## **第三章作业**

09019204 曹邹颖

## 简述多路复用和多路分解。

答：**多路复用**是在源主机从不同套接字中收集数据块，并为每个数据块封装上首部信息从而生成报文段，然后将报文段传递到网络层的这些工作。**多路分解**是将传输层报文段中的数据交付到正确的套接字的工作。

标答：从下到上，根据报文段中的端口号将传输层报文段中的数据分发到接受主机正确的套接字的工作称为多路分解。

从上到下，从源主机的不同套接字中收集数据块，并为某个数据块封装上首部信息从而生成报文段，然后将报文段传递到网络层的工作称为多路复用。

## 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)

答：，和有溢位，进行回卷，，

所以校验和为。❌ 错的，要取反码！！！

应该是1001110000111001

## (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. ( T )

答：当发送方发送包i时，接收方成功接收到并响应发送ACKi，但ACKi在途中延时，发送方定时器显示超时并重新发送包i，发送方又重新接收到但删除冗余并重新发送ACKi，发送方成功接收到后来重新发的ACKi，并且在包i之前不存在接收但未确认的包从而窗口后移，此时序号i已不在窗口中，但之后之前延时的ACKi被发送方接收到，即发送方接收到了一个序号不在当前窗口内的ACK，发送方不必做出任何动作。

## b. With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window. ( T )

## 答：同上。

## c. The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1.

## ( T ) 注：比特交换协议又称rdt3.0

答：当SR/GBN协议的窗口大小只有1时，便从流水线可靠数据传输协议变为停等协议，SR和GBN协议不再有区别，而SR/GBN协议与rdt3.0唯一的区别就是不以停等方式运行，允许发送方发送多个分组无须确认，窗口为1便和rdt3.0功能一样了。

## d. The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1 ( T )

## 答：同上。

## 可靠传输有哪些策略？

答：检验和、重传、序号、确认、滑动窗口（流量控制）和定时器。

**检验和：**用于确认包是否出错。

**重传：**用于包出错、丢失后的恢复。

**序号：**发送方对其数据分组编号，可供删除冗余与处理包失序。

**确认：**回退N步和选择重传，当接收到正确的分组时接收方对所接收的分组发送一个肯定确认，用于处理包失序。

**滑动窗口：**窗口即为缓存，提高网络性能，允许发送方发送多个分组无须确认，且用于流量控制。

**定时器：**处理包丢失，诱发重传机制。

标答：1. 校验和检测比特位错误

1. 接收者反馈ACK，NAK
2. 超时重传
3. 数据报添加序号
4. 定时器timer

（建议进行适当解释）

## 回退N步（GBN）和选择重传（SR）有什么相同和不同点？

答：回退N步（GBN）和选择重传（SR）的相同点：

1. 发送方都有一个窗口——缓存；
2. 都采用定时器，处理包丢失的情况；
3. 都增加了序号的范围，应对多个在输送中的未确认报文；

回退N步（GBN）和选择重传（SR）的不同点：

1. 回退N步（GBN）的接收方不存在缓存，对于乱序到来的包采用丢弃的处理方法，而选择重传（SR）的接收方有一个窗口——缓存，对于乱序的包接收并缓存下来；
2. 回退N步（GBN）采用累积确认的方式，发送方timeout(n)重传包n和窗口中序号大于n、已发送但未确认的包，而选择重传（SR）的接收方可单独确认所有正确接收的数据包，发送方超时只重传未接收到ACK的数据包。

标答: 相同点是都使用了流水线的滑动窗口机制，使用序号标明数据报，以及超时重传，都采用了校验和。

不同点是GBN采用了累积确认，以及定时器只对最老的一个未被确认的数据报进行计时，如果接收者收到的序号不连续，就会丢弃该报文，不返回确认报文。一旦超时，就会对窗口中的N个包一起重发。而SR采用了单个确认的方式，每一个发送的包都会有一个定时器。接收端对单个数据包进行确认。超时只会重传未被确认的包。

