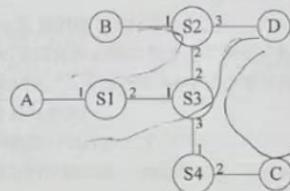


## 二、接口层（10分）

下图中 S1-S4 是以太交换机，所有交换机的地址映射表初始为空。A、B、C、D 为以太网节点，MAC 地址分别表示为 MAC<sub>A</sub>、MAC<sub>B</sub>、MAC<sub>C</sub>、MAC<sub>D</sub>。如果首先 A 发送数据帧到 B，之后 D 发送数据帧到 A，接着 C 发送数据帧到 D，经历这三次数据帧传输后，S1-S4 的地址映射表应记录哪些信息，请填写相应的地址映射表。



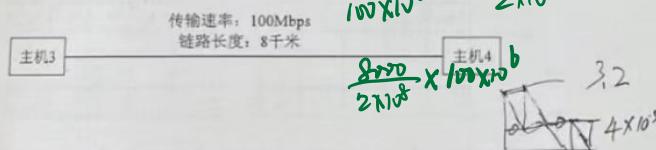
### 一、简答题（20分，每小题 10 分）

- (1) 如图 1 所示，主机 1 与主机 2 之间通过两条链路和一台存储转发设备 R 进行连接，主机 1 向主机 2 发送长度为 2000 字节的报文，采用报文交换，请计算端到端的最小时延。如图 2 所示，主机 3 与主机 4 之间通过一条链路直接连接，计算这条链路的时延带宽积。（注：要求给出计算过程，计算中设电磁波传播速度为  $2 \times 10^8$  米/秒）

图 1



图 2



- 2) 发送端要发送数据的位模式为 1100 1101，采用 CRC 校验，生成多项式 G 为 1101，如果接收端接收到的位模式为 11001111110，接收端是否报告差错？请给出计算过程。如果使用 4 位的校验和，校验和放在数据后面传输，给出发送端实际发送的位模式。

$$\begin{array}{r} 1100 \\ \text{-----} \\ 1101 | 11001101000 \\ \text{-----} \\ 1101 \quad 1101 \\ \text{-----} \\ 1111110 \\ \text{-----} \\ 1101 \quad 1101 \\ \text{-----} \\ 1011110 \\ \text{-----} \\ 1101 \quad 1101 \\ \text{-----} \\ 11001101000 \end{array}$$

$$\begin{array}{r} 11001111110 \\ \text{-----} \\ 1101 | 11001111110 \\ \text{-----} \\ 1101 \quad 1101 \\ \text{-----} \\ 11001111110 \\ \text{-----} \\ 01011101 \\ \text{-----} \\ 11110 \\ \text{-----} \\ 01011101 \\ \text{-----} \\ 11001111110 \end{array}$$

$$\begin{array}{r} 11001111110 \\ \text{-----} \\ 1101 | 11001111110 \\ \text{-----} \\ 1101 \quad 1101 \\ \text{-----} \\ 0001 \\ \text{-----} \\ 1101 \\ \text{-----} \\ 11001111110 \end{array}$$

MAC 地址	接口
MAC <sub>A</sub>	1
MAC <sub>B</sub>	2
MAC <sub>D</sub>	2

MAC 地址	接口
A	2
B	3
C	3

MAC 地址	接口
A	1
B	2
C	3

MAC 地址	接口
A	1
B	2

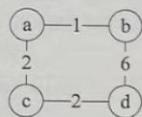
$$77 = 64 + 8 + 4 + 2 + 1$$

### 三、网络层 (20 分)

$72 = \frac{64+8}{6 \quad 3} \quad 0|00\ 1000 \cup 0100\ 1101$

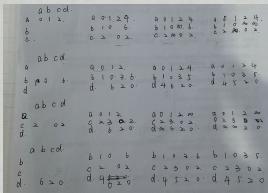
- 1) (6分) 基于CIDR机制,对202.113.72.0~202.113.77.0六个C类的IP网络地址进行~~段落~~聚合,写出聚合后的前缀和掩码(用十进制点分割形式表示)。

- 2) (6分) 有如下图所示的网络结构, a、b、c、d 为路由器, 路由器之间的链路代价已在图中标出, 例如  $c(a, b)=1$ , 使用毒性逆转的距离向量算法, 写出收敛后 a 和 c 保存的距离向量表。



a 节点的距离向量表

	a	b	c	d
a	1	1	2	4
b	1	0	40	20
c	2	0	0	2



### c 节点的距离向量表

	a	b	c	d
a	0	1	>	2
c	2	3	>	2
d	2	3	2	0

- (3) (8分) 路由器有如下路由表, 采用 CIDR 路由机制, 如果接收到目的 IP 地址为 202.113.182.8 和 202.113.191.10 的 IP 数据包, 分别给出下一跳的 IP 地址 (要求计算过程)。

