

# 2016/4/5 计算机网络课堂测试讲解

🔗 <https://www.yangzhou301.com/2016/04/07/471457613/>

📅 2016 年 4 月 12 日

# 第一题

客户端 A 和服务端 B 进行了 TCP 通信。初始序列号都为 0。连接建立后，A 以 500B 的传输单元向 B 传输 5kB 的数据。其中第二个数据单元丢失，并重传了 2 次；B 向 A 发送 10kB 的数据（传输单元也为 500B），数据和 ACK 都没有丢失。所有的数据传输完毕后，TCP 连接关闭（其中 A 先发起关闭）。

- ① 画出 TCP 连接建立的过程（连接建立过程中没有丢包）；
- ② 画出关闭 TCP 连接的完整过程。

要求标出必要的标识位、序列号、确认号等。

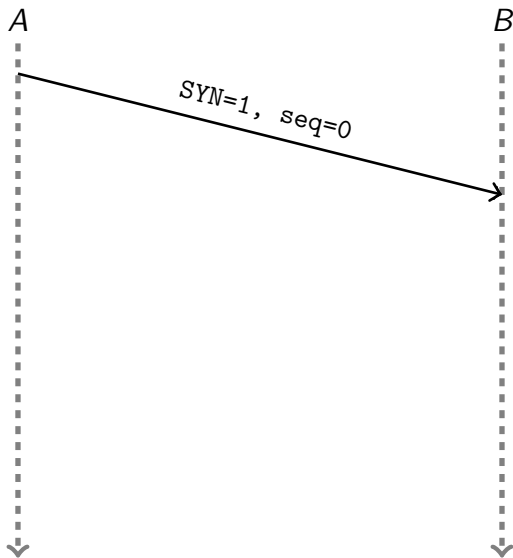
# 建立连接

## 三次握手

- ① 客户端 A 给服务器 B 发送一个 SYN 请求给服务器 B 要求建立连接，生成一个随机数作为初始序列号  $seq=X$ ，
- ② 服务器 B 收到请求，回应 SYN-ACK 表示服务器 B 能接收到客户端的数据且允许建立连接，其中  $ack=X+1$ ，这个确认号表示接收到了序列号为 X 的分片，请求发送下一个分片。而这个 SYN-ACK 消息本身的序列号由 B 随机生成  $seq=Y$
- ③ 客户端 A 收到 SYN-ACK，回应一个 ACK 给服务器 B 表示客户端能接收到服务器的数据且允许建立连接， $seq=X+1$ （也就是 B 发送的  $ack$ ），并表示接下来希望接收的服务器发来的数据为  $ack=Y+1$