## UDP和TCP报文头部有什么区别？为什么有这些区别？

答：

UDP报文头部有一个16bits的长度字段，是TCP报文头部没有的，它表示在UDP报文段中的字节数（首部加数据），因为数据字段的长度在一个UDP段中不同于在另一个段中，故需要一个明确的长度。

而TCP报文头部比UDP报文头部多了32bits的序号字段、32bits的确认号字段、16bits的接收窗口字段、4bits的首部长度字段，可选与变长的选项字段以及6bits的标志字段。

32bits的**序号**字段和32bits的**确认号**字段：用于提供TCP可靠传输服务；

6bits的**接收窗口**字段：用于流量控制；

4bits的**首部长度**字段：由于TCP选项字段的原因，TCP的首部长度是可变的，所以用于指示TCP的首部长度；

可选与变长的**选项**字段：用于发送方与接收方协商最大报文长度（MSS）时，或在高速网络环境下用作窗口调节因子时使用；

6bits的标志字段：URG、ACK、PSH、RST、SYN、FIN，共6个：

URG用于表示报文段中存在着被发送端的上层实体置为“紧急”的数据；

ACK为确认序号标志，=1时表示确认号有效，=0表示报文中不含确认信息；

PSH表示接收方在接收到该报文段后应尽快将其交给应用程序而不是在缓冲区中；

RST用于重置由于主机崩溃或其他原因而出现错误的连接，或者用于拒绝非法的报文段和拒绝连接请求；

SYN用于建立连接过程，在连接请求中，SYN=1和ACK=0表示该数据段没有使用捎带的确认域，而连接应答捎带一个确认，即SYN=1和ACK=1；

FIN用于释放连接，=1时表示发送方已经没有数据发送了，即关闭本方数据流。

标答：UDP报文头部长度是固定的八个字节，包括源端口，目的端口，报文长度以及校验和。而TCP报文头部长度是可变的，头部字段以32位为单位给出头部大小。包括源端口，目的端口，校验和，序列号（此次发送的数据在整个报文段中的起始字节数），确认号（下一个期望收到的字节，只有当ack标志为1时有效），6位标志位，窗口大小，紧急指针，选项字段。产生这些区别的原因是UDP是一种不可靠的，无连接的传输，报文头需要的信息很少。而TCP是一种可靠的，面向连接的字节流服务，为了保证连接以及数据传输的可靠性，增加了很多必要的字段。

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

## 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 for ACK0 | Wait 0 from below | data | 0 |
| 1 | Wait for ACK0 | Wait 0 from below | ACK | 1 |
| 2 | Wait for ACK0 | Wait 1 from below | data | 0 |
| 3 | Wait for call 1 from above ❌ | Wait 1 from below | ACK | 0 |
| 4 | Wait for ACK1 | Wait 0 from below❌ | data | 1 |
| 5 | Wait for call 0 from above ❌ | Wait 0 from below | ACK | 1 |
| 6 | Wait for ACK0 | Wait 1 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?

答：3次交付给应用层，at t=3,5,7。❌

标答：2 packets were passed up to the higer layer at the receiver, at times t = 3, 5.

## （P27）主机A和B经一条TCP连接通信，并且主机B已经收到了来自A的最长为126字节的所有字节。假定主机A随后向主机B发送两个紧接着的报文段，第一个和第二个报文段分别包含了80字节和40字节的数据。在第一个报文段中，序号是127，源端口号是302，目的地端口号是80。无论何时主机B接受到来自主机A的报文段，它都会确认发送。

