第 5 章作业——运输层：

1、端口的作用是什么？为什么端口要划分为三种？

2、TCP 如何实现端到端可靠性传输？

3、描述 TCP 连接建立的三握手过程和连接释放的四报文握手？

4、如何计算 RTTs、RTT_D、RTO？

5、第 8 版课后习题：5-33，5-34

6、在建立 TCP 连接时，发送方设定超时重传时间是 RTO =2.2s，且上一次测量计算的 RTTs 值为 1.4s。当发送方接到对方的连接确认报文段时，测量出 RTT 样本值为 1.5s，试计算现在的 RTO 值

7、在 TCP 的拥塞控制中，什么是慢开始、拥塞避免、快重传、快恢复算法？

8、拥塞控制和流量控制的作用和区别？

9、若采用滑动窗口机制对于两个相邻接点 A（发送方）和 B（接收方）的通信过程进行流量控制。假定帧的序号长度为 3 个二进制位，发送窗口和接受窗口的大小都是 7，当 A 发送了编号为 0、1、2、3 这 4 个帧后，而 B 接受了这 4 个帧，但仅应答了 0、1 两个帧。请问：此时，A 的发送窗口将要发送的帧序号为哪些？此时，B 的接收窗口内可能的最大帧序号为多少？

10、设 TCP 的 ssthresh 的初始值为 8(单位为报文段)。

当拥塞窗口上升到 12 时该网络发生了超时，TCP 使用慢开始和拥塞避免。

试分别求出第 1 次到第 15 次传输的各拥塞窗口大小并说明拥塞控制窗口每一次变化的原因。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cwnd															

11、设 TCP 的 ssthresh 的初始值为 8(单位为报文段)。

采用 Reno TCP 拥塞控制方法，当拥塞窗口上升到 12 时收到 3 个重复的确认，在第 13 次传输后发生了超时，试分别求出第 1 次到第 15 次传输的各拥塞窗口大小，给出不同算法的执行阶段，说明不同阶段的 ssthresh 门限值的大小。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cwnd															

12、通信信道带宽为 1Gb/s，端到端时延为 10ms。TCP 的发送窗口为 65535 字节。试问：网络可能达到的最大吞吐量是多少?信道的利用率是多少?

按以下三种情况分别进行计算:
（1）不考虑协议首部的封装，直接把数据发送到信道上
（2）考虑下层协议封装，但是不考虑数据链路层
（3）考虑所有的下层协议封装，数据链路层为 802.3MAC 帧

13、第 8 版课后习题：5-38，5-39，5-41，5-49

14、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 1 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 100 字节，则甲立即发送给乙的 TCP 段的序号和确认序号分别是多少?

第 5 章习题的其它题目请自行独立完成!!!

第 5 章作业答案与解析——运输层：

5、第 8 版课后习题：5-33

【解析】

根据 RFC 2988 建议，$RTO = RTTs + 4\times RTT_D$

初次测量时，$RTTs(1) = RTT(1)$

$RTT_D(1) = RTT(1)/2$

后续测量时，

$RTT_S(i) = (1-\alpha)\times RTT_S(i-1) + \alpha \times RTT(i)$　　$\alpha = 1/8$

$RTT_D(i) = (1-\beta)\times RTT_D(i-1) + \beta \times |RTTs(i) - RTT(i)|$　　$\beta = 1/4$

(1):

RTO=6 秒是人为初始设定的

测试出 RTT 样本值为 1.5 秒，比 6 秒少，没有超时，所以这个 RTT 样本值是可用的，可以用来计算新的 RTO，否则就得用修正算法了

$RTT(1) = 1.5$ s

$RTTs(1) = 1.5$ s

$RTT_D(1) = RTT(1)/2 = 0.75$ s

$RTO(1) = RTTs(1) + 4\times RTT_D(1)$

$= 1.5 + 4\times 0.75$

$= 4.5$ s

(2):