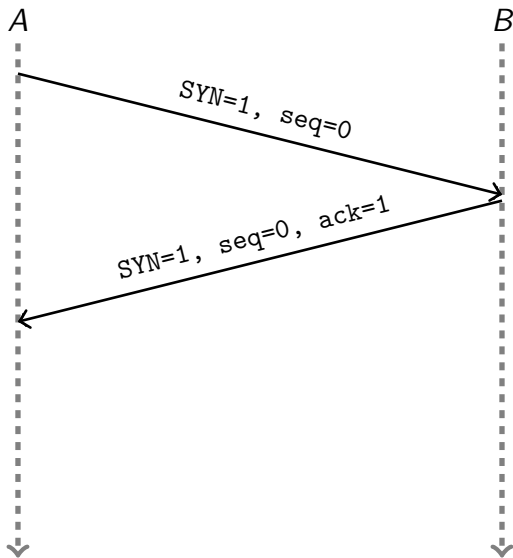
# 建立连接

A 和 B 的初始序列号为 0, A 向 B 发送 SYN 请求建立连接



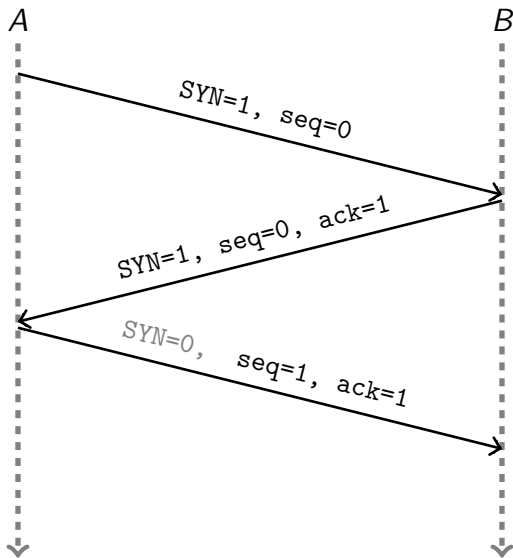
# 建立连接

B 回应 SYN-ACK 允许请求



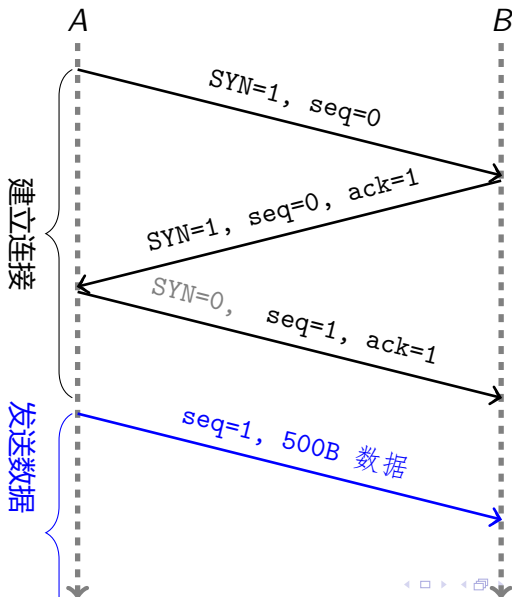
# 建立连接

A 回应 ACK 表示客户端能接收到服务器的数且允许建立连接



# 数据传输

A 发送给 B 的第一个数据单元



# 数据传输

接下来可能的情况

连接建立后，A 以 500B 的传输单元向 B 传输 5kB 的数据。其中第二个数据单元丢失，并重传了 2 次；B 向 A 发送 10kB 的数据（传输单元也为 500B），数据和 ACK 都没有丢失。



# 数据传输

接下来可能的情况

连接建立后，A 以500B的传输单元向 B 传输5kB的数据。其中第二个数据单元丢失，并重传了 2 次；B 向 A 发送10kB的数据（传输单元也为500B），数据和 ACK 都没有丢失。

- 停止等待 (*Stop-and-wait*)
- 退回 N 步 (*Go Back N*)
- 选择重传 (*Selective Repeat*)

# 数据传输

接下来可能的情况

连接建立后，A 以500B的传输单元向 B 传输5kB的数据。其中第二个数据单元丢失，并重传了 2 次；B 向 A 发送10kB的数据（传输单元也为500B），数据和 ACK 都没有丢失。

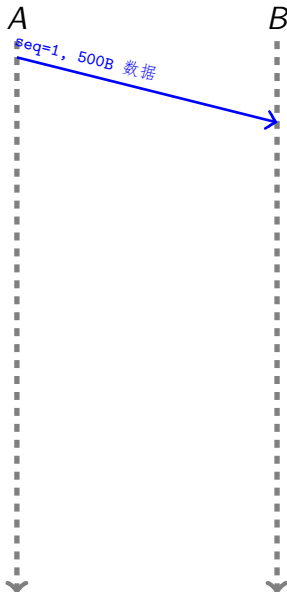
- 停止等待 (*Stop-and-wait*)
- 退回 N 步 (*Go Back N*)
- 选择重传 (*Selective Repeat*)

但都不会影响 seq 和 ack 最后的编号

- 可靠数据传输：收到的分片必须有序，重传的分片序列号 seq 必然与原分片相同
- 只有当上一个分片传输了数据（包括 SYN 和 FIN 也算在内），且不是重传的情况下，下一个分片的 seq 才可以增加

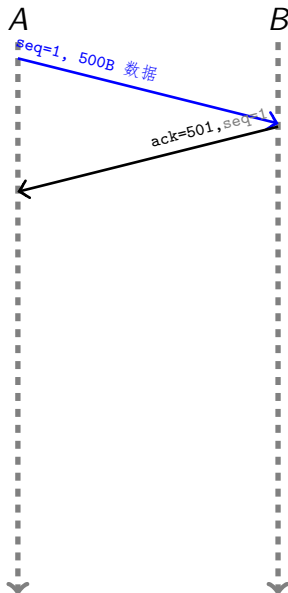
# 数据传输

一种可能的情况 (选择重传)