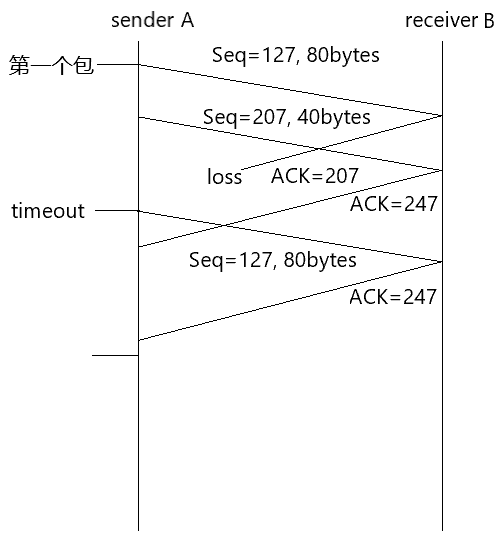
1. 在从主机A发往B的第二个报文段中，序号、源端口号和目的端口号各是什么？
2. 如果第一个报文段在第二个报文段之前到达，在第一个到达报文段的确认中，确认号、源端口号和目的端口号各是什么？
3. 如果第二个报文段在第一个报文段之前到达，在第一个到达报文段的确认中，确认号是什么？
4. 假定由A发送的两个报文段按序到达B。第一个确认丢失了而第二个确认在第一个超时间隔之后到达。画出时序图，显示这些报文段和发送的所有其他报文段和确认。（假设没有其他分组丢失。）对于图上每个报文段，标出序号和数据的字节数量；对于你增加的每个应答，标出确认号。

答：a. 序号：207，源端口号：302，目的端口号：80；**√**

b. 确认号：207，源端口号：80，目的端口号：302；**√**

c. 确认号：127，因为TCP只确认该流中至第一个丢失字节为止的字节（累计确认）；**√**

d.



## 流量控制和拥塞控制有什么区别？TCP的拥塞控制是如何实现的？

答：流量控制是一个速度匹配服务，要求发送方的发送速率与应用程序的读取速率相匹配，以消除发送方使接收方缓存溢出的可能性，而拥塞控制是发送方感知IP网络的拥塞程度来限制其向连接发送流量的速率以防止发送方因为IP网络的拥塞而被遏制。

TCP的拥塞控制实现方式：

1. AIMD（加性增、乘性减）：每个RTT周期增加cwnd（拥塞窗口）by 1 MSS，直到检测到报文段丢失，若检测到报文丢失则将cwnd减少至之前的一半大小；

2. **慢启动**：cwnd的值以1个MSS开始，每当传输的报文段首次被确认就增加一个MSS，从而每过一个RTT，发送速率就翻番，以指数速度增加。结束慢启动的几种情况：①一个报文段丢失事件发生即存在拥塞（超时），cwnd置1，ssthresh设置为cwnd/2，重新开始慢启动过程。②当cwnd超过ssthresh时，结束慢启动并且TCP转移到**拥塞避免**模式，此时TCP更为谨慎地增加cwnd例如线性增长。③如果检测到3个冗余ACK，这时TCP执行快速重传并进入**快速恢复**状态。

3. **拥塞避免：**此时每个RTT只将cwnd的值增加一个MSS。当出现超时，处理方式同慢启动，进入**慢启动**状态；当遇到3个冗余ACK时，ssthresh变为cwnd的值的一半，cwnd的值减半再+3，之后进入**快速恢复**状态。

4. **快速恢复：**此时若继续收到冗余ACK，cwnd的值增加一个MSS，保持在快速恢复状态；当收到新的ACK，cwnd降低至ssthresh的值，进入**拥塞避免**状态；当出现超时，处理方式同慢启动，进入**慢启动**状态。

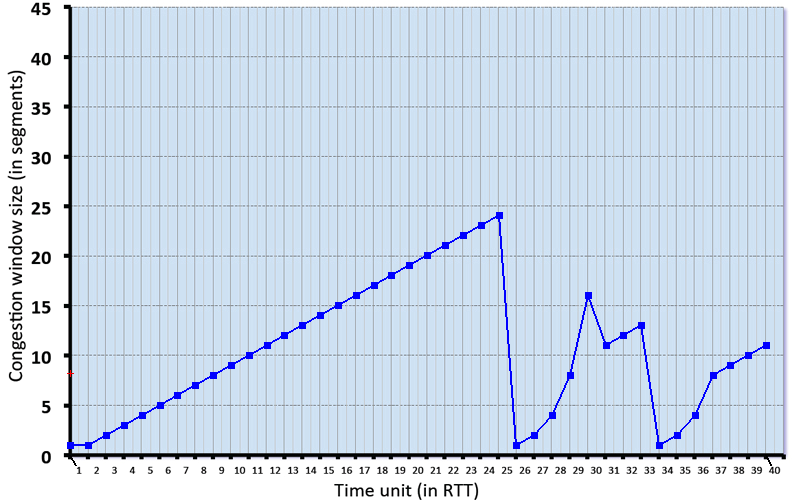
标答：流量控制是为了解决发送方和接收方速度不匹配而导致的数据丢失问题；拥塞控制是为了处理网络出现拥塞的现象，例如丢包，长延迟等，在出现拥塞现象时遏制发送方，以减少分组的丢失。

