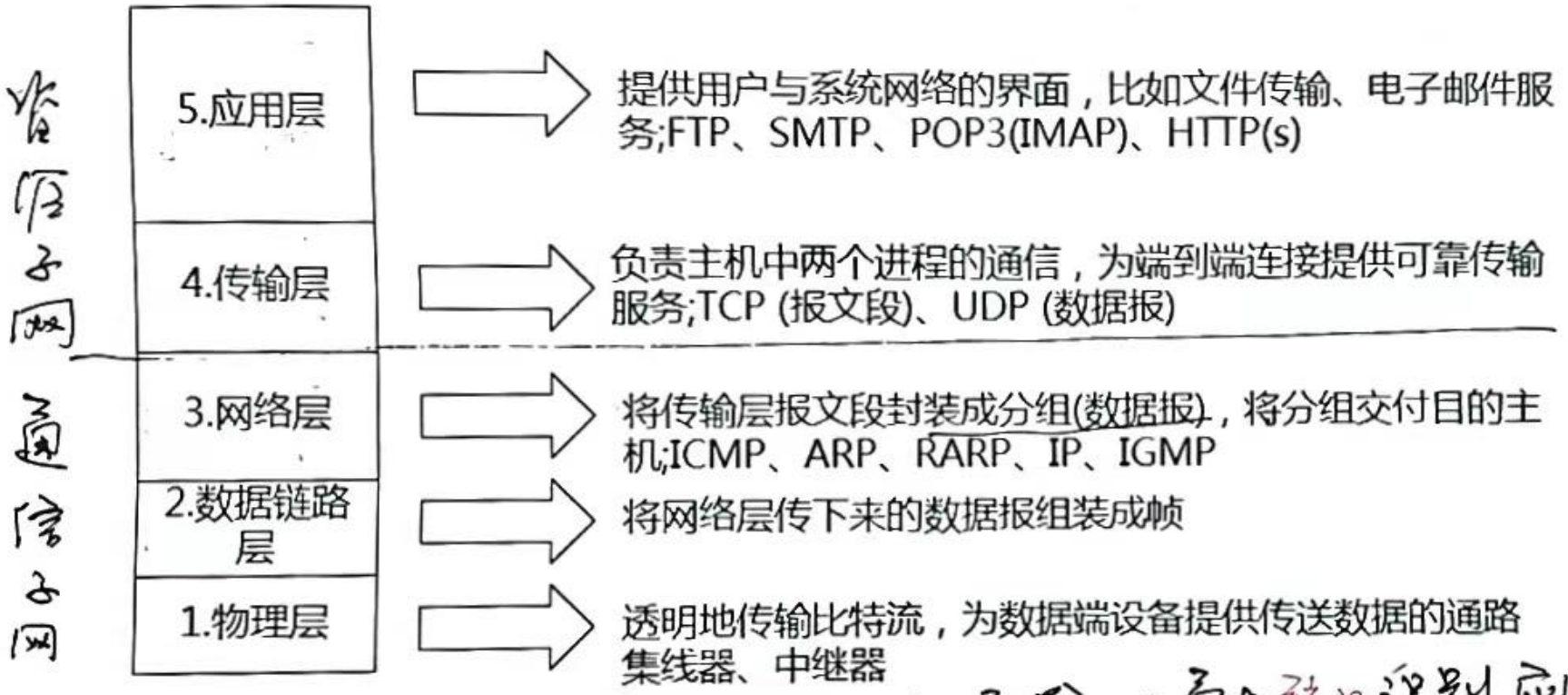
传输层



传输层：两台主机的进程之间通信（通过端口识别应用程序）

网络层：两台主机之间进行通信

——网络层只负责将IP数据报投递到目的主机

*使用网络层服务，向应用层提供服务

1.传输层都有哪些功能?

•传输层提供的服务

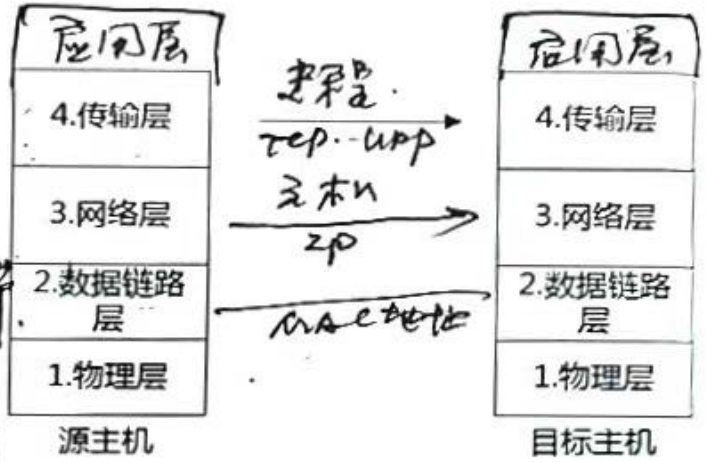
1)传输层的功能

为应用层提供服务，使用网络层的服务

提供进程与进程之间的逻辑通信

复用和分用

差错检测



两种协议:UDP 和 TCP

UDP:

(1)无连接的用户数据报协议

(2)不可靠、时延小、适用小文件

TCP:

(1)有连接的传输控制协议

(2)不提供广播和多播服务

(3)可靠、时延大、适用大文件

2)传输层的寻址与端口

端口标识主机中的应用进程

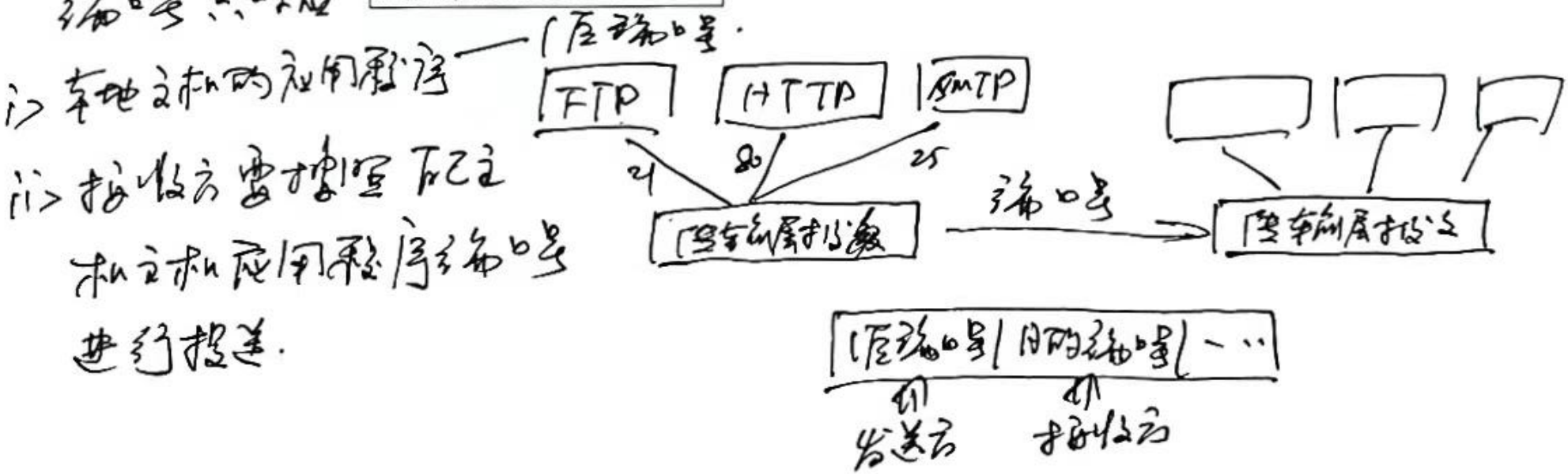
端口:即传输层的服务访问点(TSAP)
数据链路层SAP是MAC地址
网络层SAP是IP地址
传输层SAP是端口

端口号:
只有本地意义
端口号长度16bit，共65536个

应用程序	FTP	TELNET	SMTP	DNS	TFTP	HTTP	SNMP
熟知端口号	21	23	25	53	69	80	161

通过IP识别主机，通过端口号识别主机中的应用进程，所以:
套接字=(IP，端口号)
套接字(Socket)唯一标识了网络中的一台主机和它的一个进程。

熟知端口号:0-1023
登记端口号:1024-49151
短暂端口号:49152-65535



2.传输层有哪些通信方式?

- 传输层涉及到的主要协议

UDP 协议

- **UDP 数据报**

在 IP 数据报基础上增加了复用、分用和差错检测功能

UDP 主要特点:

无连接，减少开销和发送前的时延

尽最大努力交付，即不保证可靠交付

面向报文，一次传输一个报文

无拥塞控制，适合实时应用

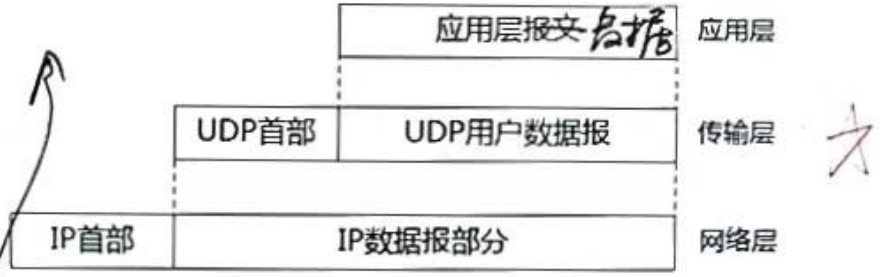
首部开销小，8B;TCP 首部 20B

- **UDP 校验**

伪首部:不向下传递，不向上递交

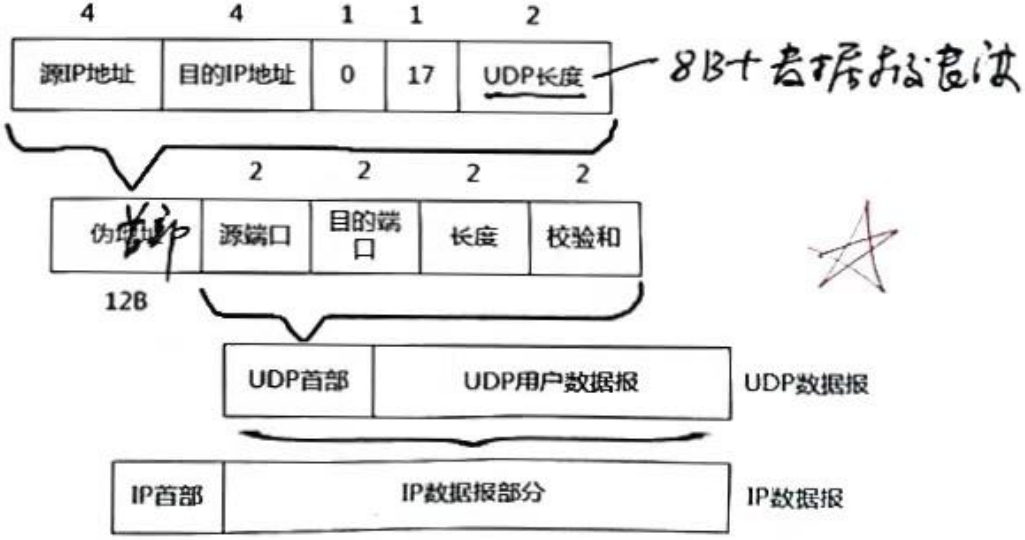
UDP 长度:UDP 首部 8B+数据部分长度

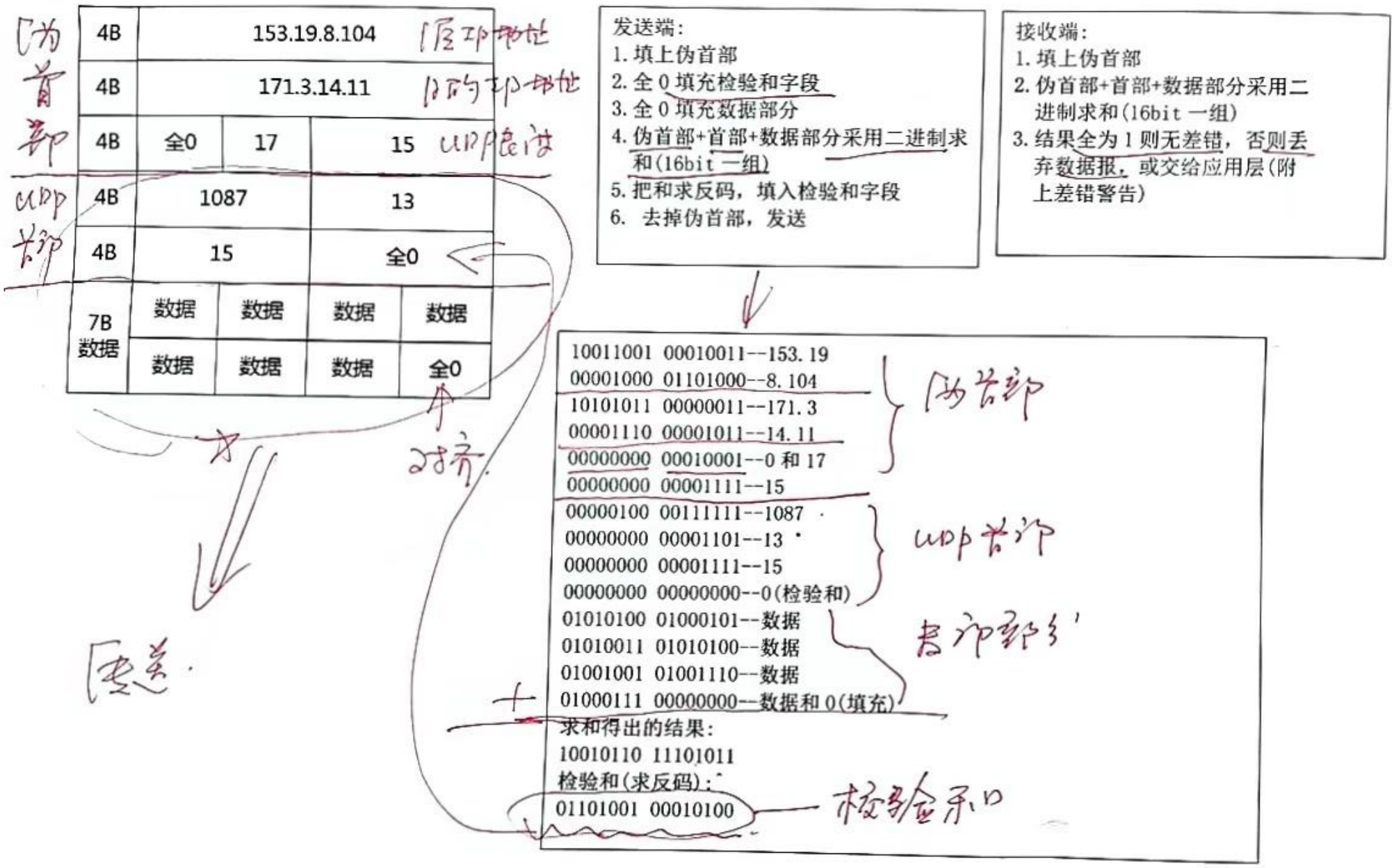
17:封装 UDP 报文的 IP 数据报首部协议



4B	源端口号，16位	目的端口号，16位
4B	数据报长度，16位	校验和，16位
数据字段	数据	

UDP首部格式




伪首部
UDP首部
4B 153.19.8.104
4B 171.3.14.11
4B 全0 17 15
4B 1087 13
4B 15 全0
7B 数据 数据 数据 数据 数据 数据 数据 全0
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
4. 伪首部+首部+数据部分采用二进制求和(16bit 一组)
5. 把和求反码，填入检验和字段
6. 去掉伪首部，发送
接收端:
1. 填上伪首部
2. 伪首部+首部+数据部分采用二进制求和(16bit 一组)
3. 结果全为 1 则无差错，否则丢弃数据报，或交给应用层(附上差错警告)
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)
求和得出的结果:
10010110 11101011
检验和(求反码):
01101001 00010100
伪首部
UDP首部
数据部分
检验和
发送

注意:

1)校验时，若 UDP 数据报部分的长度不是偶数个字节，则需填入一个全 0 字节，如图中所示。但是此字节和伪首部一样，是不发送的。

2)如果 UDP 校验和校验出 UDP 数据报是错误的，那么可以丢弃，也可以交付给上层，但是需要附上错误报告，即告诉上层这是错误的数据报。

3)通过伪首部，不仅可以检查源端口号、目的端口号和 UDP 用户数据报的数据部分，还可以检查 IP 数据报的源 IP 地址和目的地址。

这种简单的差错校验方法的校错能力并不强，但它的好处是简单、处理速度快。

TCP 协议 ☆☆☆

- **TCP 协议的特点**

1)面向连接(虚连接)的传输协议

2)每一条 TCP 连接只能有两个端点:点对点

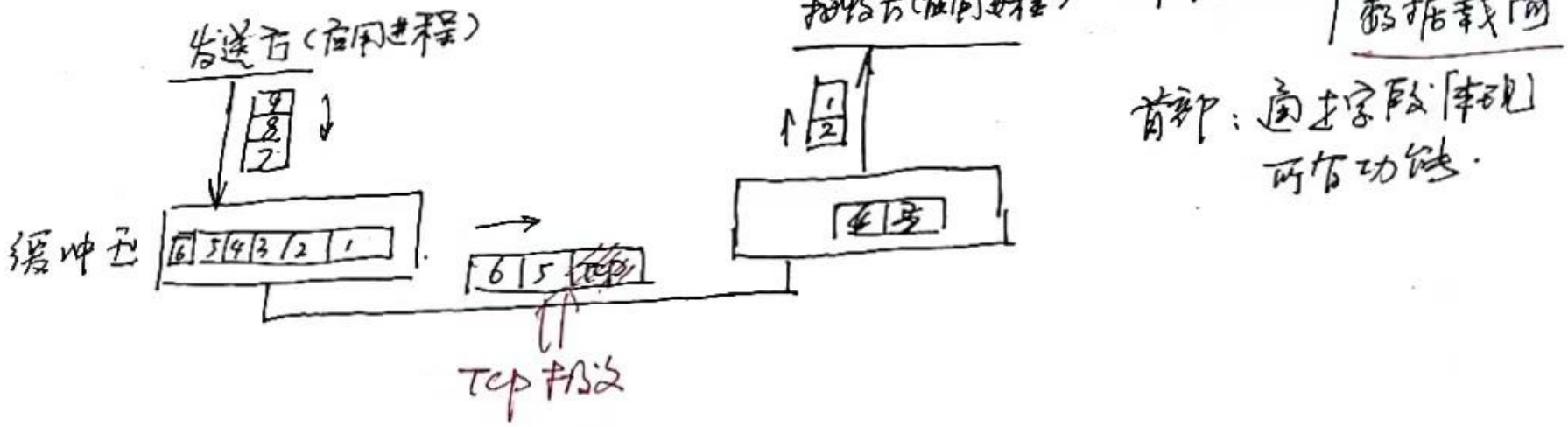
3)提供可靠交付服务

4)无差错、不丢失、不重复、按序到达(可靠有序，不丢不重)

5)提供全双工通信--发送缓存;接收缓存;

