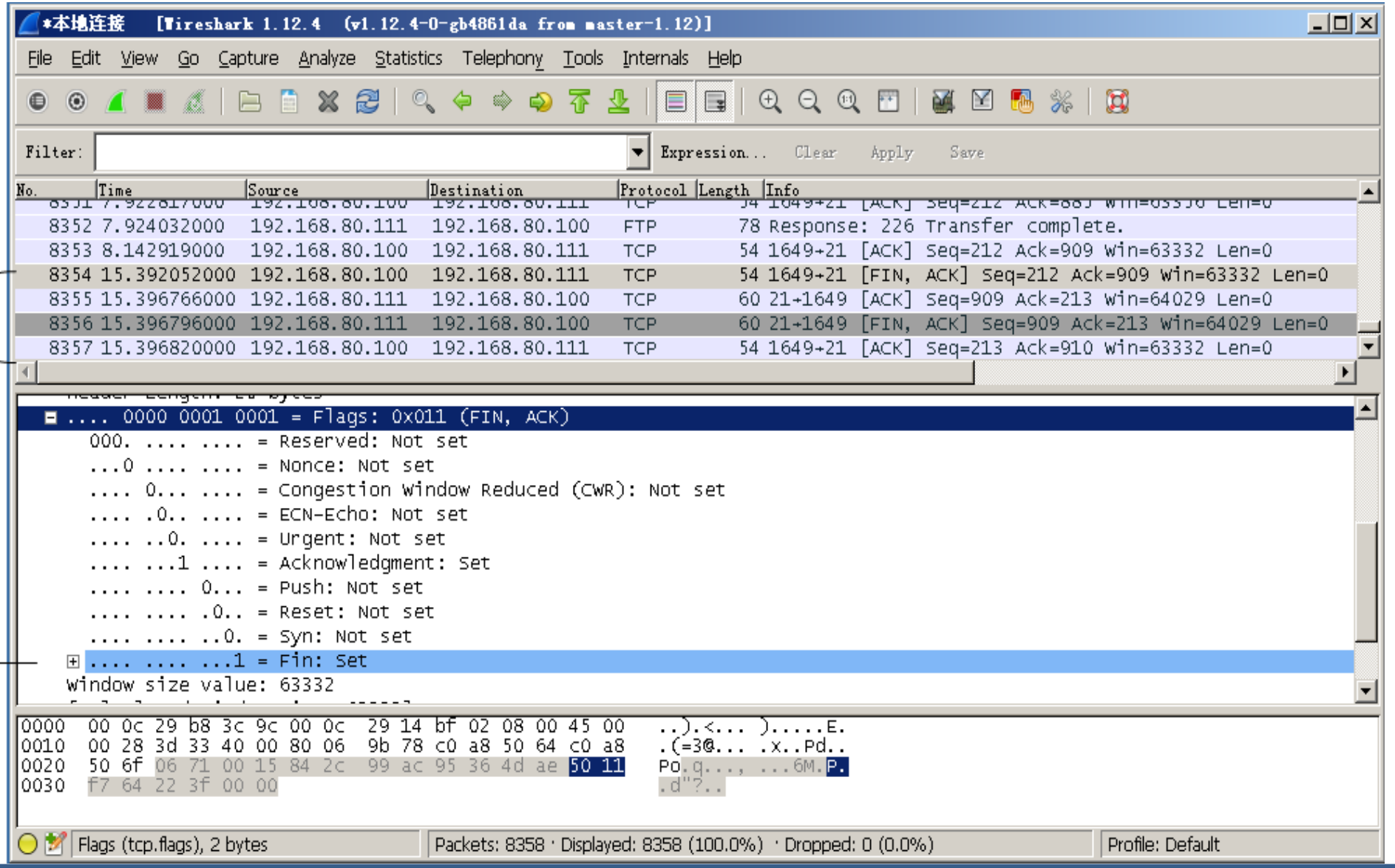
8.7.2TCP连接释放

- TCP协议通信结束后，需要释放连接，TCP连接释放过程比较复杂，我们仍结合双方状态的改变来阐明连接释放的过程。
- 数据传输结束后，通信的双方都可释放连接。如图8-74所示，现在A和B都处于ESIABLISHED状态，A的应用进程先向其TCP发出连接释放报文段，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FIN置1，其序号seq=u，它等于前面已传送过的数据的最后一个字节的序号加1。这时A进入FIN-WAIT-1（终止等待I）状态，等待B的确认。