$RTTs(1) = 1.5$ s

$RTT_D(1) = 0.75$ s

$RTT(2) = 2.5$ s

$RTTs(2) = (1-\alpha)\times RTT_S(1) + \alpha \times RTT(2)$

$= (1 - 1/8)\times 1.5 + 1/8 \times 2.5$

$= 1.625$ s

$RTT_D(2) = (1-\beta)\times RTT_D(1) + \beta \times |RTTs(2) - RTT(2)|$

$= (1 - 1/4)\times 0.75 + 1/4 \times |1.625 - 2.5|$

$= 0.78125$ s

$RTO(2) = RTTs(2) + 4\times RTT_D(2)$

$= 1.625 + 4\times 0.78125 = 4.75$ s

5、第 8 版课后习题：5-34

【解析】

$\alpha = 0.1$

$RTT(1) = 30$ ms

$RTTs(1) = 30$ ms

$RTT(2) = 26$ ms

RTT(3) = 32 ms
RTT(4) = 24 ms
RTTs(2) = (1 – α) × RTT_S(1) + α × RTT(2)
= (1 – 0.1) ×30 + 0.1 ×26
= 29.6 ms
RTTs(3) = (1 – α) × RTT_S(2) + α × RTT(3)
= (1 – 0.1) ×29.6 + 0.1 ×32
= 29.84 ms
RTTs(4) = (1 – α) × RTT_S(3) + α × RTT(4)
= (1 – 0.1) ×29.84 + 0.1 ×24
= 29.256 ms

RTT 测量样本值的变化幅度可以超过 20%，但是加权平均后平滑的往返时间 RTTs 的变化幅度最多只有 0.1%

6、在建立 TCP 连接时，发送方设定超时重传时间是 RTO =2.2s，且上一次测量计算的 RTTs 值为 1.4s。当发送方接到对方的连接确认报文段时，测量出 RTT 样本值为 1.5s，试计算现在的 RTO 值

【解析】
根据 RFC 2988 建议，RTO = RTTs + 4×RTT_D
初次测量时，RTTs(1) = RTT(1)
RTT_D(1) = RTT(1) /2
后续测量时，
RTT_S (i) = (1 – α) × RTT_S(i-1) + α × RTT(i)　　α = 1/8
RTT_D(i)= (1- β) × RTT_D(i-1) + β × | RTTs (i)–RTT(i)|　　β = 1/4

RTO(1) = 2.2 s
RTTs(1) = 1.4 s
RTT(2) = 1.5 s
RTTs(2) = (1 – α) × RTT_S(1) + α × RTT(2)
= (1 – 1/8) ×1.4 + 1/8 ×1.5 = 1.4125 s
RTO(1) = RTTs (1) + 4×RTT_D(1)
= 1.4 + 4×RTT_D(1)
= 2.2 s
=> RTT_D(1) = 0.2 s
RTT_D(2)= (1- β) × RTT_D(1) + β × | RTTs(2) –RTT(2)|
= (1 – 1/4)×0.2 + 1/4 ×| 1.4125 – 1.5|
= 0.171875 s
RTO(2) = RTTs(2) + 4×RTT_D(2)
= 1.4125 + 4×0.171875

= 2.1 s

9、若采用滑动窗口机制对于两个相邻接点 A（发送方）和 B（接收方）的通信过程进行流量控制。假定帧的序号长度为 3 个二进制位，发送窗口和接受窗口的大小都是 7，当 A 发送了编号为 0、1、2、3 这 4 个帧后，而 B 接受了这 4 个帧，但仅应答了 0、1 两个帧。请问：此时，A 的发送窗口将要发送的帧序号是哪些？此时，B 的接收窗口内可能的最大帧序号为多少？

【解析】

帧序号长度为 3 位，则帧的序号为 0，1，...，7。

发送窗口大小为 7，则初始窗口内的帧序号为 0，1，2，3，4，5，6。窗口中的帧是在接收到 B 确认前，可以连续发送的帧序号。

A 的初始窗口：

0	1	2	3	4	5	6

A 连续发送序号为 0、1、2、3 的 4 个帧后的窗口状态如下：

0	1	2	3	4	5	6

B 收到了 4 个帧后，确认了 0、1 两个帧，则 A 的发送窗口将移出 0、1 两个帧序号，移入序号 7 和新的序号 0，即此时发送窗口的序号为 2、3、4、5、6、7、0（新）等。其中，序号为 2、3 的帧是已经发送并等待确认的，序号为 4、5、6、7、0 的帧是可以发送还没有发送的。