☆ 6)面向字节流 — 可靠传输

TCP 发送数据时，从发送方缓存中取为取为一部分或全部字节并添加首部使之成为 TCP 报文段后进行传送。

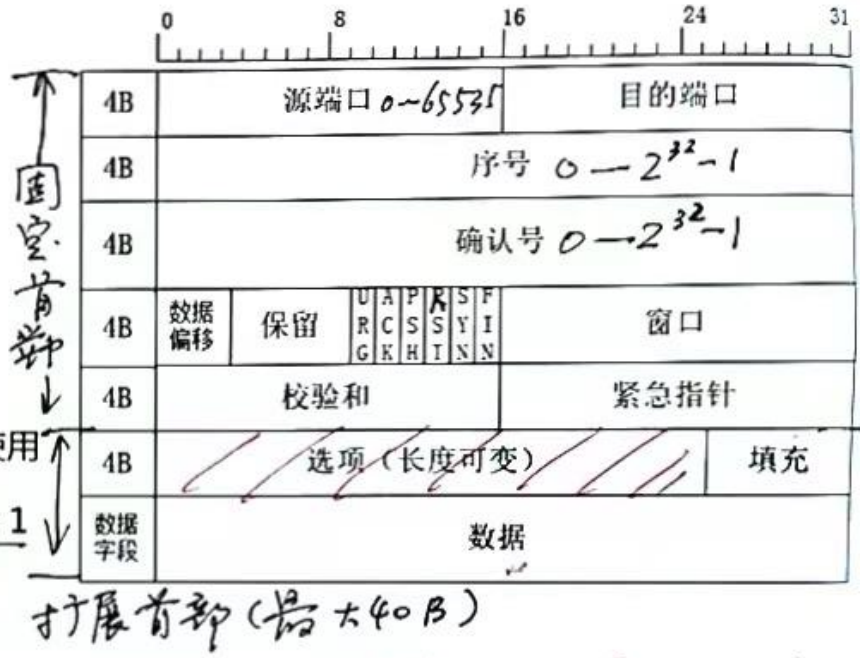


TCP 报文段 { 首部 / 数据载荷

首部：由主字段[标识]所有功能。

TCP 协议

- **TCP 报文段首部格式**



源端口和目的端口

序号:本报文段所发送数据的第一个字节的序号

确认号:期望收到下个报文段第一个字节的序号

数据偏移:即首部长度，报文段数据与报文段起始的距离

紧急位 URG: 为 1 时有紧急数据，优先级高，配合紧急指针使用

确认位 ACK:为 1 时确认号有效，连接后报文段须把 ACK 置为 1

推送位 PSH: 为 1 时接收方尽快交付，不需等缓存满

复位 RST: 为 1 时 TCP 连接出错，须释放后重连

同步位 SYN: 为 1 时表明连接请求/连接接受报文

终止位 FIN: 为 1 时此报文段已发完，要求释放连接

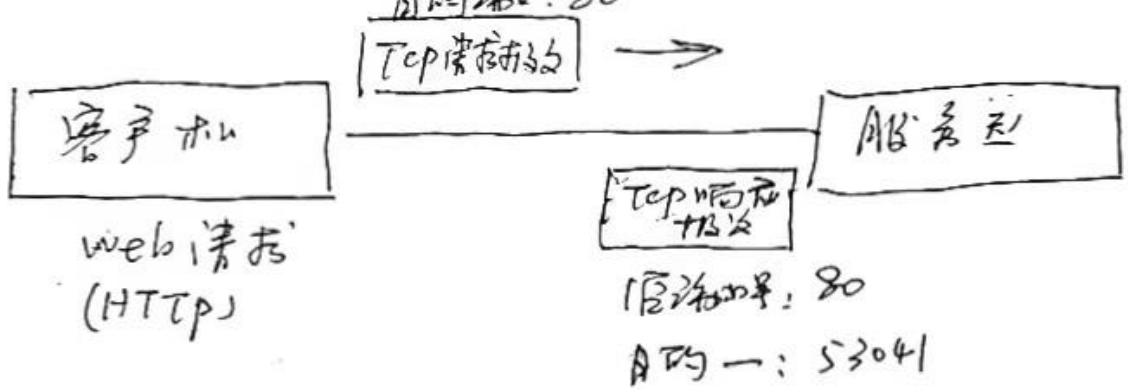
窗口:发送方的接收窗口，允许对方发送的数据量

检验和:检验首部+数据，要加上 12B 伪首部，第 4 个字段为 6

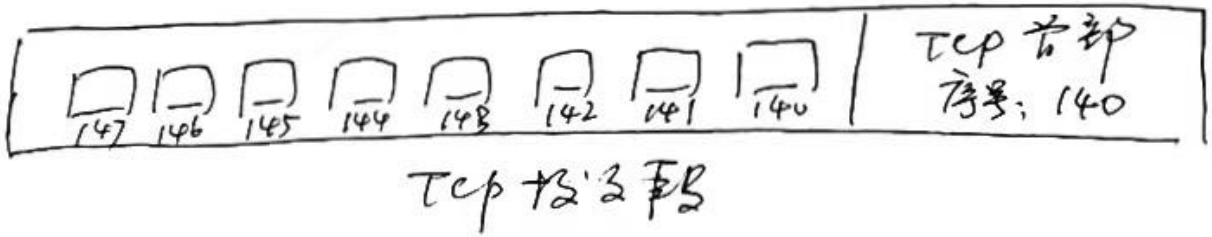
紧急指针:URG 为 1 时有意义，本报文段紧急数据字节数

选项:最大报文段长度 MSS、窗口扩大、时间戳、选择确认等

① 源端口和目的端口：

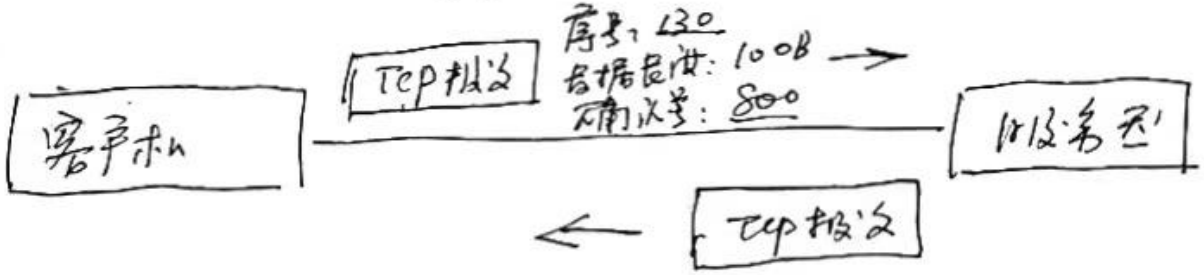


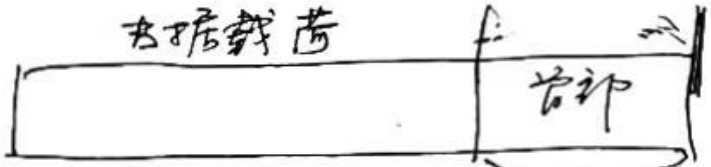
② 序号：



确认号 ack
- 指为期望接收对方下一个报文的数据载荷的第一个字节序号
- 对之前收到的数据的确认.

确认号 ack=x:
- 希望接收 x-1 之前的数据
- 希望接收序号为 x 的数据报文.





数据偏移.

TCP报文 数据偏移即首部长度.

↳ 占4bit.
最长4B字节.

固定长度: 20B 数据偏移: 0101

首部最大长度: (1111)₂ × 4B
= 15 × 4 = 60B

窗口：—— 16bit —— 以字节为单位使用，指收
—— 指为发送方的发送窗口（指收方）
—— 流量控制：{ 防止指收方被发送方发送数据大小的限制
以指收方的指要能力控制发送方的发送能力

☆发送方一次可发送多少字节的数据
1> 指收方给发送方设置的窗口值
2> 拥塞窗口大小
窗口值 = min（拥塞窗口，指收窗口）

- **TCP 连接管理**

TCP 连接的三个阶段:建立->传送->释放

客户/服务模式(C/S):

主动发起连接建立的应用进程叫客户(Client)

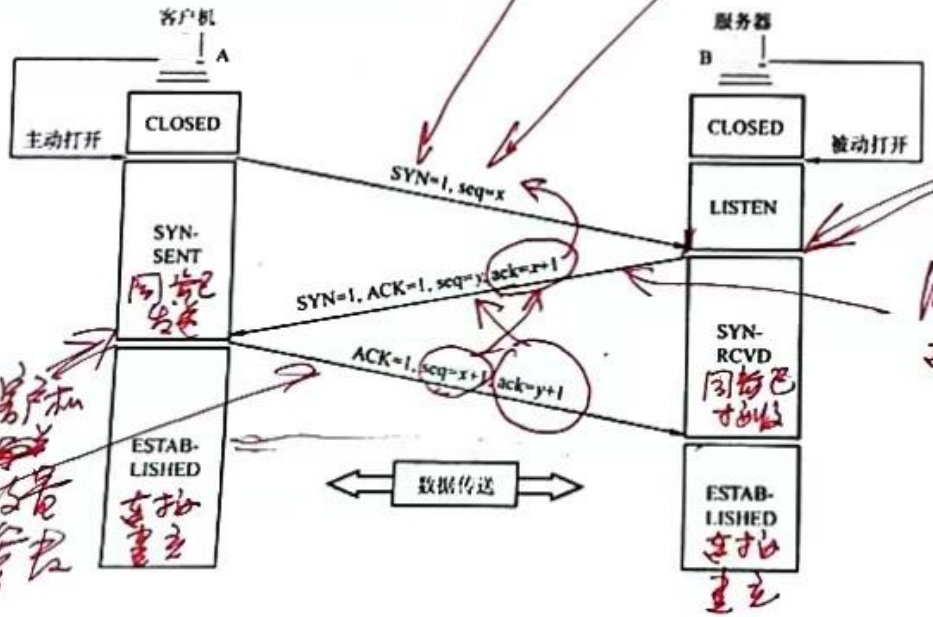
被动等待连接建立的应用进程叫服务器(Server)

三次握手建立连接:

1)客户机向服务器发送一个连接请求报文段

2)服务器同意连接，分配缓存和变量，向客户机发回确认

3)客户机收到确认报文段，分配缓存和变量，向服务器给出确认



TCP连接建立的三个问题：

① 保证TCP双方确认对方存在

客户：SYN=1 seq=x ← 确认 ack=x+1

服务：SYN=1 seq=y

⇒保证双方都知道对方存在

② 保证TCP双方协商参数

（最大窗口值，是否使用窗口扩大选项，时间戳选项）。

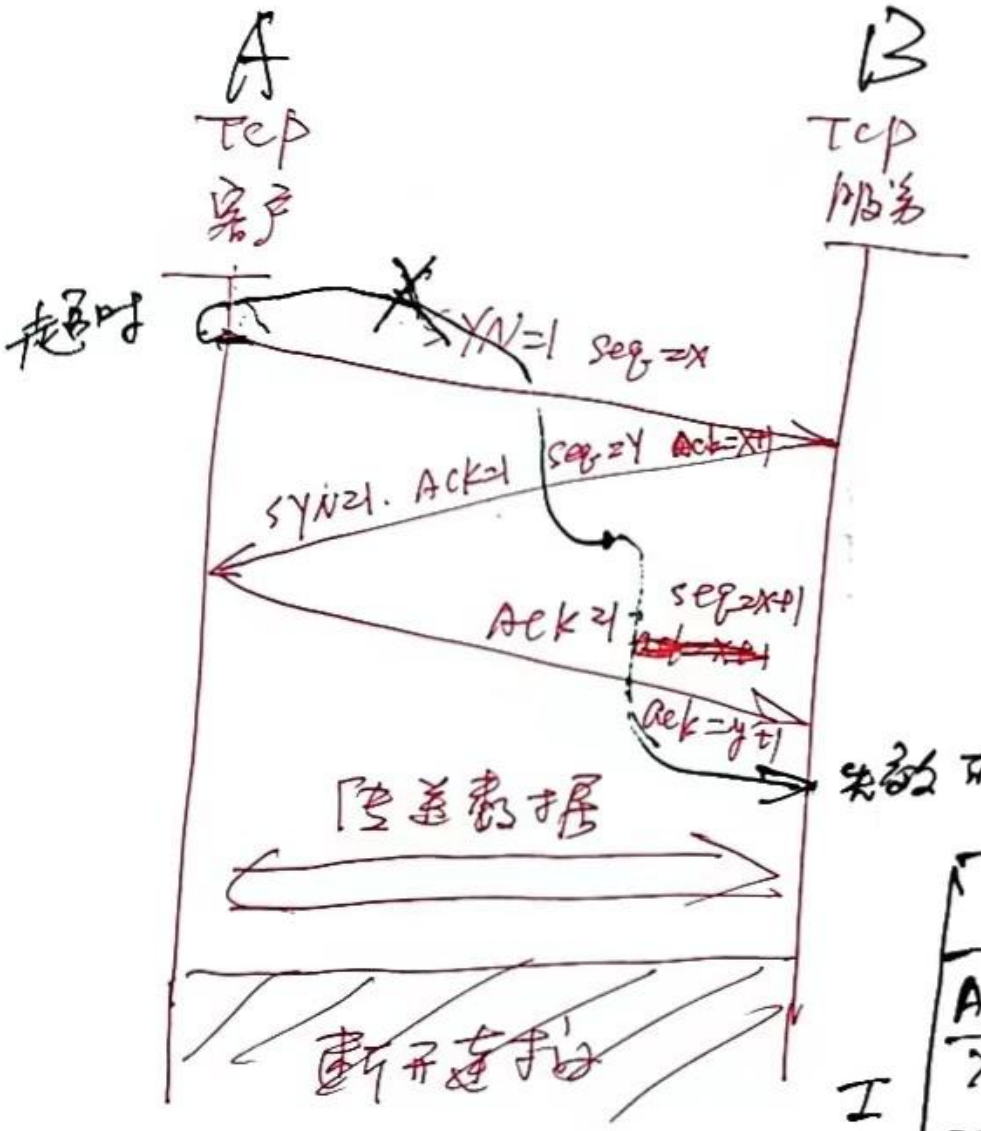
③ TCP双方分配资源（缓存大小，连接表选项）。

可不可以两次握手？——不可以。

——防止已失效的连接请求报文段突然又传送到TCP服务器而导致错误。

三次握手 —— 确认既能和对方的发送、接收都正常。

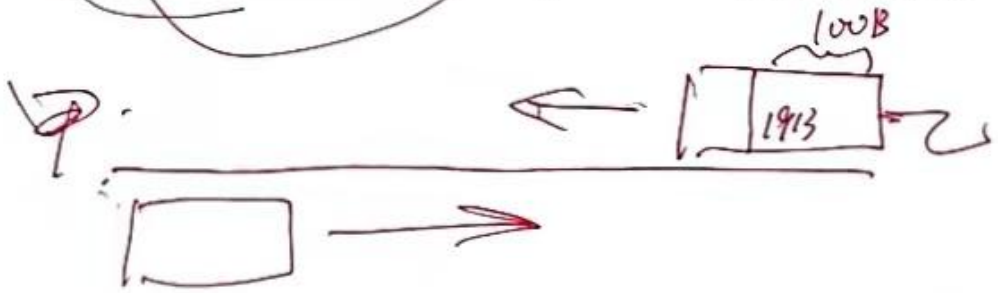
台式机



	A 确认				B 确认			
	A发	A收	B发	B收	B发	B收	A发	A收
I	×	×	×	×	×	√	√	×
II	√	√	√	√	×	√	√	×
III	√	√	√	√	√	√	√	√

【2013 统考真题】主机甲与主机乙之间已建立一个 TCP 连接，双方持续有数据传输，且数据无差错与丢失，若甲收到一个来自乙的 TCP 段，该段的序号为 1913、确认序号为 2046、有效载荷为 100B，则甲立即发送给乙的 TCP 段的序号和确认序号分别是(　)。

A.2046、2012　　B.2046、2013　　C.2047、2012　　D.2047、2013

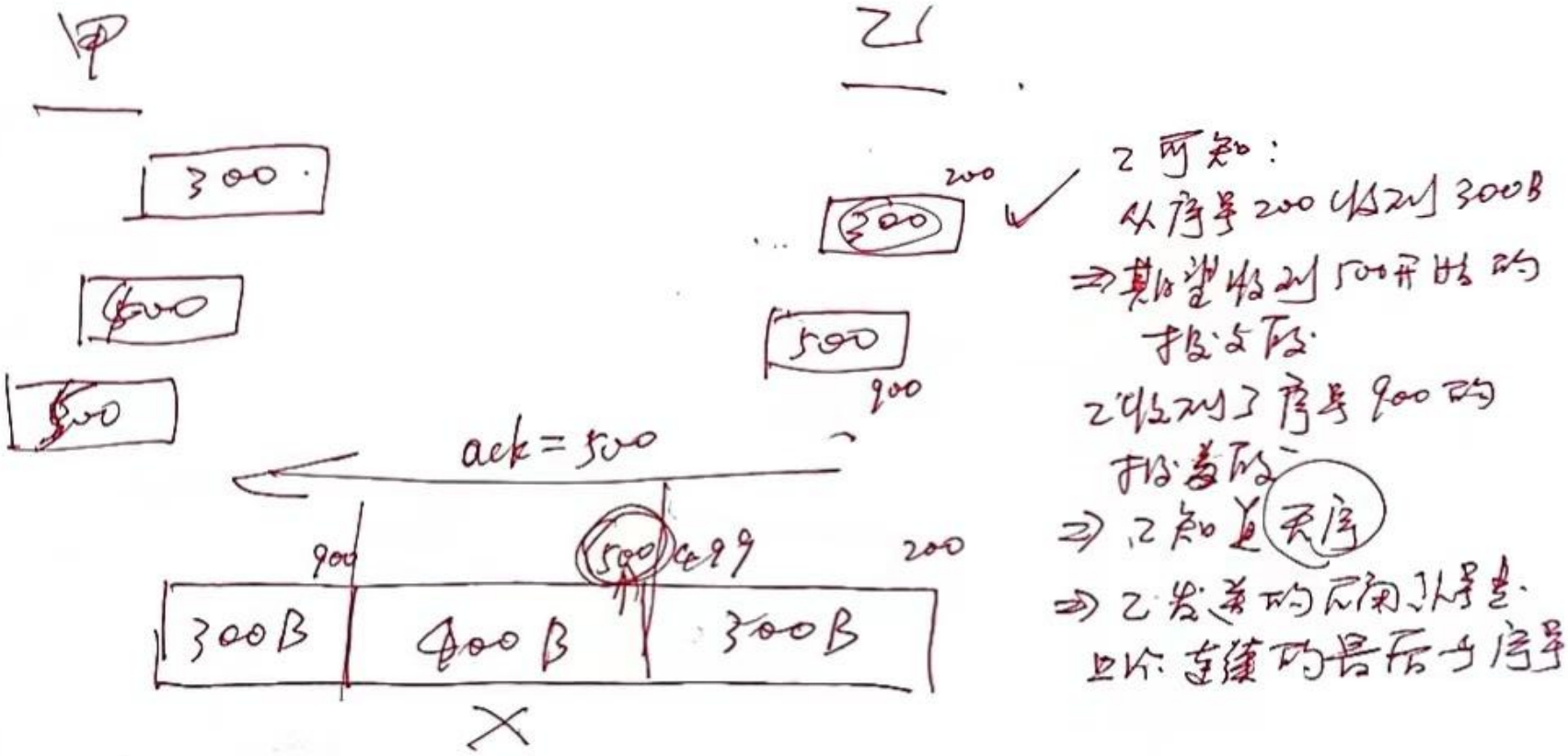


乙→甲：seq=1913. ack=2046 有效载荷：100B

甲→乙：seq=2046 ack=2013.