前缀	网络掩码	下一跳 IP 地址
202.113.184.0	255.255.248.0 <u>11111100</u>	182.10.200.1
202.113.190.0	255.255.254.0 <u>11111110</u>	182.11.200.1
0.0.0.0	0.0.0.0	182.12.200.1

$$\begin{array}{r} 182 = 1011\ 0110 \\ 181 = 1011\ 1111 \end{array}$$

$$\begin{array}{r} 1011 \quad 0110 \\ 111 \quad 1000 \\ \hline 1011 \quad 0000 \\ 176. \end{array}$$

$$\begin{array}{r} 1011 \quad 0110 \\ 1111 \quad 1110 \\ \hline 1011 \quad 0110 \\ 182 \quad .92 \end{array} \Rightarrow 182, 12.200.1$$

$$\begin{array}{r}
 1011 & 1111 \\
 1111 & 1000 \\
 \hline
 1011 & 1000 \\
 & 184
 \end{array}$$

$$\begin{array}{r}
 1803 \quad 111 \\
 11172 \cancel{+} 160 \rightarrow 182.11.200.1 \\
 1011 \quad 1110 \\
 \hline
 188 + \quad 32 + 16 + \quad 4 + 12 \\
 \checkmark \quad 64
 \end{array}$$

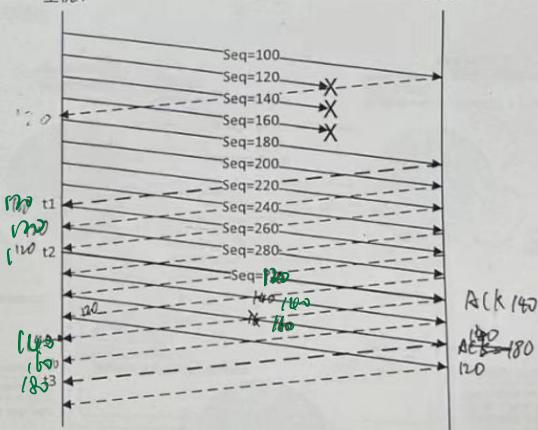
#### 四、传输层 (20 分)

如下图所示，在主机 A 和主机 B 之间建立一个 TCP 连接，A 向 B 发送 TCP 数据段，B 收到数据段立即返回 ACK（采用累积确认），拥塞控制使用 RENO 算法。假设 A 持续有数据发送，且每个数据段均包含 20 字节的数据。

- (1) (15 分) 如果整个过程中没有超时事件发生, 基于 RENO 算法的状态机回答下列问题, 并给出适当的解释:

- (a) A 在 t1 时刻收到的 ACK 段的确认序列号是多少? 120  
 (b) A 在 t2 时刻收到 ACK, 在收到该 ACK 之前 A 处于拥塞避免状态, 阈值  $ssthresh$  为 32MSS, 拥塞窗口  $cwnd$  为 40MSS。A 在收到该 ACK 之后重传的数据段的序列号为多少? 这时  $cwnd$  和  $ssthresh$  的分别为多少? 23 20

- (2) (5分) 分析 RENO 算法存在的性能问题，并给出一种合适的解决策略（开放问题）



### ► RENO算法的问题

- ✓ RENO仅考虑了丢包发生在头部丢失一个报文的情况，在一次网络拥塞引起的多个报文丢失的场景中，会把拥塞窗口多次折半
  - ✓ 后缀没有足够的报文发送速率由丢包限制不能发送更多的报文，那么，丢失的报文段只能等到重传时，进入慢启动状态
  - ✓ **问题：**增大丢包报文段的恢复时延，并严重降低TCP的吞吐率

RENO算法

  - ✓ 主要解决一次**丢弃多个报文段丢失**，而造成的选择窗口和拥塞窗口折半问题

## 五、编程 (10 分)

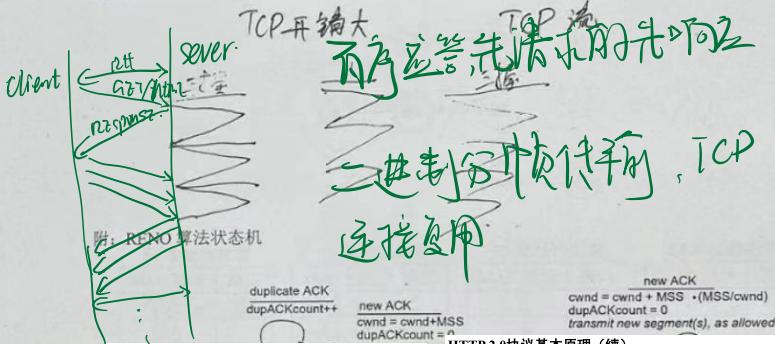
一个基于 Socket 的网络应用程序，下层使用 UDP 协议。其主要功能是客户向服务器发送一个字符串，在接收到服务器响应的字符串后退出；服务器接收客户发送的字符串，在响应一个字符串后退出。已知客户机的 IP 地址为 192.168.1.99/24，系统支持的 Socket 版本为 1.1、2.0、2.2；服务器的 IP 地址为 192.168.1.100/24，系统支持的 Socket 版本为 1.0、1.1 和 2.0。程序员初步编写的服务器端程序如下所示。请根据该程序回答下列问题。