TCP拥塞控制：

使用控制变量—拥塞窗口来表征网络的拥塞情况，对发送方能发送的速率进行限制。TCP发送方通过丢包事件感知它在与目的地之间路径上出现的拥塞。TCP拥塞算法包括三个主要部分：加性增、乘性减，慢启动，对超时时间的反应。算法对拥塞窗口的调整是以最大报文段长度字节为单位进行的。TCP的拥塞控制包括三种状态：慢启动状态、拥塞避免状态和快速恢复状态。TCP 拥塞控制的实现 :

1. TCP 拥塞控制采用端到端的控制，即没有网络层的反馈，而是通过当前网络状况进行动态控制(timeout, 冗余包的数目)
2. TCP 中有一个参数 cwnd 用来控制所有未收到回应的包的数目。
3. 当没有发生 timeout 事件以及没有发生 收到3次冗余包的时候，每一次收到 Ack, TCP就逐渐增大cwnd 的值，提升发送速率 (先指数上升，到达某个阈值的时候再线性上升)。
4. 当发生loss event的时候(time out 或者连续收到 3 个冗余包) TCP就减少 cwnd的值来降低 TCP的发送速率。

10. Consider the figure below, which plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT); see Figure 3.53 in the text. In the abstract model for this problem, TCP sends a "flight" of packets of size cwnd at the beginning of each time unit. The result of sending that flight of packets is that either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. In this problem, you are asked to reconstruct the sequence of events (ACKs, losses) that resulted in the evolution of TCP's cwnd shown below.



Consider the evolution of TCP's congestion window in the example above and answer the following questions. The initial value of cwnd is 1 and the initial value of ssthresh (shown as a red +) is 8.

### QUESTION LIST

1. Give the times at which TCP is in slow start. Format your answer like: 1,3,5,9 (If none submit blank)

答：1，2，26，27，28，29，34，35，36.（∵慢启动以一个MSS开始并以指数形式增长）**√**

1. Give the times at which TCP is in congestion avoidance. Format your answer like: 1,3,5,9 (If none submit blank)

答：3，4，5，6，7，8，9，10，11，12，13，14，15，16，17，18，19，20，21，22，23，24，25，30，37，38，39，40.（拥塞避免）**√**

1. Give the times at which TCP is in fast recovery. Format your answer like: 1,3,5,9 (If none submit blank)

答：31，32，33.（快速恢复）**√**

1. Give the times at which packets are lost via timeout. Format your answer like: 1,3,5,9 (If none submit blank)

答：1，25，33.（超时）**√**

1. Give the times at which packets are lost via triple ACK. Format your answer like: 1,3,5,9 (If none submit blank)

答：30.（3个重复的ACK）**√**

1. Give the times at which the value of ssthresh changes (if it changes between t=3 and t=4, use t=4 in your answer)

答：2，26，31，34.（ssthresh改变为cwnd的一半）**√**

**标答：**The complete solution is shown in the figure below:

For intervals of time when TCP is in slow start, the plotted value of cwnd is shown as a green square

For intervals of time when TCP is in congestion avoidance, the plotted value of cwnd is shown as a yellow square

For intervals of time when TCP is in fast reccovery, the plotted value of cwnd is shown as an orange square

The values for ssthresh are shown following a change as a red plus sign

A flight of packets experiencing a loss has the loss type (which determines the next value of cwnd) labeled above