【2011 统考真题】主机甲与主机乙之间已建立一个 TCP 连接，主机甲向主机乙发送了 3 个连续的 TCP 段，分别包含 300B、400B 和 500B 的有效载荷，第 3 个段的序号为 900，若主机乙仅正确接收到第 1 个段和第 3 个段，则主机乙发送给主机甲的确认序号是(　)。

A.300　　B.500　　C.1200　　D.1400



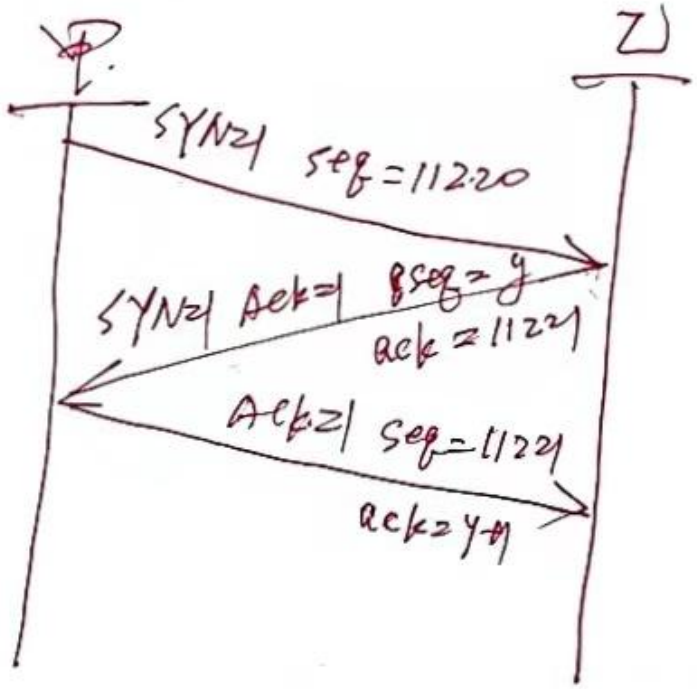
【2011 统考真题】主机甲向主机乙发送一个(SYN=1，seq=11220)的 TCP 段，期望与主机乙建立 TCP 连接，若主机乙接受该连接请求，则主机乙向主机甲发送的正确的 TCP 段可能是(　)。

A.(SYN=0，ACK=0，seq=11221，ack=11221)

B.(SYN=1，ACK=1，seq=11220，ack=11220)

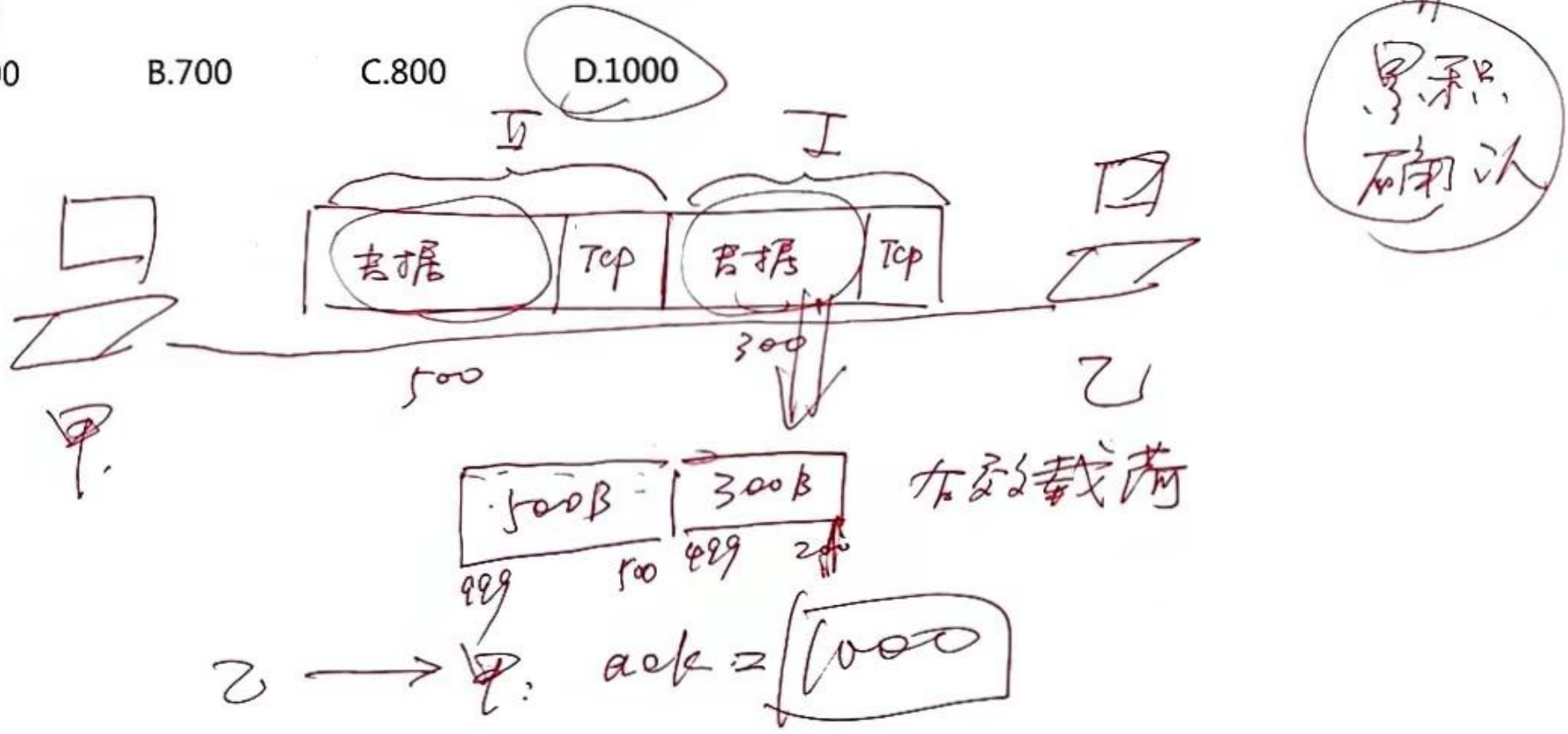
C.(SYN=1，ACK=1，seq=11221，ack=11221)

D.(SYN=0，ACK=0，seq=11220，ack=11220)



【2009 统考真题】主机甲与主机乙之间已建立一个 TCP 连接，主机甲向主机乙发送了两个连续的 TCP 段，分别包含 300B 和 500B 的有效载荷，第一个段的序列号为 200，主机乙正确接收到这两个数据段后，发送给主机甲的确认序列号是( )。

A.500　　B.700　　C.800　　D.1000



四次挥手释放连接:

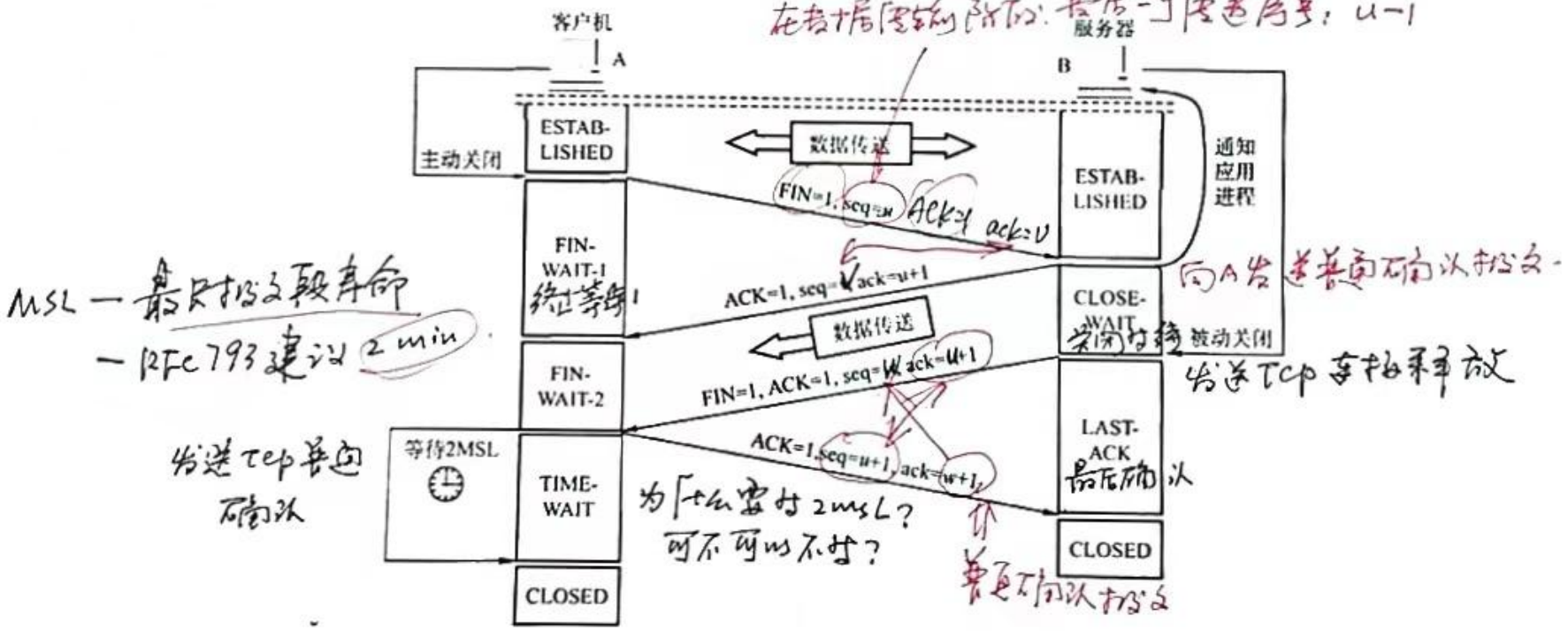
1)客户机发送连接释放报文段，停发数据，主动关闭连接

2)服务器回复确认报文段，客户到服务器方向已释放

3)服务器发送释放连接报文段，主动关闭连接

4)客户端回复确认报文段，等待超时(2MSL)后彻底关闭 4

FIN=1 的报文段.
— 释放连接报文段
— 不携带数据. 但要消耗一个序号



在数据传输时所发: 最后一个序号: u-1

MSL — 最长报文段寿命
— RFC 793 建议 2 min

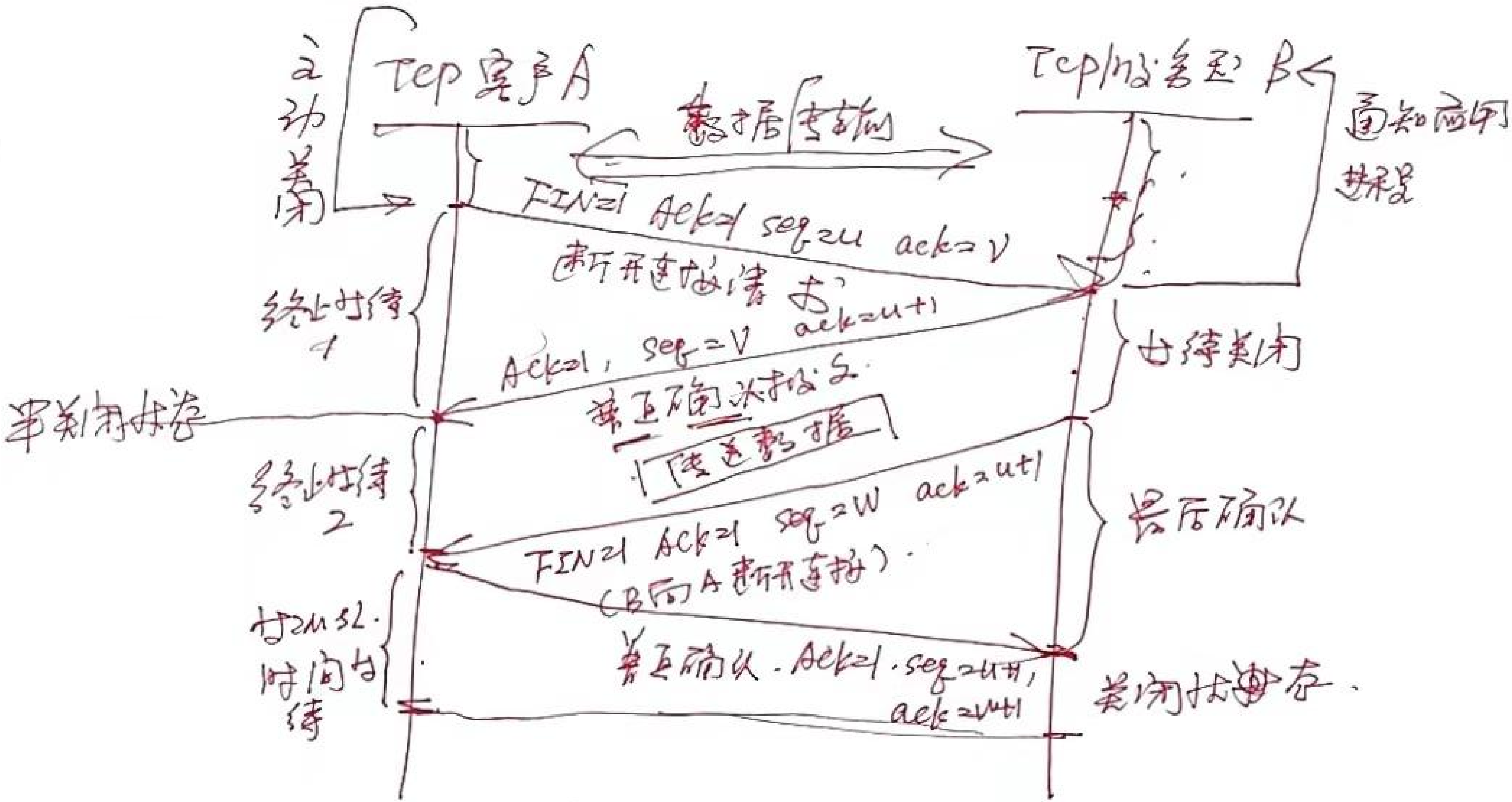
向A发送最后确认报文.

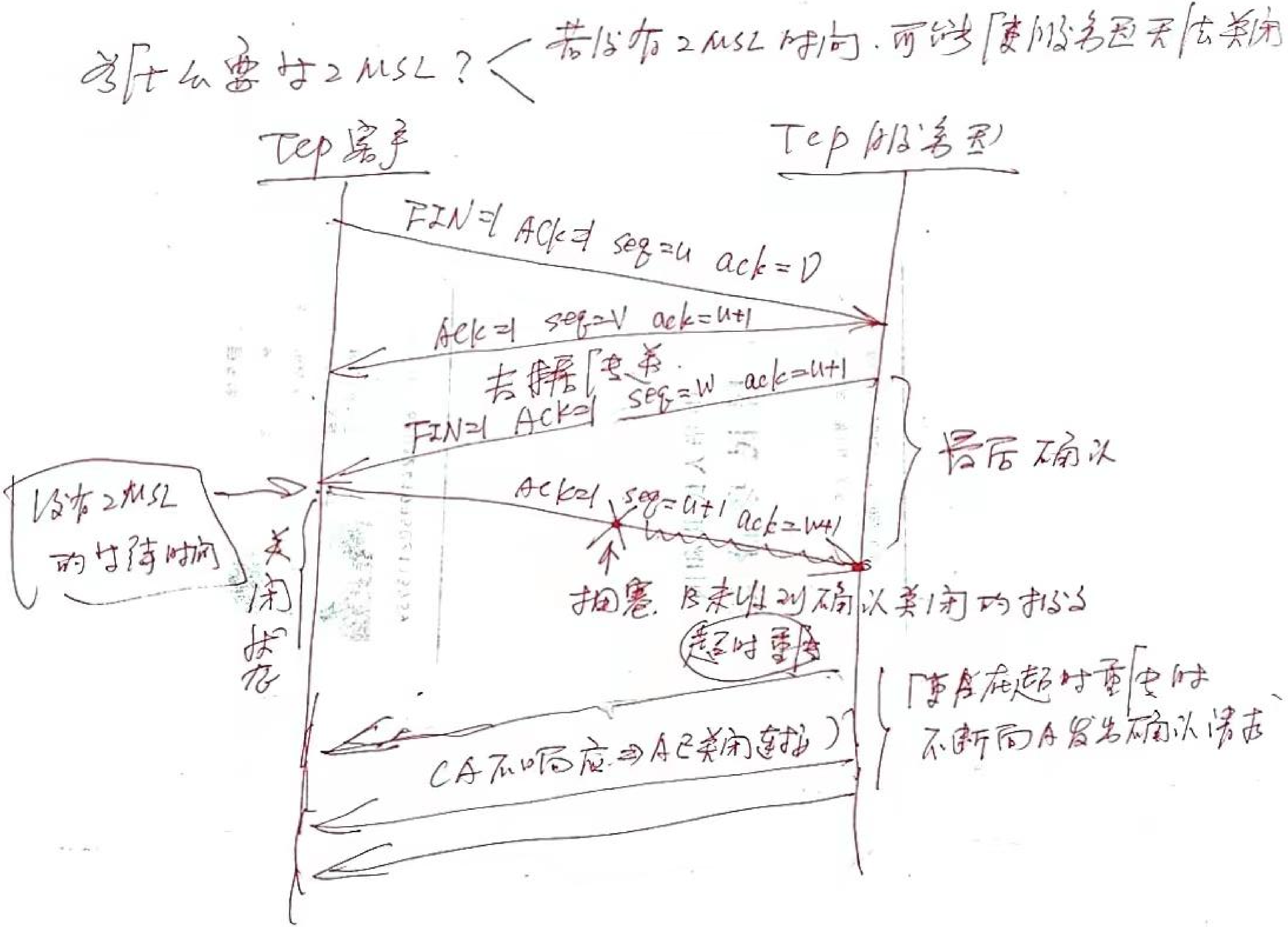
发送TCP连接释放

发送TCP连接确认

为什么要 2MSL?
可不可以不要?

最后确认报文


主动关闭
TCP客户A
TCP服务器B
数据传输
通知应用进程
FIN=1 ACK=1 seq=u ack=v
断开连接请求
终止等待1
ACK=1, seq=v ack=u+1
等待关闭
半关闭状态
传送数据
终止等待2
FIN=1 ACK=1 seq=w ack=u+1
(B向A断开连接)
最后确认
时间等待
ACK=1, seq=u+1, ack=w+1
关闭状态


为什么要等2MSL?
若没有2MSL时间，可能使服务器无法关闭
TCP客户
TCP服务器
FIN=1 ACK=1 seq=u ack=v
ACK=1 seq=v ack=u+1
FIN=1 ACK=1 seq=w ack=u+1
最后确认
没有2MSL的等待时间
ACK=1 seq=u+1 ack=w+1
关闭状态
阻塞，B未收到确认关闭的报文
超时重传
B会在超时重传时，不断向A发出确认请求
CA不响应⇒A已关闭连接

MSL 作用：（默认 2min）

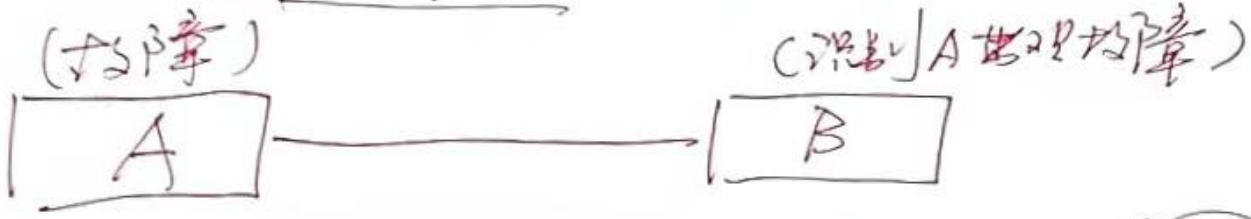
① 使服务器在发送的时间收到客户的关闭连接确认（TCP）

② 使数据阶段发送的报文在释放时间

☆ 为什么一定要4次挥手？

如何判断传送的2端方有一方出现故障？

—— 保活计时器（2h）



① 接收方每接收一次数据（一个报文 / 多个报文），就会启动保活计时器。

—— B每收到A的报文数据最大时间间隔：2h

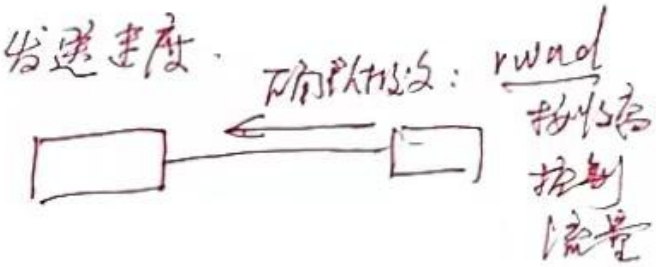
② 若保活计时器到时后，仍未收到发送方的下一个报文。

⇒ 接收方向发送方发送探测报文（每75发送一次）

☆ 若连续发送10次都未得到响应

可确认机器出现故障，服务器断开连接

TCP 流量控制 —— 接收方的目的是做力决定发送方的发送速度。



滑动窗口机制:在发送端控制流量，以便接收端正常接收

接收方根据自己接收缓存大小动态调整发送方窗口大小

1)接收窗口 rwnd 设置确认报文段的窗口字段，并通知发送方

2)发送方取接收窗口 rwnd 和拥塞窗口 cwnd 的较小值 min(rwnd, cwnd)

3)A 向 B 发送数据，连接建立时，B 告诉 A 自己的 rwnd 大小

4)TCP 为连接设置持续计时器，收到零窗口通知则启动

5)持续计时器到期则发送零窗口探测报文段，请求目前窗口值

6)若此时窗口值为 0，发送方重置持续计时器

窗口值为零.