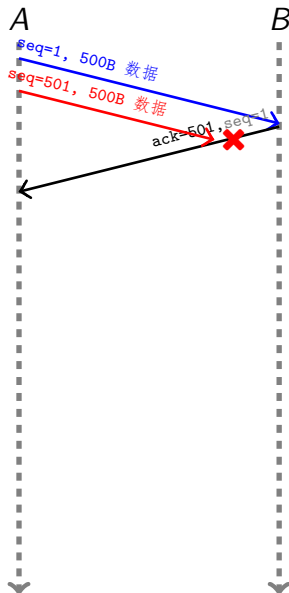
# 数据传输

一种可能的情况 (选择重传)



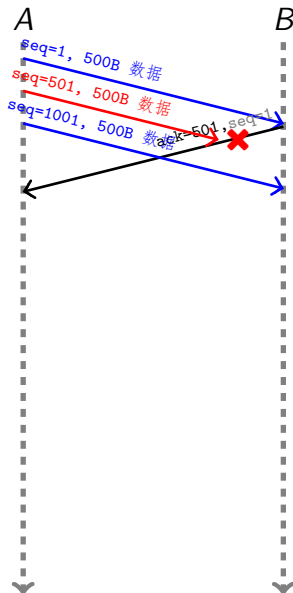
# 数据传输

一种可能的情况 (选择重传)



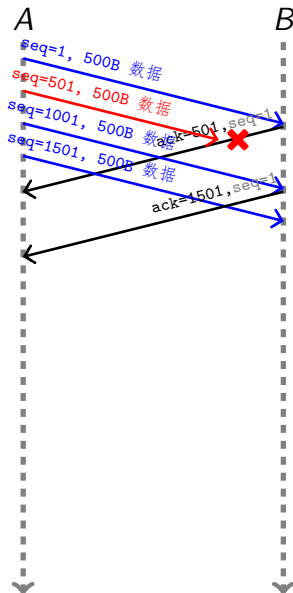
# 数据传输

一种可能的情况 (选择重传)



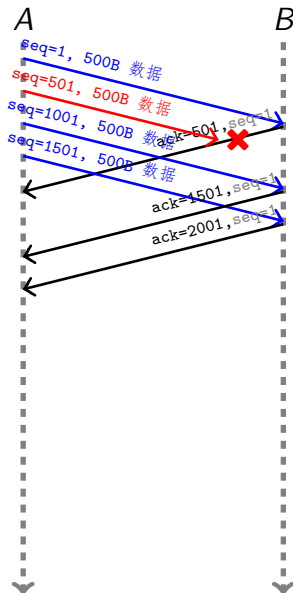
# 数据传输

一种可能的情况 (选择重传)



# 数据传输

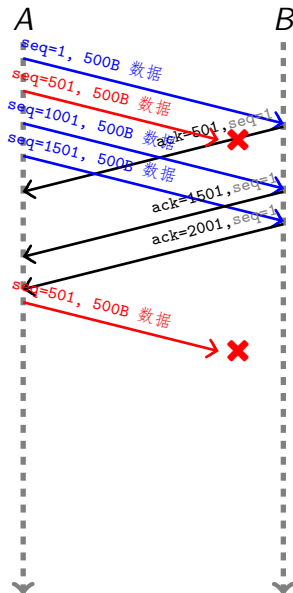
一种可能的情况 (选择重传)





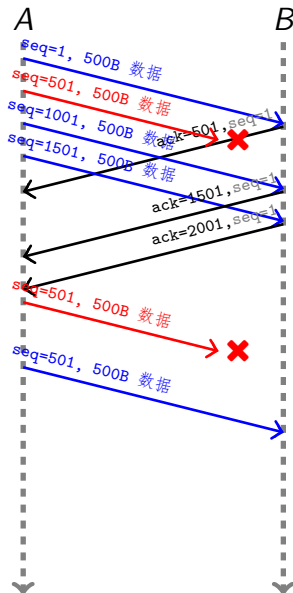
# 数据传输

一种可能的情况 (选择重传)



