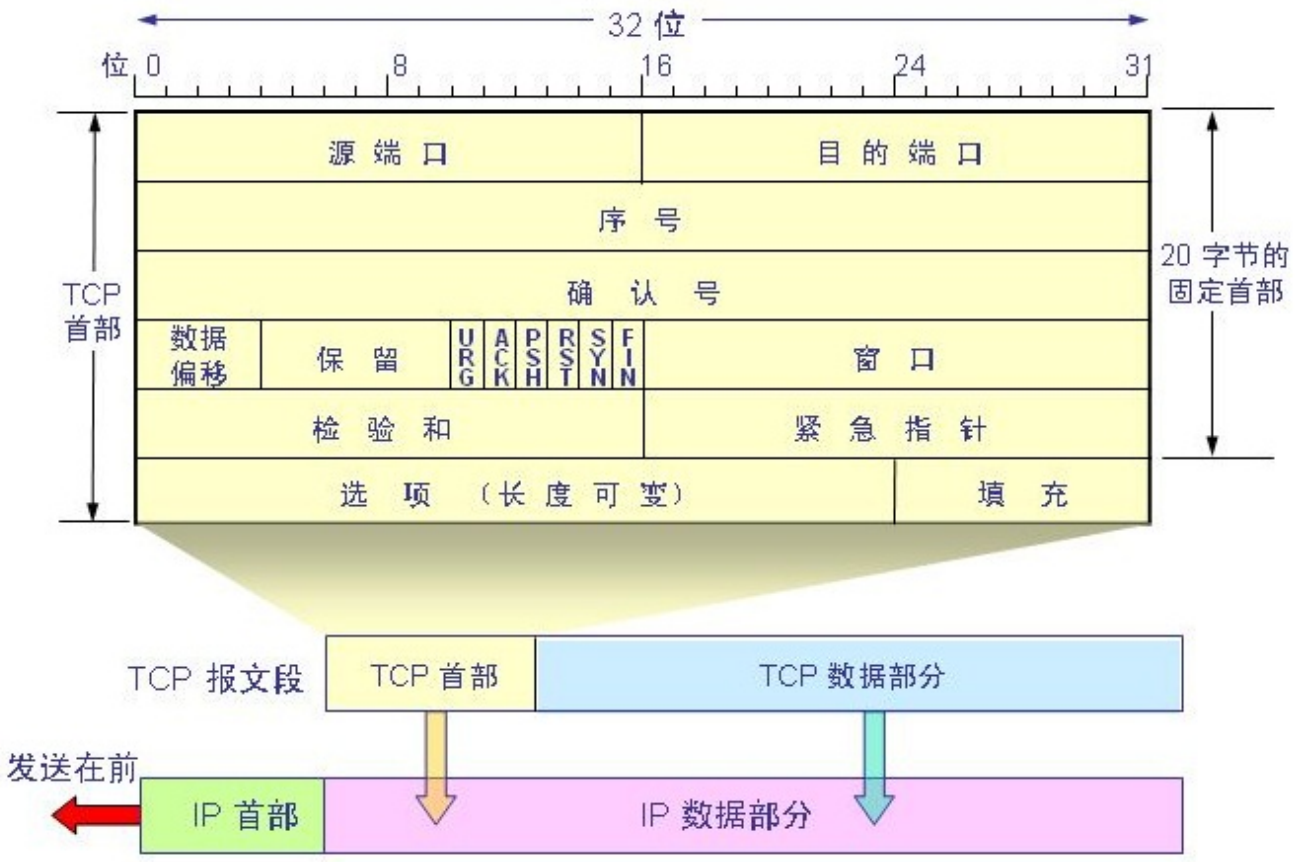
TCP/IP协议整理——报文/流程解析

TCP报文解析

1 tcp报文格式图示



2 报文头各项解析

固定长度20字节，变长0~40字节，整个报文<=60字节且必为4的倍数。

2.1 端口项

源端口(16bit)：发送方的程序进程的端口；

目的端口(16bit)：接收方的程序进程的端口；

2.2 序号与确认号项

序号(32bit)：标识本报文段数据内容的第一个字节的号码；

确认号(32bit)：标识期望接收方回复本报文的报文**序号**，仅当ack位为1时有效；

EX:假设在**A收到B的tcp报文**后，回复了一个报文段【序号:20;确认号100,ack位:1】。正常情况下，A从B收到的报文的确认号是20；而期望B回复A的报文的序号是100。