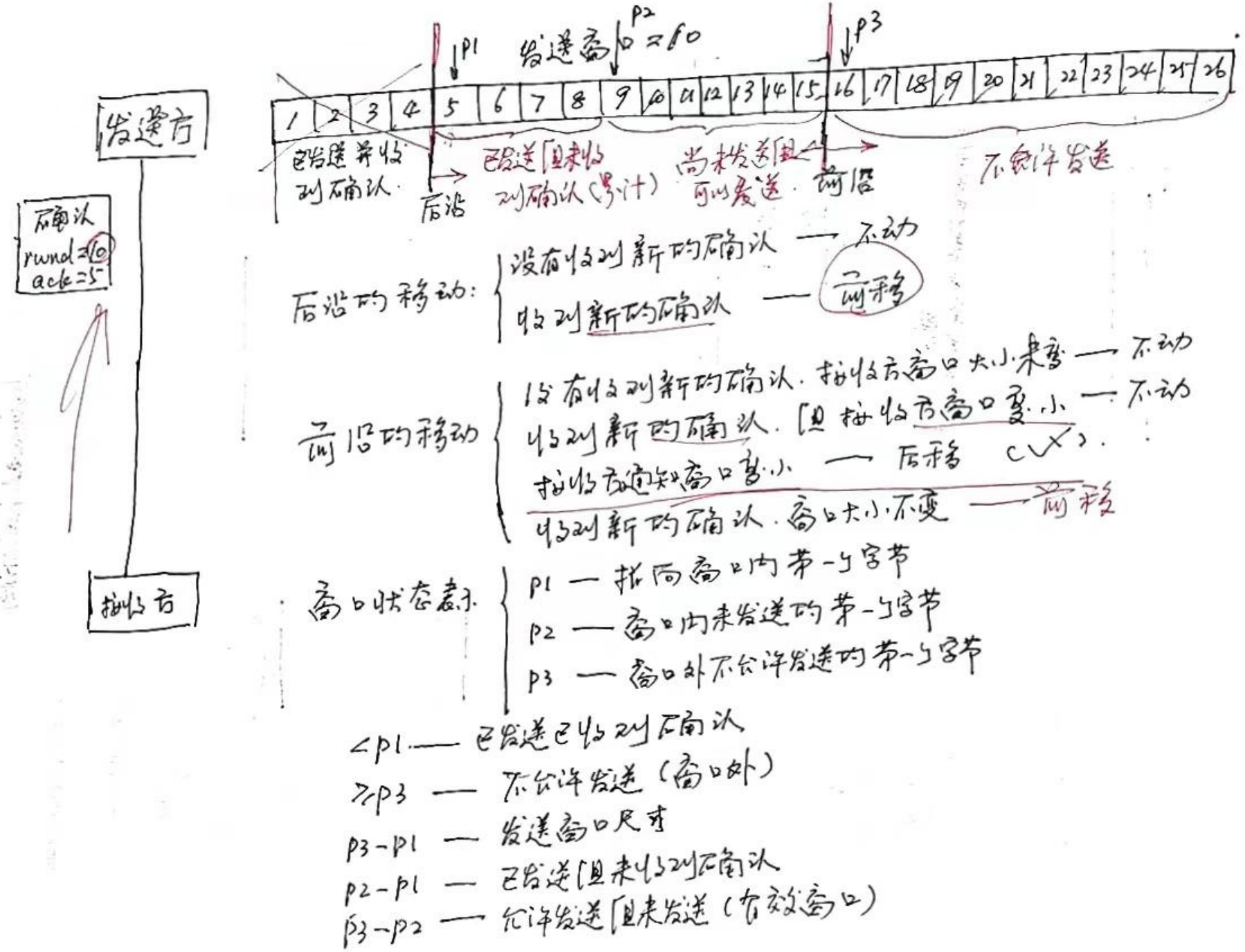
A> 每收到一次确认. 滑动窗口移动一次 —— 可以累计确认

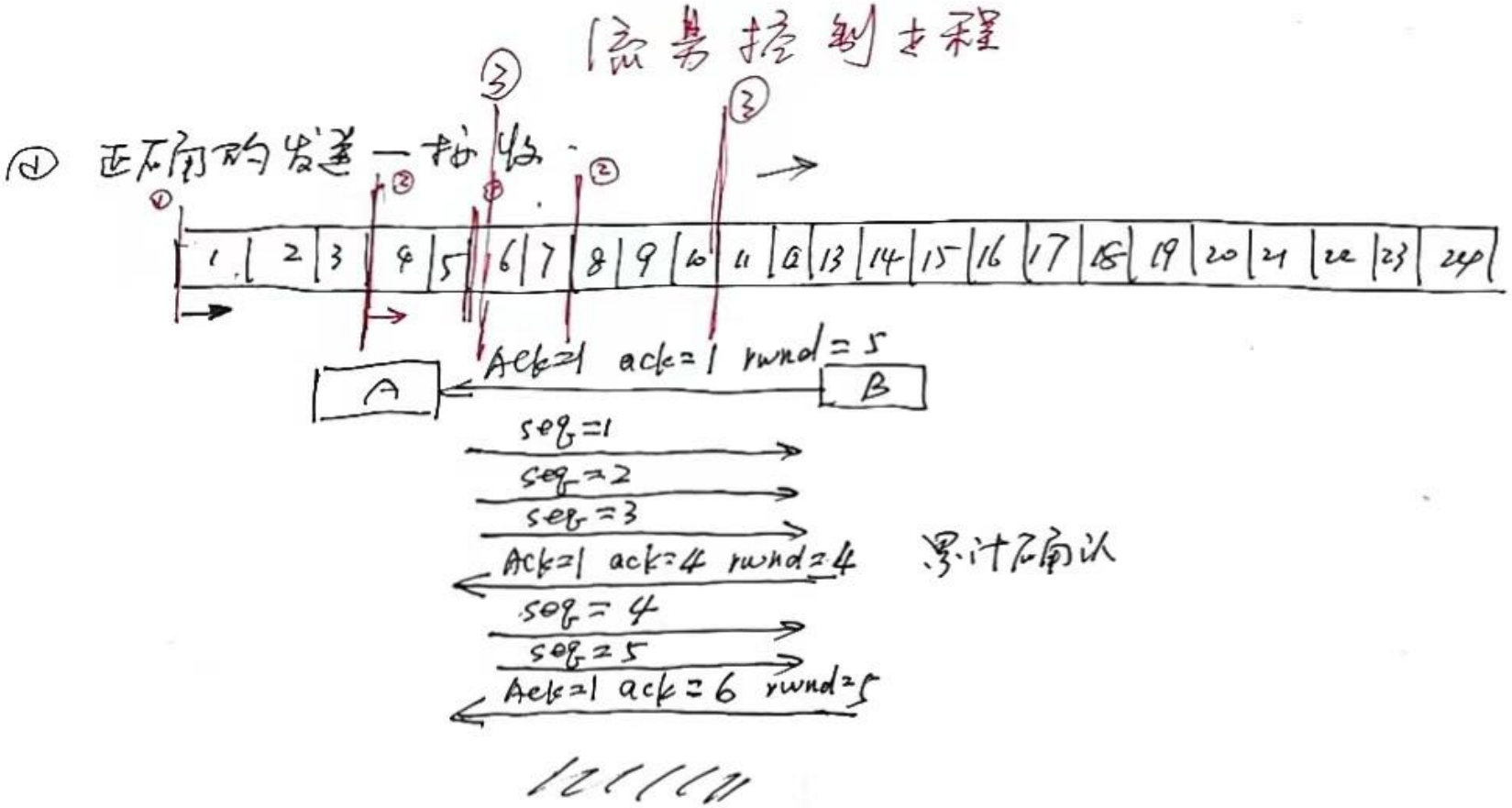
确认报文可以修改滑动窗口大小

B> 接收窗口为0 —— 启动持续计时器.

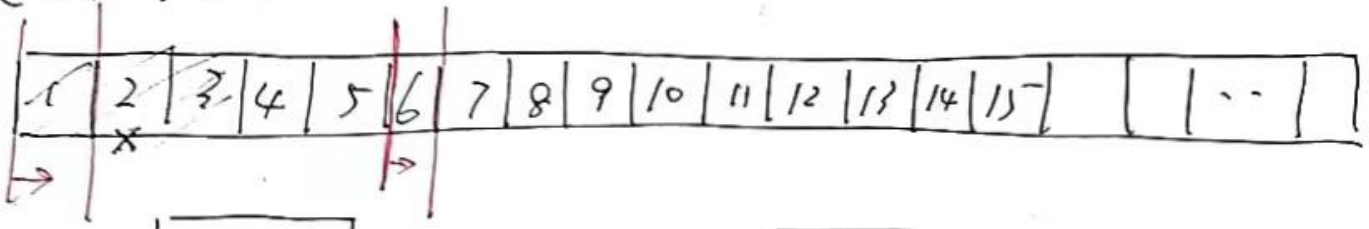
C> 持续计时器超时 —— 发送探测报文段.

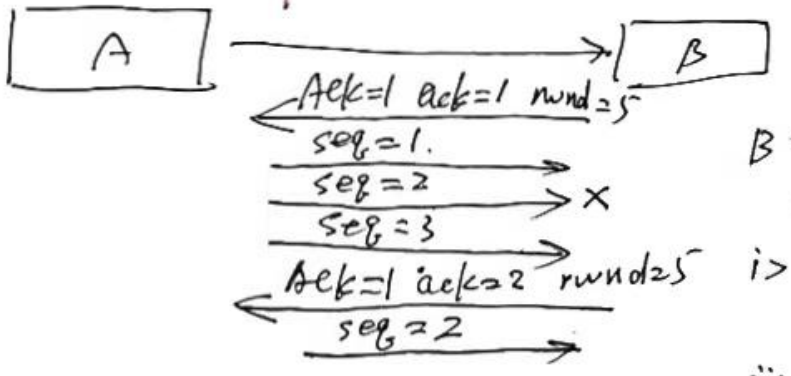
D> 探测报文段
确认报文段
有效数据报文段
} 接收窗口为0也必须相应.