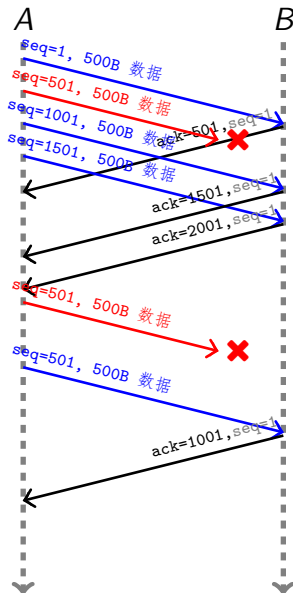
# 数据传输

一种可能的情况 (选择重传)



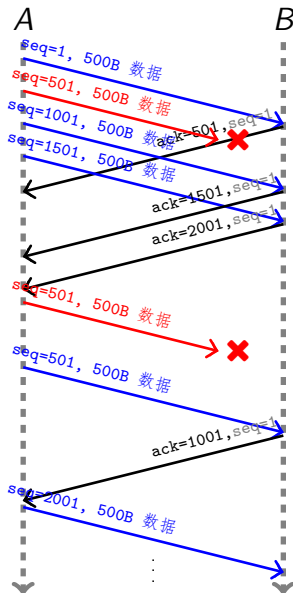
# 数据传输

一种可能的情况 (选择重传)



# 数据传输

一种可能的情况 (选择重传)



# 关闭连接

## 四次挥手

- ① 客户端 A 向服务器 B 发送一个 FIN 请求关闭连接
- ② 服务器 B 接收到请求后回应一个 ACK 表示允许关闭连接，至此 A 到 B 的连接已经关闭，A 不能再向 B 发送数据，这是半关闭 (*half-closed*) 状态
- ③ 服务器 B 向客户端 A 发送一个 FIN 请求关闭连接
- ④ 客户端 A 接收到请求后回应一个 ACK 表示允许关闭连接，至此连接关闭

# 关闭连接

## 四次挥手

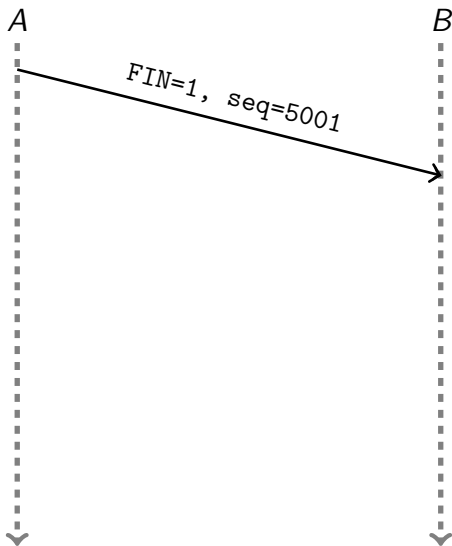
- 1 客户端 A 向服务器 B 发送一个 FIN 请求关闭连接
- 2 服务器 B 接收到请求后回应一个 ACK 表示允许关闭连接，至此 A 到 B 的连接已经关闭，A 不能再向 B 发送数据，这是半关闭 (*half-closed*) 状态
- 3 服务器 B 向客户端 A 发送一个 FIN 请求关闭连接
- 4 客户端 A 接收到请求后回应一个 ACK 表示允许关闭连接，至此连接关闭