2.3 数据偏移项(4bit)

TCP报文段的数据部分的起始处距离TCP报文段的起始处的距离 = tcp数据头的长度；

该值的单位是4byte，4bit表示0~15 => 能表示header长度为0~60byte，header的长度也一定是4的倍数。

2.4 保留字段(6bit)

留待后用未使用字段，一般置为0。

2.5 控制位项(共6位，每个控制位各1bit)

URG：紧急指针标志，为1时表示紧急指针有效，为0则忽略紧急指针；

ARK：确认序号标志，为1时表示确认号有效，为0表示忽略确认号字段；

PSH：push标志，报文段会被尽快地交付给目的方，不会对报文段使用缓存策略；

RST：重置连接标志，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：同步标志，用于建立连接过程，连接请求(SYN:1,ACK:0)，连接应答(SYN:1,ACK:1)；

FIN：结束标志，用于释放连接，为1时表示本方已没有数据传送。

2.6 窗口项 (16bit)

滑动窗口大小，用于流量控制。用于告知对方本进程当前能够用于接收的缓存大小。

2.7 校验和 (16bit)

对整个报文进行奇偶校验以保证数据正确性，由发送方计算和存储，接收方验证。

2.8 紧急指针 (16bit)

URG位置1时有效。该指针是偏移量，与**序号**相加表示紧急数据的最后一个字节的序号。

2.9 选项 (0~40byte)

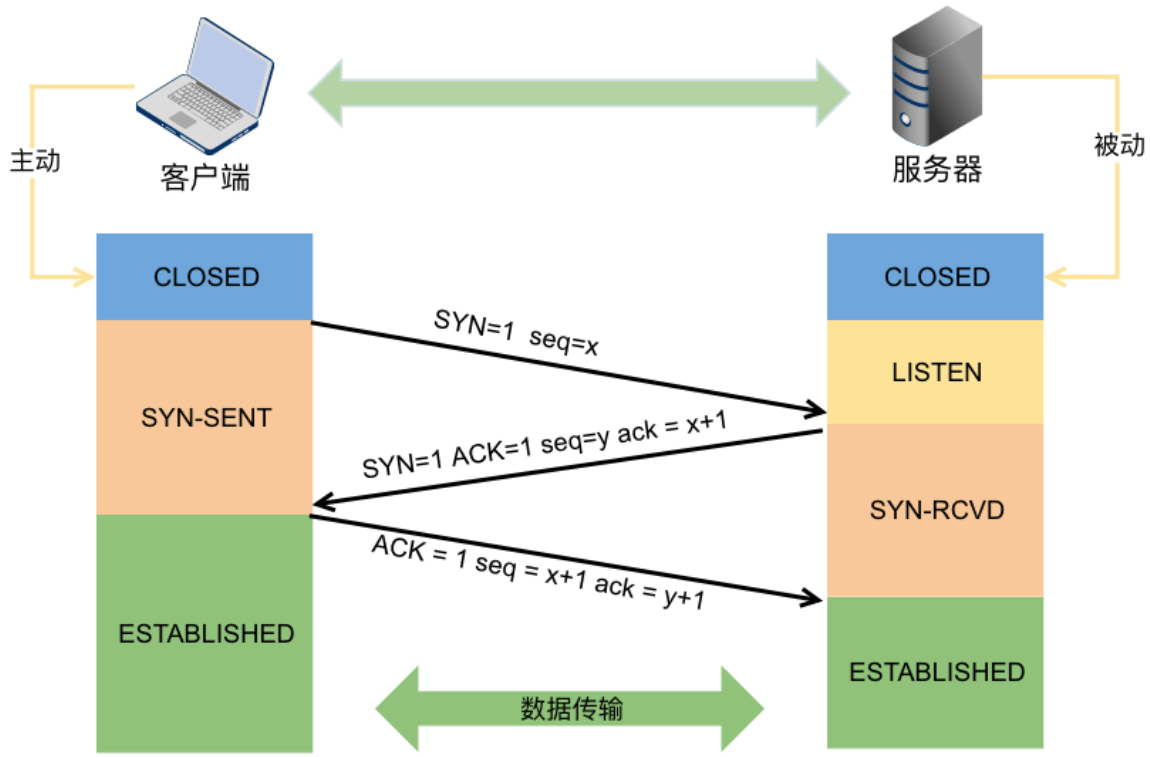
常用option如下，一般为TLV格式(kind/type-length-value)，根据实际使用的选项略有不同。（此处不再展开，有兴趣请自行查询）