收到 0、1 帧确认后的 A 窗口：

0	1	2	3	4	5	6	7	0

此时，将要发送的帧分两种情况：

（1）如果未发生超时，4、5、6、7、0 是将要发送的帧。

（2）如果发生了超时，则 2、3 将会重传，那么将要发送的帧序号为 2、3、4、5、6、7、0

此时，3 是 A 已经发送出去的最大帧序号，因此，B 的接收窗口内可能的最大帧序号为 3（上一轮确认前可能发出的最大序号），但此后 B 的接收窗口内可能的最大帧序号为 7。

10、设 TCP 的 ssthresh 的初始值为 8(单位为报文段)。

当拥塞窗口上升到 12 时该网络发生了超时，TCP 使用慢开始和拥塞避免。

试分别求出第 1 次到第 15 次传输的各拥塞窗口大小并说明拥塞控制窗口每一次变化的原因。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cwnd	1	2	4	8	9	10	11	12	1	2	4	6	7	8	9

【解析】

- TCP 使用慢开始，第一次发送 1 段，窗口大小为 1；
- 随后按 2 的指数增长，增长到 ssthresh 的初始值 8，需要经过 $\log_2 8=3$ 次，即第 4 次；
- 随后进入第一轮拥塞避免算法，按线性增长到 12，需要 12-8=4 次，即第 8 次；
- 此时，发生超时将开始新一轮的慢开始，窗口重新设置为 1，同时新的 ssthresh 值更新为 12/2=6；
- 新一轮慢开始阶段由 1 按指数增长到大于 6，需要 3 次（$2^3=8>6$），即发生超时后的第 4 次，总第 8+4=12 次。
- 进入第二轮拥塞避免，窗口值由新的 ssthresh 值 6 开始线性增长，传输到第 15 次时，线性增长了 15-12=3 次，此时窗口值为 6+3=9。

11、设 TCP 的 ssthresh 的初始值为 8(单位为报文段)。

采用 Reno TCP 拥塞控制方法，当拥塞窗口上升到 12 时收到 3 个重复的确认，在第 13 次传输后发生了超时，试分别求出第 1 次到第 15 次传输的各拥塞窗口大小，给出不同算法的执行阶段，说明不同阶段的 ssthresh 门限值的大小。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cwnd	1	2	4	8	9	10	11	12	6	7	8	9	10	1	2

【解析】

- TCP 使用慢开始，第一次发送 1 段，窗口大小为 1；
- 随后按 2 的指数增长，增长到 ssthresh 的初始值 8，需要经过 $\log_2 8=3$ 次，即第 4 次；
- 随后进入第一轮拥塞避免算法，按线性增长到 12，需要 12-8=4 次，即第 8 次；
- 此时，收到 3 个重复的确认，窗口 cwnd 和 ssthresh 值更新为原来 cwnd 的一半，即 12/2=6，开始快重传和快恢复算法；
- 进入新一轮拥塞避免阶段，按线性增长到 10；
- 在第 13 次传输后发生超时，开始新一轮的慢启动，窗口重新设置为 1，同时新的 ssthresh 值更新为 10/2=5；
- 新一轮慢启动阶段，在第 15 次传输时由 1 按指数增长到 2。

12、通信信道带宽为 1Gb/s，端到端时延为 10ms。TCP 的发送窗口为 65535 字节。试问：网络可能达到的最大吞吐量是多少?信道的利用率是多少?

按以下三种情况分别进行计算：

（1）不考虑协议首部的封装，直接把数据发送到信道上
（2）考虑下层协议封装，但是不考虑数据链路层
（3）考虑所有的下层协议封装，数据链路层为 802.3MAC 帧

【解析】

（1）不考虑协议首部的封装，直接把数据发送到信道上

信道上发送的数据长度 L 为

$L = 65,535\ \text{Byte} = 65,535 \times 8\ \text{bit} = 524,280\ \text{bit}$

发送时间 $= 524,280\ \text{bit} \div 10^9\ \text{bit/s}$
$\approx 5.24 \times 10^{-4}\ \text{s}$
$= 0.524\ \text{ms}$

往返时延 $= 2 \times 10\ \text{ms} = 20\ \text{ms}$

- TCP 采用确认重传机制，成功的数据发送过程包括 TCP 报文段的发送和确认两个部分，缺一不可。
- 在连续 AQR 协议和累积确认机制下，在可能达到的最大吞吐量情况下，网络时延最小可以按照(发送时延+往返时延)来计算。