数据传输结束后的初始序列号和确认号：

- A:  $seq = 5001, ack = 10001$
- B:  $seq = 10001, ack = 5001$

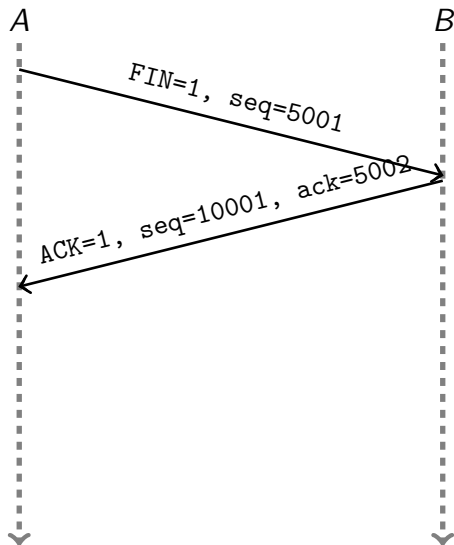
# 关闭连接

客户端 A 向服务器 B 发送一个 FIN 请求关闭连接



# 关闭连接

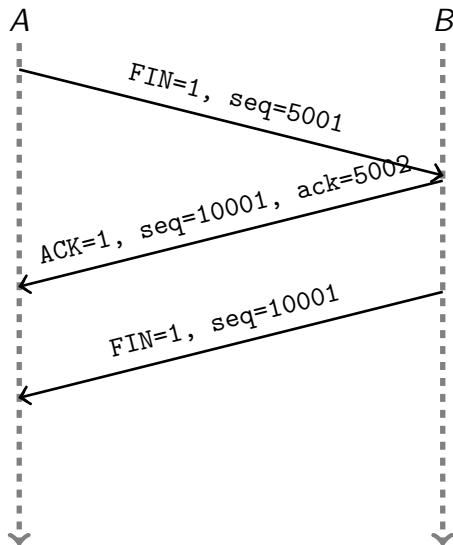
服务器 B 接收到请求后回应一个 ACK 表示允许关闭连接





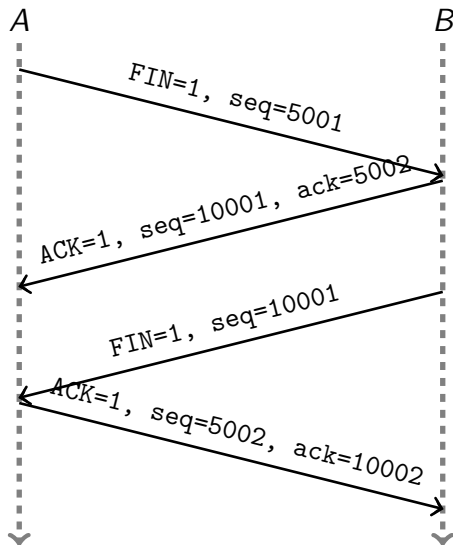
# 关闭连接

服务器 B 向客户端 A 发送一个 FIN 请求关闭连接



# 关闭连接

客户端 A 接收到请求后回应一个 ACK 表示允许关闭连接



## 第二题

某路由器路由表如下表所示，现在收到 5 个分组，其目的 IP 地址分别为：

目的 IP 地址	子网掩码	下一跳路由器
128.96.39.0	255.255.255.128	接口 0
128.96.39.128	255.255.255.128	接口 1
128.96.40.0	255.255.255.128	$R_2$
192.4.153.0	255.255.255.192	$R_3$
默认		$R_4$

分别计算并写出下一跳路由器

- ① 128.96.39.10
- ② 128.96.40.12
- ③ 128.96.40.151
- ④ 192.4.153.17
- ⑤ 192.4.153.90

将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

128.96.39.10 和 255.255.255.128 相与得到 128.96.39.0，下一跳为接口 0

将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

128.96.39.10 和 255.255.255.128 相与得到 128.96.39.0，下一跳为接口 0

128.96.40.12 和 255.255.255.128 相与得到 128.96.40.0，下一跳  $R_2$

将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

128.96.39.10 和 255.255.255.128 相与得到 128.96.39.0，下一跳为接口 0

128.96.40.12 和 255.255.255.128 相与得到 128.96.40.0，下一跳  $R_2$

128.96.40.151 和 255.255.255.128 相与得到 128.96.40.128 不在目的 IP 地址表中，再与 255.255.255.192 的运算结果为 128.96.40.128 不在目的 IP 地址表中，所以走缺省路由  $R_4$

将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

128.96.39.10 和 255.255.255.128 相与得到 128.96.39.0，下一跳为接口 0

128.96.40.12 和 255.255.255.128 相与得到 128.96.40.0，下一跳  $R_2$

128.96.40.151 和 255.255.255.128 相与得到 128.96.40.128 不在目的 IP 地址表中，再与 255.255.255.192 的运算结果为 128.96.40.128 不在目的 IP 地址表中，所以走缺省路由  $R_4$

192.4.153.17 和 255.255.255.128 相与得到 192.4.153.0 不在目的 IP 地址表中，与 255.255.255.192 的运算结果为 192.4.153.0，下一跳  $R_3$



将 IP 地址和子网掩码作与运算，找到目的 IP 地址和运算结果一致的路由项，若不存在，则走缺省路由

128.96.39.10 和 255.255.255.128 相与得到 128.96.39.0，下一跳为接口 0

128.96.40.12 和 255.255.255.128 相与得到 128.96.40.0，下一跳  $R_2$

128.96.40.151 和 255.255.255.128 相与得到 128.96.40.128 不在目的 IP 地址表中，再与 255.255.255.192 的运算结果为 128.96.40.128 不在目的 IP 地址表中，所以走缺省路由  $R_4$