Kind (Type)	Length	Name	Reference	描述 & 用途
0	1	EOL	RFC 793	选项列表结束
1	1	NOP	RFC 793	无操作（用于补位填充）
2	4	MSS	RFC 793	最大Segment长度
3	3	WSOPT	RFC 1323	窗口扩大系数（Window Scaling Factor）
4	2	SACK-Premitted	RFC 2018	表明支持SACK
5	可变	SACK	RFC 2018	SACK Block（收到乱序数据）
8	10	TSPOT	RFC 1323	Timestamps
19	18	TCP-MD5	RFC 2385	MD5认证
28	4	UTO	RFC 5482	User Timeout（超过一定闲置时间后拆除连接）
29	可变	TCP-AO	RFC 5925	认证（可选用各种算法）
253/254	可变	Experimental	RFC 4727	保留，用于科研实验

TCP流程解析

1 TCP连接建立过程——“三次握手”

概念图示



抓包举例

```
1 0.000000    172.16.57.212    172.19.0.2       TCP    66 49167 → 28002 [SYN] Seq=0 Win=64240 Len=0 MSS=1380 WS=256 SACK_PERM=1
2 0.000025    172.19.0.2       172.16.57.212    TCP    66 28002 → 49167 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3 0.004236    172.16.57.212    172.19.0.2       TCP    54 49167 → 28002 [ACK] Seq=1 Ack=1 Win=131072 Len=0
```

1.1 Step 1, From client to server

```
1 0.000000    172.16.57.212    172.19.0.2       TCP    66 49167 → 28002 [SYN] Seq=0 Win=64240 Len=0 MSS=1380 WS=256 SACK_PERM=1
2 0.000025    172.19.0.2       172.16.57.212    TCP    66 28002 → 49167 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=12
3 0.004236    172.16.57.212    172.19.0.2       TCP    54 49167 → 28002 [ACK] Seq=1 Ack=1 Win=131072 Len=0
```

Server开机，进入监听状态

Client发送连接请求报文【**SYN=1，ACK=0**，Seq=0，无数据内容(len=0)】。发送完进入SYN-SENT状态等待Server应答连接建立

```
Transmission Control Protocol, Src Port: 49167, Dst Port: 28002, Seq: 0, Len: 0
    Source Port: 49167
    Destination Port: 28002
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1411361778
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····S·]
```

1.2 Step 2, From server to client

1	0.000000	172.16.57.212	172.19.0.2	TCP	66 49167 → 28002 [SYN] Seq=0 Win=64240 Len=0 MSS=1380 WS=256 SACK_PERM=1
2	0.000025	172.19.0.2	172.16.57.212	TCP	66 28002 → 49167 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3	0.004236	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [ACK] Seq=1 Ack=1 Win=131072 Len=0

Server收到连接请求报文后，回复连接应答报文【**SYN=1，ACK=1**（此处为控制位），Seq=0，Ack=1=0+1=连接请求Seq+1（此处为确认号），无数据内容】