网络可达最大吞吐量=发送的数据 / 网络时延
= 发送的数据 /(发送时延+往返时延)
= 524,280 b/ (0.524+20) ms
= 0.524280 Mb / 20.524 ms
≈ 25.544 Mb/s

信道利用率 = 数据发送时延/占用信道时间
= 发送时延/ (发送时延+往返时延)
= 0.524/20.524
≈ 2.55%

或

信道利用率 = 吞吐量/信道带宽
= 25.544(Mb/s) / 1 (Gb/s)
≈ 2.55%

（2）考虑下层协议封装，但是不考虑数据链路层

数据长度 $L = (65535+20+20)\times 8 = 524600\ \text{bit}$
带宽 $C = 10^9\ \text{b/s}$
发送时间 $T_s = L/C = 0.0005246\ \text{s}$
端到端时延 $T_d = 10\times 10^{-3}\ \text{s}$
Throught $= L/(T_s + 2\times T_d)$
$= 524600/0.0205246$
$\approx 25.5\ \text{Mb/s}$
Efficiency $= T_s/(T_s + 2\times T_d)$
$\approx 2.55\%$

进一步讨论：

IP 数据报的最大总长度为 65535 字节

那么 TCP 报文段的数据部分最大为 65495 字节

当 TCP 的发送窗口为 65535 字节时，至少需要分成两个 TCP 报文段进行发送，那么在 IP 层上向信道发送的数据长度 L 应该增加两个 IP 首部和两个 TCP 首部，也就是