192.4.153.17 和 255.255.255.128 相与得到 192.4.153.0 不在目的 IP 地址表中，与 255.255.255.192 的运算结果为 192.4.153.0，下一跳  $R_3$

192.4.153.90 和 255.255.255.128 相与得到 192.4.153.0 不在目的 IP 地址表中，与 255.255.255.192 的运算结果为 192.4.153.64 不在目的 IP 地址表中，所以走缺省路由  $R_4$

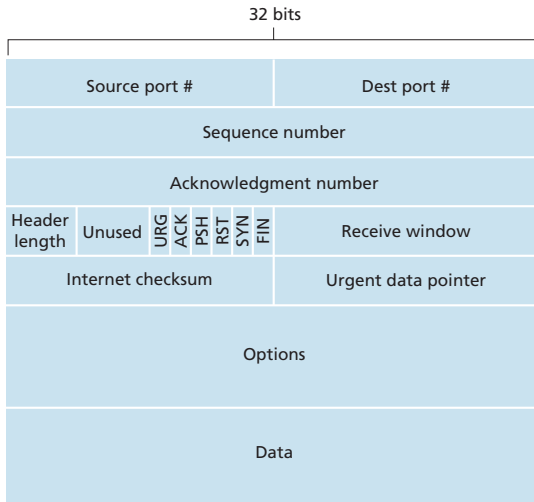
# 第一题

一个 TCP 的首部字节数据见下表所示，回答下列问题：

编号	1	2	3	4	5	6	7	8	9
数据	0d	28	00	15	00	5f	a9	06	00
编号	10	11	12	13	14	15	16	17	18
数据	00	00	00	70	02	40	00	C0	29
编号	19	20							
数据	00	00							

- ① 源端口号是多少？目的端口号是多少？
- ② 发送的序列号是多少？确认号是多少？
- ③ TCP 首部的长度是多少？
- ④ 这是一个使用什么协议的 TCP 连接？( 提示：观察端口号 ) 该 TCP 连接的状态是什么？

# TCP 报文结构



# 寻找对应字段

注意这里编号从 1 开始

# 寻找对应字段

注意这里编号从 1 开始

源端口号 编号 1-2 的数据为 0d28 , 转换为十进制数 3368

# 寻找对应字段

注意这里编号从 1 开始

源端口号 编号 1-2 的数据为 0d28 , 转换为十进制数 3368

目的端口号 编号 3-4 的数据为 0015 , 转换为十进制数 21 , 这是一个  
FTP 连接

# 寻找对应字段

注意这里编号从 1 开始

源端口号 编号 1-2 的数据为 0d28，转换为十进制数 3368

目的端口号 编号 3-4 的数据为 0015，转换为十进制数 21，[这是一个 FTP 连接](#)

序列号 编号 5-8 的数据为 005fa906，转换为十进制数 6269190

# 寻找对应字段

注意这里编号从 1 开始

源端口号 编号 1-2 的数据为 0d28，转换为十进制数 3368

目的端口号 编号 3-4 的数据为 0015，转换为十进制数 21，[这是一个 FTP 连接](#)