Server进入SYN-RECV状态等待Client确认收到连接应答

```
Transmission Control Protocol, Src Port: 28002, Dst Port: 49167, Seq: 0, Ack: 1, Len: 0
    Source Port: 28002
    Destination Port: 49167
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1953065111
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1411361779
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A··S·]
```

1.3 Step 3, From client to server

1	0.000000	172.16.57.212	172.19.0.2	TCP	66 49167 → 28002 [SYN] Seq=0 Win=64240 Len=0 MSS=1380 WS=256 SACK_PERM=1
2	0.000025	172.19.0.2	172.16.57.212	TCP	66 28002 → 49167 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3	0.004236	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [ACK] Seq=1 Ack=1 Win=131072 Len=0

Client收到连接应答报文后，发送确认连接应答报文【**SYN=0，ACK=1**，Seq=1=连接应答的Ack，Ack=1=连接应答的Seq+1，无数据内容（该步骤可以带数据内容）】

Server收到该报文确认无误后即代表两方正式建立tcp连接

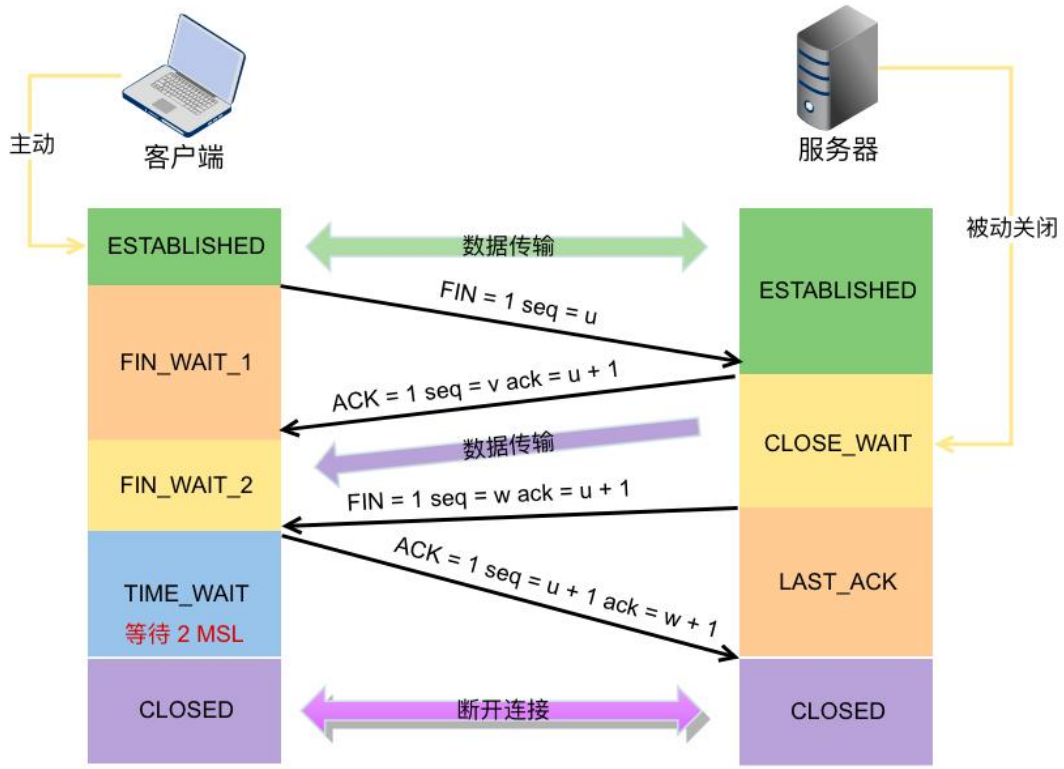
```
    Source Port: 49167
    Destination Port: 28002
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1411361779
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1953065112
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
```