- (1) (3 分) 服务端请求使用哪个版本的 Socket，最终应该使用的是哪个版本？ **2.2 2.0**
- (2) (5 分) 该程序有两个明显的错误，请指出错误所在的行号并进行修改。
- (3) (2 分) 请将程序第 12 行的程序补全。

```
(1) int main(int argc, char* argv[]) {
(2)     WORD sockVersion = 514;
(3)     WSADATA data;
(4)     if (WSAStartup(sockVersion, &data) != 0) return 0;
(5)     SOCKET sockLocal = socket(AF_INET, SOCK_DGRAM, 0);
(6)     if (sockLocal == INVALID_SOCKET) return 0;
(7)
(8)     struct sockaddr_in sockaddr;
(9)     sockaddr.sin_family = AF_INET;
(10)    sockaddr.sin_port = htons(8888);
(11)    sockaddr.sin_addr.S_un.S_addr = inet_addr("192.168.1.100");
(12)    if (bind(sockLocal, (SOCKADDR*)&sockAddress, sizeof(sockAddress)) ==
(13)        SOCKET_ERROR)
(14)        {closesocket(sockLocal); return 0;}
(15)
(16)    int sockaddrLen = sizeof(sockAddress);
(17)    recvfrom(sockLocal, recvStr, sizeof(recvStr), 0, (struct
(18)        sockaddr*)&sockAddress, &sockAddressLen);
(19)    printf("接收: %s\n", recvStr);
(20)
(21)    sendto(sockLocal, sendStr, strlen(sendStr), 0, (struct
(22)        sockaddr*)&sockAddress, sockAddressLen);
(23)
(24)    printf("发送: %s\n", sendStr);
(25)
(26)    closesocket(sockLocal);
(27)    WSACleanup();
(28)    return 0;
(29) }
```

## 六、应用层 (20 分)

- (1) (15 分) 客户端通过浏览器访问 Web 服务器，使用 HTTP 1.1 的长连接和流水线机制，访问页面的 URL 为 <http://www.abc.edu.cn/test.html>，页面中嵌入 3 幅图像，分别为 `image1.jpg`、`image2.jpg`、`image3.jpg`，并与上述页面保存在同一台服务器中，请画出客户端与服务器的交互过程。
- (2) (5 分) 说明 HTTP 1.1 的头阻塞问题，并解释 HTTP 2.0 的解决策略。



### HTTP 2.0 协议基本原理

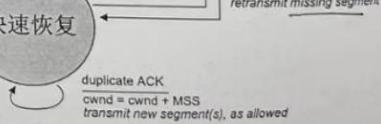
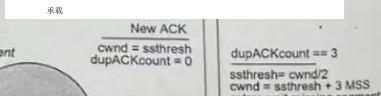
- 二进制分帧技术
  - 不改变 HTTP 原有的语义
  - 将 HTTP 请求和响应分别成帧，采用二进制编码
  - 制为最小传输单位



dupACKCount == 3  
ssthresh= cwnd/2  
cwnd = ssthresh + 3 MSS  
retransmit missing segment

### HTTP 2.0 协议基本原理（续）

- TCP 复杂度降低：提高连接利用率，解决 HTTP 的头阻塞问题
  - 消息(Message)：HTTP 一次请求或响应，包含一个或多个帧
  - 流(Stream)：简看成一次请求和应答，包含多个帧
  - 每个 TCP 连接中可以承载多个流，不同流的帧可交替穿插传输
  - 流的创建与标识
    - Stream ID：标记一个流，各客户端识别 Stream ID 为唯一；服务器识别 Stream ID 为会话；ID 为偶数 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100
  - 流创建：发送和接收的帧 HEADERS(包含 Stream ID)时创建
  - 流优先级：可以根据必要性为设置不同的优先级 (1~256)，在 HEADERS 中实现



dupACKCount == 3  
ssthresh= cwnd/2  
cwnd = ssthresh + 3 MSS  
retransmit missing segment

duplicate ACK  
cwnd = cwnd + MSS  
transmit new segment(s), as allowed

new ACK  
cwnd = cwnd + MSS •(MSS/cwnd)  
dupACKCount = 0  
transmit new segment(s), as allowed

ACK count++