发送方
确认
rwnd=10
ack=25
接收方
p1
p2
p3
发送窗口=10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
已发送并收到确认
后沿
已发送但未收到确认(累计)
尚未发送但可以发送
前沿
不允许发送
后沿的移动:
没有收到新的确认 — 不动
收到新的确认 — 前移
前沿的移动
没有收到新的确认,接收方窗口大小未变 — 不动
收到新的确认,但接收方窗口变小 — 不动
接收方通知窗口变小 — 后移 (×)
收到新的确认,窗口大小不变 — 前移
窗口状态表示
p1 — 指向窗口内第一个字节
p2 — 窗口内未发送的第一个字节
p3 — 窗口外不允许发送的第一个字节
<p1 — 已发送已收到确认
>=p3 — 不允许发送(窗口外)
p3-p1 — 发送窗口尺寸
p2-p1 — 已发送但未收到确认
p3-p2 — 允许发送但未发送(有效窗口)


流量控制过程
① 正确的发送一报文.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
A
B
Ack=1 ack=1 rwnd=5
seq=1
seq=2
seq=3
Ack=1 ack=4 rwnd=4
累计确认
seq=4
seq=5
Ack=1 ack=6 rwnd=5

② 传送过程中有缺失.

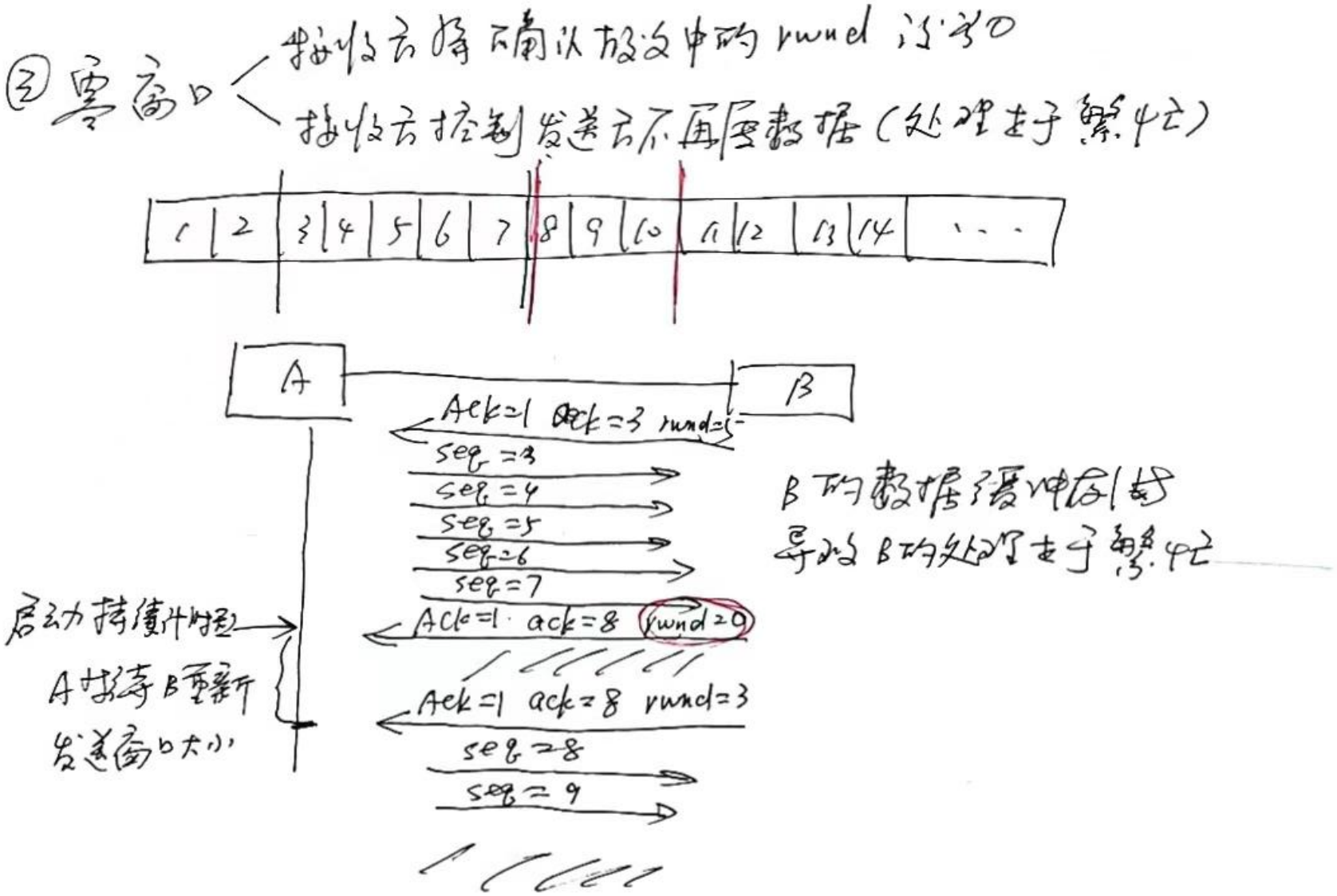




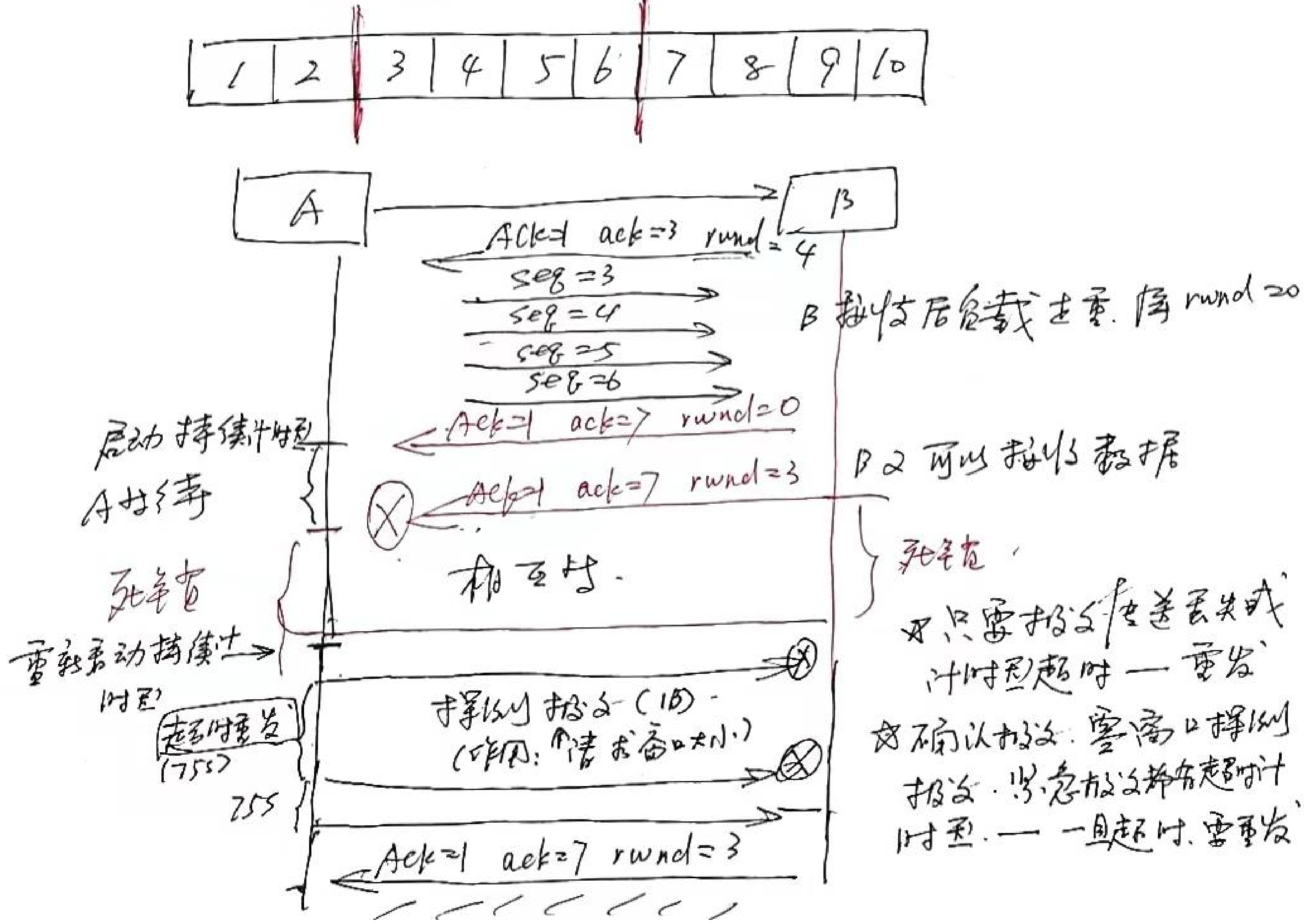
B收到①.③报文而未收到②等报文

i> 保留①.③等报文. 发送确认报文. ack=2

ii> 收到②等报文. 将①②③等报文连在一起


零窗口
接收方将确认报文中的 rwnd 设为0
接收方控制发送方不再发数据（处理进程繁忙）
1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
A
B
Ack=1 ack=3 rwnd=5
seq=3
seq=4
seq=5
seq=6
seq=7
B的数据缓冲区已满
导致B的处理进程繁忙
Ack=1 ack=8 rwnd=0
启动持续计时器
A将等B更新
发送窗口大小
Ack=1 ack=8 rwnd=3
seq=8
seq=9

④ 死锁 < 接收方等待发送方发数据 (零窗口)
发送方等待接收方发送窗口大小



TCP 的流量控制

1:一般来说，我们总是希望数据传输得更快一些。

但如果发送方把数据发送得过快，接收方就可能来不及接收，这就会造成数据的丢失。

2:所谓流量控制(flow control)就是让发送方的发送速率不要太快，要让接收方来得及接收。

3:利用滑动窗口机制可以很方便地在 TCP 连接上实现对发送方的流量控制。

1)TCP 接收方利用自己的接收窗口的大小来限制发送方发送窗口的大小。

2)TCP 发送方收到接收方的零窗口通知后,应启动持续计时器。持续计时器超时后,向接收方发送零窗口探测报文。