1.4 “三次握手”是为了确保数据的对等性

第 N 次握手	A 机器确认				B 机器确认			
	自己发报能力	自己收报能力	对方发报能力	对方收报能力	自己发报能力	自己收报能力	对方发报能力	对方收报能力
第 1 次握手后	NO	NO	NO	NO	NO	YES	YES	NO
第 2 次握手后	YES	YES	YES	YES	NO	YES	YES	NO
第 3 次握手后	YES	YES	YES	YES	YES	YES	YES	YES

2 TCP连接释放过程——“四次挥手”

概念图示



抓包举例

```
160 4.140326    172.19.0.2      172.16.57.212   TCP      54 28002 → 49167 [FIN, ACK] Seq=641 Ack=224575 Win=313856 Len=0
161 4.146769    172.16.57.212   172.19.0.2      TCP      54 49167 → 28002 [ACK] Seq=224575 Ack=642 Win=130304 Len=0
162 4.146773    172.16.57.212   172.19.0.2      TCP      54 49167 → 28002 [FIN, ACK] Seq=224575 Ack=642 Win=130304 Len=0
163 4.146795    172.19.0.2      172.16.57.212   TCP      54 28002 → 49167 [ACK] Seq=642 Ack=224576 Win=313856 Len=0
```

2.1 Step 1,From client to server

```
160 4.140326    172.19.0.2      172.16.57.212   TCP      54 28002 → 49167 [FIN, ACK] Seq=641 Ack=224575 Win=313856 Len=0
161 4.146769    172.16.57.212   172.19.0.2      TCP      54 49167 → 28002 [ACK] Seq=224575 Ack=642 Win=130304 Len=0
162 4.146773    172.16.57.212   172.19.0.2      TCP      54 49167 → 28002 [FIN, ACK] Seq=224575 Ack=642 Win=130304 Len=0
163 4.146795    172.19.0.2      172.16.57.212   TCP      54 28002 → 49167 [ACK] Seq=642 Ack=224576 Win=313856 Len=0
```

Client请求释放连接【**FIN=1**，ACK=1（这个ACK与连接释放过程**无关**），Seq=641，Ack=224575】

发送完后进入FIN_WAIT_1，等待server确认

```
Transmission Control Protocol, Src Port: 28002, Dst Port: 49167, Seq: 641, Ack: 224575, Len: 0
    Source Port: 28002
    Destination Port: 49167
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 641    (relative sequence number)
    Sequence Number (raw): 1953065752
    [Next Sequence Number: 642    (relative sequence number)]
    Acknowledgment Number: 224575    (relative ack number)
    Acknowledgment number (raw): 1411586353
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x011 (FIN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
    > .... .... ...1 = Fin: Set
      [TCP Flags: ·····A···F]
```

2.2 Step 2,From server to client

160	4.140326	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [FIN, ACK] Seq=641 Ack=224575 Win=313856 Len=0
161	4.146769	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [ACK] Seq=224575 Ack=642 Win=130304 Len=0
162	4.146773	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [FIN, ACK] Seq=224575 Ack=642 Win=130304 Len=0
163	4.146795	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [ACK] Seq=642 Ack=224576 Win=313856 Len=0

Server收到释放连接请求报文后，发送确认报文（这里仅**确认收到**连接释放请求报文，**没有答复**该报文）**【ACK=1**，Seq=224574=释放连接请求的Ack，Ack=642=641+1=释放连接请求的Seq+1**】**

Server进入CLOSED_WAIT状态，传输未传完的数据（如果有）

Client收到后继续接受server未传完的数据（如果有的话），进入Fin_Wait_2阶段继续等待server发送连接释放应答。

```
Transmission Control Protocol, Src Port: 49167, Dst Port: 28002, Seq: 224575, Ack: 642, Len: 0
    Source Port: 49167
    Destination Port: 28002
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 224575    (relative sequence number)
    Sequence Number (raw): 1411586353
    [Next Sequence Number: 224575    (relative sequence number)]
    Acknowledgment Number: 642    (relative ack number)
    Acknowledgment number (raw): 1953065753
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
```