$$L=65535+2\times(20+20)=65615 \text{ 字节}$$

按照流水发送和累积确认方式，则下一步的计算方法同前。

（3）考虑所有的下层协议封装，数据链路层为 802.3MAC 帧

MAC 帧的最大长度为 6+6+2+1500+4 =1518 字节。

再加 7 个字节的前同步码，1 个字节的帧开始定界符，

信道上允许发送的数据长度 L 为 1526 字节。

如果再考虑曼切斯特编码会导致带宽效率下降一半，那吞吐量的估算就更繁琐了。

显然，如此复杂的情况与题目关注的网络吞吐量的初衷是不符的，可以不考虑。

13、第 8 版课后习题：5-38

【解析】

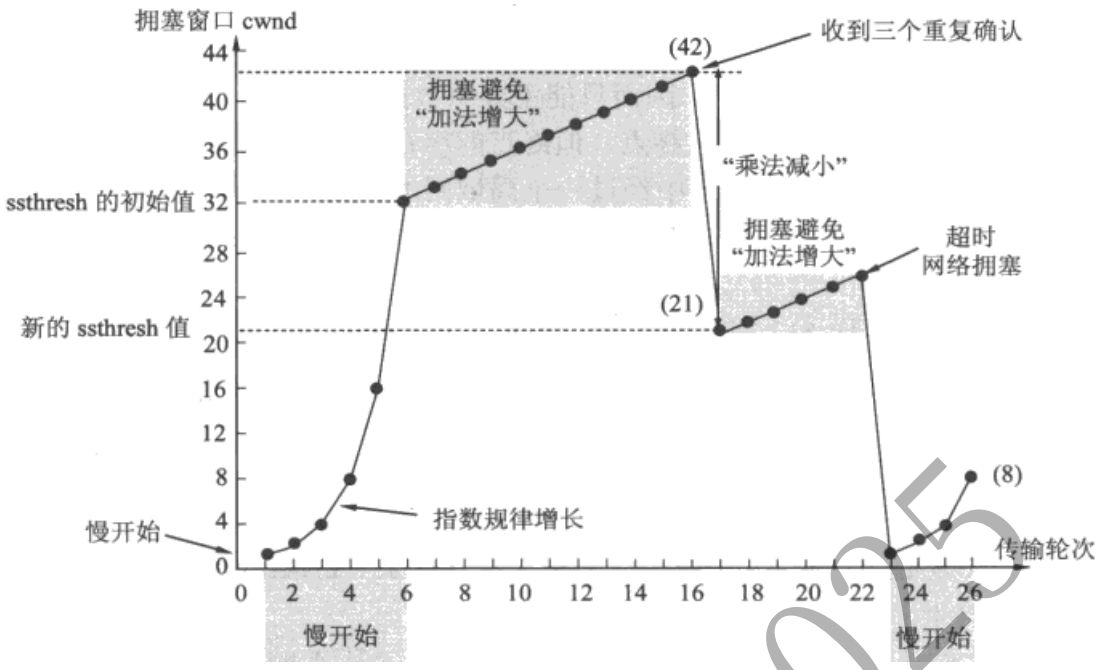
15 个轮次拥塞窗口大小及变化原因

轮次	拥塞窗口	拥塞窗口变化的原因
1	1	网络发生了超时，TCP 使用慢开始算法
2	2	拥塞窗口值加倍
3	4	拥塞窗口值加倍
4	8	拥塞窗口值加倍，这是 ssthresh 的初始值
5	9	TCP 使用拥塞避免算法，拥塞窗口值加 1
6	10	TCP 使用拥塞避免算法，拥塞窗口值加 1
7	11	TCP 使用拥塞避免算法，拥塞窗口值加 1
8	12	TCP 使用拥塞避免算法，拥塞窗口值加 1
9	1	网络发生了超时，TCP 使用慢开始算法
10	2	拥塞窗口值加倍
11	4	拥塞窗口值加倍
12	6	拥塞窗口值加倍，但到达 12 的一半时，改为拥塞避免算法
13	7	TCP 使用拥塞避免算法，拥塞窗口值加 1
14	8	TCP 使用拥塞避免算法，拥塞窗口值加 1
15	9	TCP 使用拥塞避免算法，拥塞窗口值加 1

