## 简述多路复用和多路分解。

（1）多路复用(multiplexing at sender): 从上到下，从socket中收集数据，加报文头传递到网络层 (可以理解为把一个主机里的数个process的message 一起通过传输层发送出去)

（2）多路分解(demultiplexing at receiver): 自下而上，将传输层里发送给不同进程的报文分开，转交给正确的socket.

多路分解的实现需要:

1. IP报文里有源主机和目的主机的IP 2. 传输层报文里有源端口号和目的端口号 3. 通过 IP 和 port 就可以定位到主机的进程实现精确的分解。

除此之外，面向无连接的多路分解(UDP)只需要目的地IP和目的地端口号，而面向连接的堵路分解(TCP) 既需要目的地IP和目的地端口号，也需要发送方IP和发送方端口号。

## 计算下列两个16位字的校验和。

## 01111001 10111001            this binary number is 31161 decimal (base 10) 11101010 00001100            this binary number is 59916 decimal (base 10)

校验和的计算方法为: 把所有的位都划分成16位的字，然后一个字一个字进行累加 ，如果在某次加法有进位 就 把 1 加到后一位，然后接着累加，后所有字相加的结果取反

1 , 先把这两个16位的数相加 结果为： 0110 0011 1100 0101 ，最高位进位 1

2. 把最高位的进位再累加 得到结果 : 0110 0011 1100 0110

3 把最终结果取反 因此校验和为 : 1001 1100 0011 1001

## (P24) Answer true or false to the following questions and briefly justify your answer: a. With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window. ( true ) timeout 时间过短，导致再收到ack之前已经timeout了，重发，这时候上一次发送的ack到达，窗口移动。当第二次重发的包的ack到达时已经在窗口外了

## b. With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window. ( true ) timeout 时间过短，导致再收到ack之前已经timeout了，重发，这时候上一次发送的ack到达，窗口移动。当第二次重发的包的ack到达时已经在窗口外了 c. The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1.

## ( true ) 因为SR 返回的 ACK是每个包的ACK如果收到 pkt0 它返回 ack0，收到 pkt1 返回 ack1 d. The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1 ( true ) 因为GBN发送已经收到的包里序号最高的包，当窗口大小为1时，ACK就是 sender所期望的ACK

## 可靠传输有哪些策略？

可靠的传输可以采用的策略:

1. 加入错误检测机制(比如校验和)来检测是否有bit丢失现象
2. 在 1 的基础上加入应答机制(ACK 和 NAK)
3. 在2 的基础上发送方收到NAK就重传，也就是出错重传
4. 检测ACK包和 NAK包防止 应答包出错
5. 增加包序号，防止出现包冗余
6. 发送方增加计时器解决丢包问题

## 回退N步（GBN）和选择重传（SR）有什么相同和不同点？

相同点:

1. GBN 和 SR 的接收者和发送者都有窗口
2. GBN 和 SR 都属于流水线协议，可以在 rtt 中发送其它的包提升网络利用效率
3. GBN 和 SR的发送者都可可以最多有 N 个为确认的数据报在 pipline中

不同点:

1. GBN接收者发送累积 ACK(只发送按序收到的序号最高的序号)，而 SR接收者对每个数据报都发送单独的 ACK
2. GBN 接收者不缓存失序的报文，而SR接收者缓存失序的报文
3. GBN 为还没收到确认信息的序号最低的数据报设置定时器，而 SR的 sender为每个未确认报文设置定时器
4. GBN sender收到 ACK的时候，所有的序号<ACK的报文都变为已接受状态(ACK是累计ACK) 而 SR sender 只将对应的数据报设为已接受(只对单个数据报进行确认)。
5. 如果 timeout了 GBN重传 packet n 和窗口中索引顺序号高于n,已发送但是未确认的 pkt，而 SR只会发送 timeout 的那个数据报

## UDP和TCP报文头部有什么区别？为什么有这些区别？

区别: UDP的报文头只有 8 个字节，UDP的报文头只有源端口号和目的端口号，长度和校验和

而 TCP 的报文头有20个字节，除了源端口号和目的端口好，之外还有 序列号，ACK号，窗口大小，校验和等等附加信息。

产生这些区别的原因是:

UDP是无连接的简单传输协议，它提供基本的传输层协议，因此不需要这些附加信息。而TCP是面向连接的，稳定传输的传输层协议。为了提供稳定传输的服务，TCP需要 使用 应答机制(ACK)，给数据报添加序号，同时TCP也是一种流水线协议，自然也需要 窗口大小等附加信息。因为TCP和UDP提供的服务不同，TCP和UDP的报文头部也不同。

## Consider the rdt2.2 protocol from the text (pages 209-212). The sender and receiver FSMs for the sender and receiver are shown below:

## http://gaia.cs.umass.edu/kurose_ross/interactive/fig2.13.jpg

## http://gaia.cs.umass.edu/kurose_ross/interactive/fig2.14.jpg

## Suppose that the channel connecting the sender and receiver can corrupt but not lose or reorder packets. Now consider the figure below, which shows four data packets and three corresponding ACKs being exchanged between an rdt 2.2 sender and receiver. The actual corruption or successful transmission/reception of a packet is indicated by the corrupt and OK labels, respectively, shown above the packets in the figure below.

## network

## Fill out the table below indicating *(i)*the state of the sender and the receiver just *after*the the transmission of a new packet in response to the received packet at time *t*, *(ii)*the sequence number associated with the data packet or the ACK number associated with the ACK packet sent at time *t.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t | sender state | receiver state | packet type sent | seq. # or ACK # sent |
| 0 | Wait ACK0 | Wait 0 from below | data | 0 |
| 1 | Wait ACK0 | Wait 0 from below | ACK | 1 |
| 2 | Wait ACK0 | Wait 0 from below | data | 0 |
| 3 | Wait ACK0 | Wait 1 from below | ACK | 0 |
| 4 | Wait ACK1 | Wait 1 from below | data | 1 |
| 5 | Wait ACK1 | Wait 0 from below | ACK | 1 |
| 6 | Wait ACK0 | Wait 0 from below | data | 0 |

## How many times is the payload of the received packet passed up to the higher layer at the receiver in the above example? At what times is the payload data passed up?

payload 被提交 2 次。在 t =3 和 t =5 被提交。

## (P27)Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A. a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

## Sequence number 是 128， source port是 302， destination port 是 80 b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?

## Acknowledgement number : 128 , source port 80, destination port :302 c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

## Acknowledgement number 是 127 (第二个报先到达了，失序了，接收方会先缓存下来，但是发送期望的报文序号,此时它希望的是 127)

## d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number。