序列号 编号 5-8 的数据为 005fa906，转换为十进制数 6269190

确认号 编号 9-12 的数据为 00000000，转换为十进制数 0



# 寻找对应字段

注意这里编号从 1 开始

**源端口号** 编号 1-2 的数据为 0d28，转换为十进制数 3368

**目的端口号** 编号 3-4 的数据为 0015，转换为十进制数 21，**这是一个 FTP 连接**

**序列号** 编号 5-8 的数据为 005fa906，转换为十进制数 6269190

**确认号** 编号 9-12 的数据为 00000000，转换为十进制数 0

**首部长度** 编号 13 的前 4 位，对应的数值为 7 首部长度字段的单位是字 (4B,32bit)，也就是说这个报文的首部长度是 28B

# TCP 连接状态

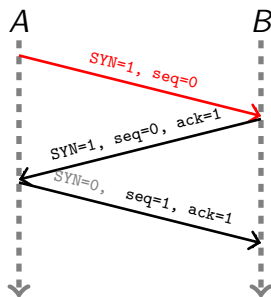
注意 SYN、ACK、FIN 标识位

ACK 编号 14 的第 4 位, 数值为 0, 该消息没有捎带 ACK

SYN 编号 14 的第 7 位, 数值为 1, 发送了一个 SYN 消息

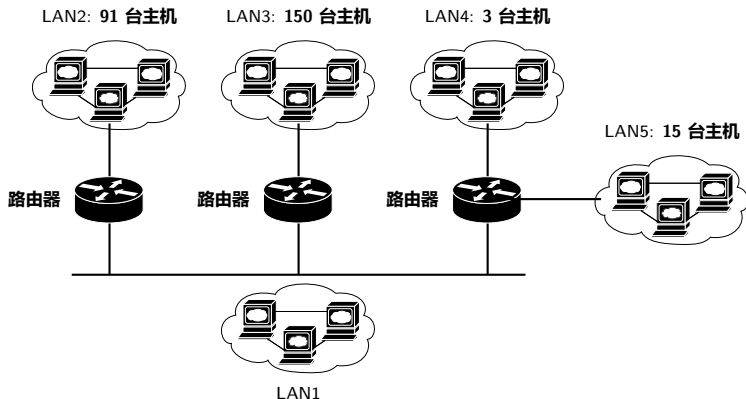
FIN 编号 14 的第 8 位, 数值为 0, 没有捎带 FIN

ack 确认号为 0, 说明没有收到对方发来的数据 (同步未进行), 因此这是一个发起的 SYN 请求, 是 TCP 建立连接时三次握手的第一条消息, 这个连接在建立状态。



## 第二题

一个自治系统有 5 个局域网，如下图所示。LAN2 至 LAN5 上的主机数分别是 91、150、3 和 15，该自治系统分配到的 IP 地址块为 30.138.118/23，试给出每个局域网的地址块。（提示：路由器（包括边界路由器）也需占用一个 IP 地址）



# 分配原则

- 有多种分配方案
- 优先考虑主机数多的局域网
- 主机数  $x$ , 连接的路由数  $y$ , 那么分配的主机地址数  $n$  需要满足

$$2^{n-1} \leq x + y < 2^n$$

# 一种可行的分配方案

仅供参考

- LAN3 有 150 台主机和一个路由器，需要 8 位的主机地址。网络地址则是剩下的 24 位，取 IP 地址的第 24 位为 0，将 30.138.118.0/24 分配给 LAN3
- LAN2 有 91 台主机和一个路由器，需要 7 位的主机地址。网络地址则是剩下的 25 位，可以取 IP 地址的第 24 位为 1，第 25 位为 0，将 30.138.119.0/25 分配给 LAN2
- LAN5 有 15 台主机和一个路由器，需要 5 位主机地址，剩下 27 位为网络地址，取 24,25,26,27 位为 1110，将 30.138.118.192/27 分配给 LAN5

# 一种可行的分配方案

仅供参考

- LAN1 有三个路由，而且作为自治域至少要有有一个边界路由与其他自治域相连，所有至少需要 4 个 IP，取 3 位作为主机地址，IP 地址的第 24,25,26,27,28,29 位可以取 111101，分配地址段 30.138.119.232/29
- LAN4 有 3 台主机和一个路由同理使用 3 位主机地址，第 24,25,26,27,28,29 位可以取 111110，分配地址段 30.138.119.240/29

## 第三题 \*

将以下 IPV6 地址用零压缩方法写成简洁形式。

- ① 0000:0000:0F53:6382:AB00:67DB:BB27:7332
- ② 0000:0000:0000:0000:0000:0000:004D:ABCD
- ③ 0000:0000:0000:AF36:7328:000A:87AA:0398
- ④ 2819:00AF:0000:0000:0000:0035:0CB2:B271

# 前导零压缩

- 如果一个位段为 0000，那么可以用 0 进行代替。
- IPv6 每段地址前端的 0 可以省略，中间和后端则不可以。
- 几个连续的位段为 0，可以用:: 代替这些 0。
- :: 在 IPv6 中只能出现一次。



# 答案

- ::F53:6382:AB00:67DB:BB27:7332
- ::4D:ABCD
- ::AF36:7328:A:87AA:398
- 2819:AF::35:CB2:B271