13、第 8 版课后习题：5-39

【解析】

拥塞窗口与传输轮次的关系曲线



(2) 慢开始时间间隔：[1, 6]和[23, 26]。

(3) 拥塞避免时间间隔：[6, 16]和[17, 22]。

(4) 在第 16 轮次之后发送方通过收到三个重复的确认，检测到丢失了报文段，因为题目给出，下一个轮次的拥塞窗口减半了。

在第 22 轮次之后发送方是通过超时检测到丢失了报文段，因为题目给出，下一个轮次的拥塞窗口下降到 1 了。

(5) 在第 1 轮次发送时，门限 ssthresh 被设置为 32，因为从第 6 轮次起就进入了拥塞避免状态，拥塞窗口每个轮次加 1。

在第 18 轮次发送时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 42 的一半，即 21。

在第 24 轮次发送时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 26 的一半，即 13。

(6) 第 1 轮次发送报文段 1。（cwnd = 1）

第 2 轮次发送报文段 2, 3。（cwnd = 2）

第 3 轮次发送报文段 4 ~ 7。（cwnd = 4）

第 4 轮次发送报文段 8 ~ 15。（cwnd = 8）

第 5 轮次发送报文段 16 ~ 31。（cwnd = 16）

第 6 轮次发送报文段 32 ~ 63。（cwnd = 32）

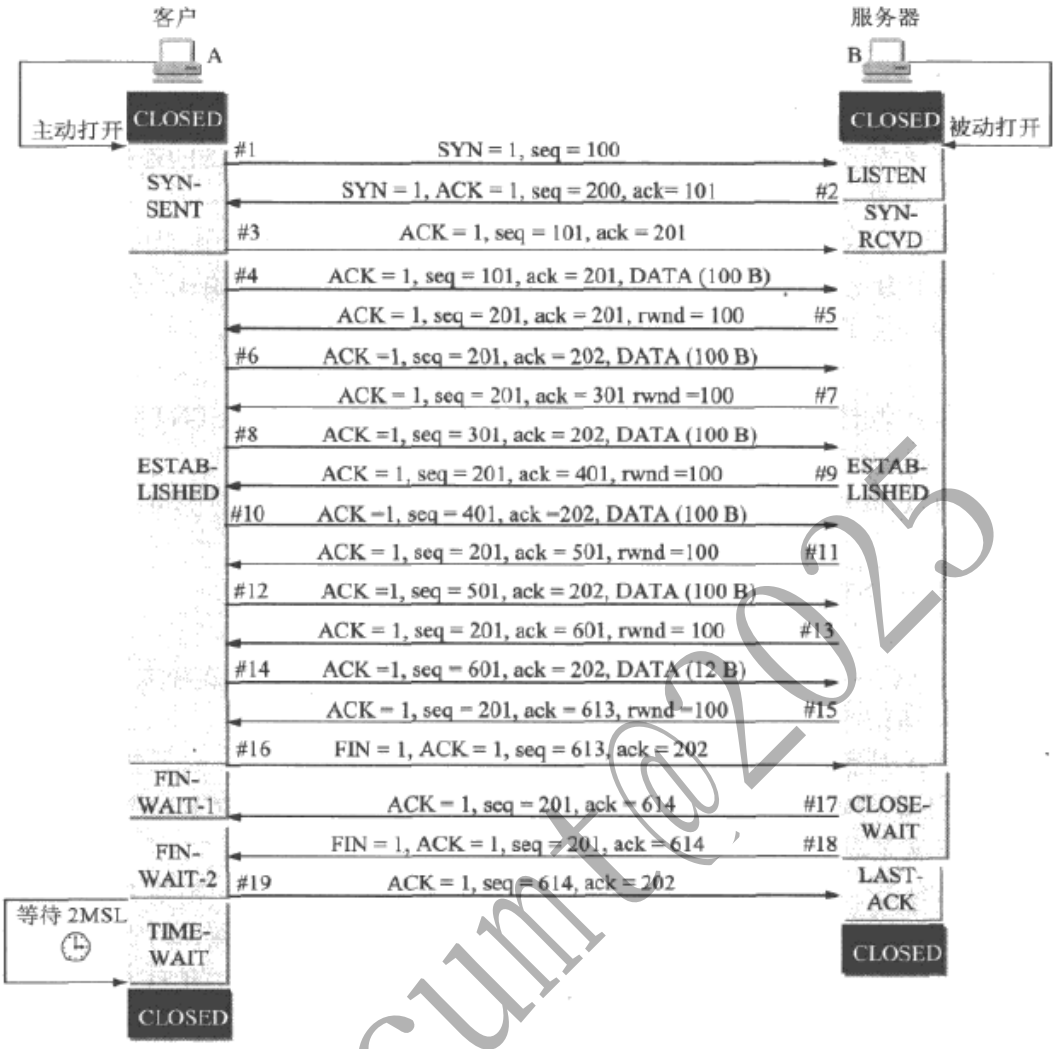
第 7 轮次发送报文段 64 ~ 94。（cwnd = 33）

因此第 70 报文段在第 7 轮次发送出。

(7) 检测出了报文段的丢失时拥塞窗口 cwnd 是 8，因此拥塞窗口 cwnd 的数值应当减半，等于 4，而门限 ssthresh 应设置为检测出报文段丢失时拥塞窗口 8 的一半，即 4。

13、第 8 版课后习题：5-41

【解析】



说明：

- ✓ 左边是客户端，右边是服务器
- ✓ 因为服务器端是独立确认，所以不消耗序号。
- ✓ 客户端对服务器端数据的捎带确认，所以确认序号也保持不变
- ✓ #4 的 ack=201，是客户端对 #2 的服务器端 seq=200 的确认，所以是 201
- ✓ 在三握手过程中，服务器会消耗一个序号，所以#5 中的服务器的 seq 会变成 201
- ✓ 因为，#5 的服务器端发送的是独立确认，不消耗序号
- ✓ 所以，#7 的服务器端的序号 seq 保持为 201
- ✓ #6 和#8 分别是对#5 和#7 的捎带确认
- ✓ 因为#5 和#7 的 seq 是 201，所以 #6 和#8 的 ack 就是 202
- ✓ 后续依次如此

14、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 1 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 100 字节，则甲立即发送给乙的 TCP 段的序号和确认

序号分别是多少？

【解析】

甲 ⇐ 乙

甲收到乙的 TCP 段： seq = 1913 ， ack = 2046， payload=100 字节

表明：

（1） 乙的 TCP 段的序号是 1913，数据长度为 100 字节，那么甲收到该报文段后则希望下次收到的报文段从 1913+100=2013 开始，即确认号 ack =2013

（2） 乙的 TCP 段的确认号是 2046，那么乙已经从甲收到 2046 之前的字节，希望下次收到的报文段从 2046 开始，即序号 seq = 2046

甲 ⇒ 乙

乙收到甲的 TCP 段：

seq = 2046 ， ack= 2013