8.7.2TCP连接释放


客户机
服务器
A
B
ESTAB-LISHED
ESTAB-LISHED
主动关闭
数据传送
FIN = 1, seq = u
通知应用进程
FIN-WAIT-1
ACK = 1, seq = v, ack= u + 1
CLOSE-WAIT
数据传送
被动关闭
FIN-WAIT-2
FIN = 1, ACK = 1, seq = w, ack= u + 1
LAST-ACK
等待 2MSL
ACK = 1, seq = u + 1, ack = w + 1
TIME-WAIT
CLOSED
CLOSED

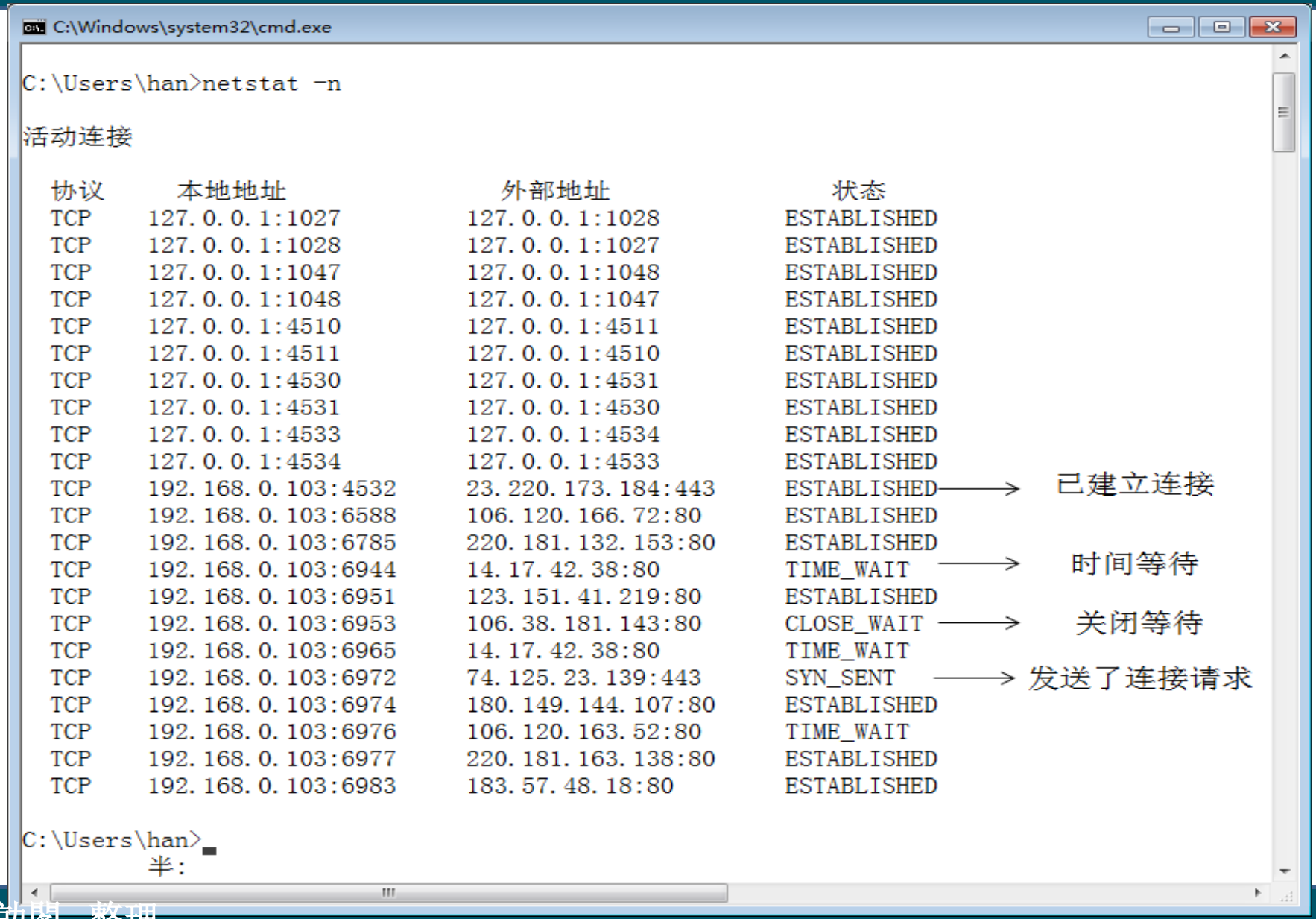
8.7.3实战：查看TCP释放连接的数据包1



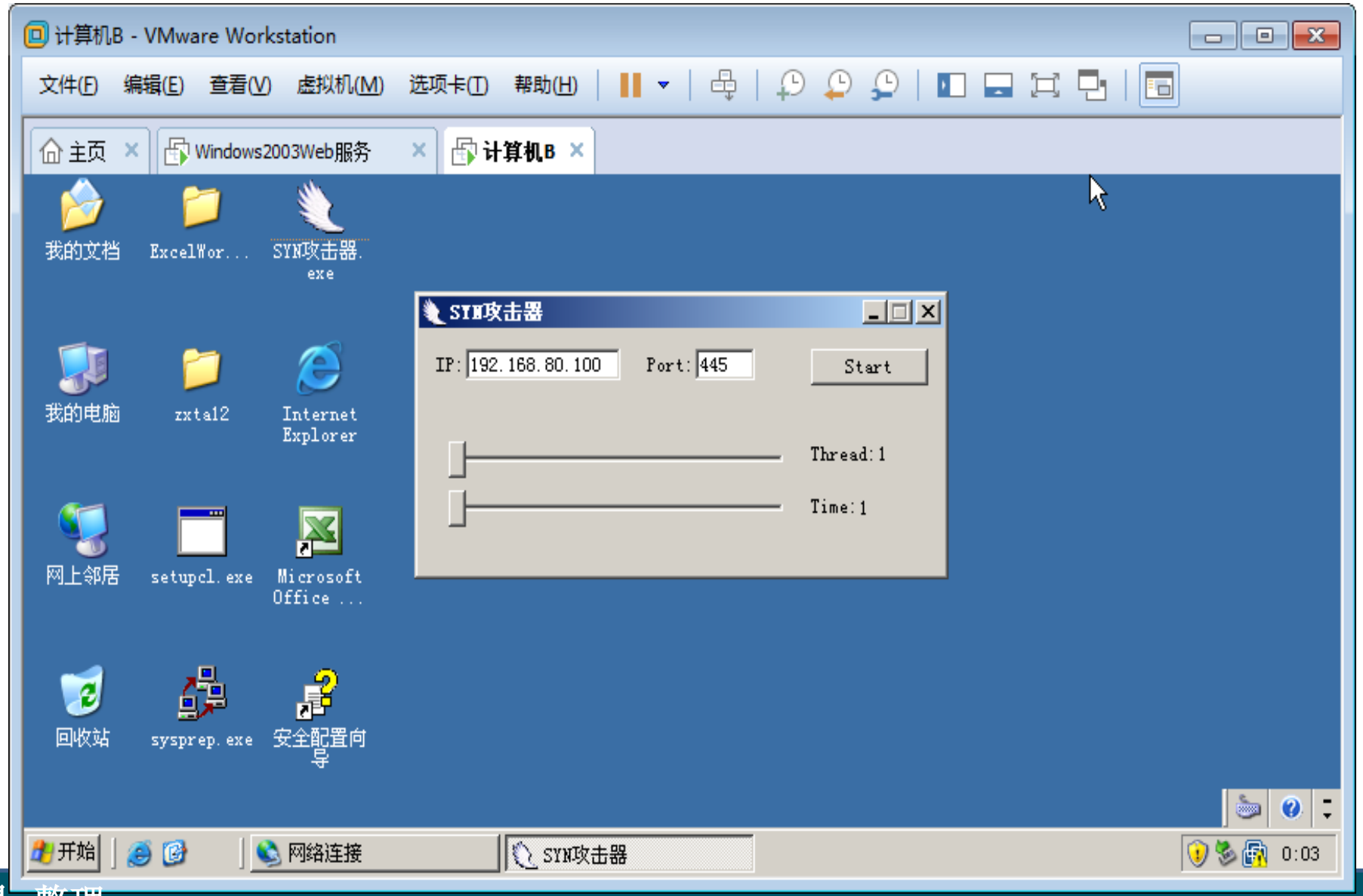
释放连接的4个数据包

FIN=1

8.7.3实战：查看TCP释放连接的数据包2



8.7.4实战：SYN攻击1



8.7.4实战：SYN攻击2