2.3 Step 3, From server to client

160	4.140326	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [FIN, ACK] Seq=641 Ack=224575 Win=313856 Len=0
161	4.146769	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [ACK] Seq=224575 Ack=642 Win=130304 Len=0
162	4.146773	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [FIN, ACK] Seq=224575 Ack=642 Win=130304 Len=0
163	4.146795	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [ACK] Seq=642 Ack=224576 Win=313856 Len=0

Server在传输完自己所有的数据后，发送连接释放连接应答报文，允许释放连接**【FIN=1，ACK=1**，Seq=224574，Ack=642（Seq与Ack相比Step2都不变，因为在期间没有收到Client的报文）**】**

Server发送释放应答报文后进入LAST-ACK状态，等待Client确认

```
Transmission Control Protocol, Src Port: 49167, Dst Port: 28002, Seq: 224575, Ack: 642, Len: 0
    Source Port: 49167
    Destination Port: 28002
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 224575    (relative sequence number)
    Sequence Number (raw): 1411586353
    [Next Sequence Number: 224576    (relative sequence number)]
    Acknowledgment Number: 642    (relative ack number)
    Acknowledgment number (raw): 1953065753
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x011 (FIN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
    > .... .... ...1 = Fin: Set
      [TCP Flags: ·····A···F]
```

2.4 Step 4, From client to srever

160	4.140326	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [FIN, ACK] Seq=641 Ack=224575 Win=313856 Len=0
161	4.146769	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [ACK] Seq=224575 Ack=642 Win=130304 Len=0
162	4.146773	172.16.57.212	172.19.0.2	TCP	54 49167 → 28002 [FIN, ACK] Seq=224575 Ack=642 Win=130304 Len=0
163	4.146795	172.19.0.2	172.16.57.212	TCP	54 28002 → 49167 [ACK] Seq=642 Ack=224576 Win=313856 Len=0

Client接收到释放连接应答报文后，回复确认报文【**ACK=1**，Seq=642=连接释放应答的Ack，Ack=224576=224575+1=连接释放应答的Seq+1】

Server接到Client的确认后直接关闭连接，不再回复；Client进入TIME_WAIT阶段等待两个MSL保证所有的数据都接收完毕后关闭连接。

```
Transmission Control Protocol, Src Port: 28002, Dst Port: 49167, Seq: 642, Ack: 224576, Len: 0
    Source Port: 28002
    Destination Port: 49167
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 642    (relative sequence number)
    Sequence Number (raw): 1953065753
    [Next Sequence Number: 642    (relative sequence number)]
    Acknowledgment Number: 224576    (relative ack number)
    Acknowledgment number (raw): 1411586354
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
```

3 TCP连接可以应用于更多的模式，不局限于C-S

虽然说明举例中用的是c-s模式去说明tcp过程，但是可以发现在示例中提出连接请求和提出释放连接的不是同一台主机，说明不是c-s模式。

IP数据报解析

1 IP报文格式图示

2 报文头各项解析

最大长度60byte，固定长度20byte，可变长度0~40byte，一定为4的倍数。

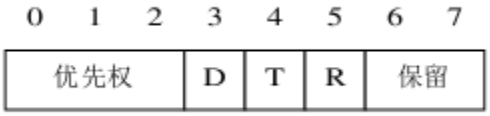
2.1 版本号(4bit)

区分ipv4和ipv6

2.2 首部长度(4bit)

该值的单位是4byte，4bit表示0~15 => 能表示header长度为0~60byte，header的长度也一定是4的倍数。

2.3 服务类型(TOS,Type Of Service) (8bit)



2.3.1 字段说明

a. 优先权（0-7）数越大，表示该数据报优先权越高。优先权可被用于拥塞控制。8个优先级的定义如下：

- 111--Network Control（网络控制）；
- 110--Internetwork Control（网间控制）；
- 101--Critic（关键）；
- 100--Flash Override（疾速）；
- 011--Flash（闪速）；
- 010--Immediate（快速）；
- 001--Priority（优先）；
- 000 - Routine（普通）。

b. 短延迟位D(Delay)：该位置1时，数据报请求以短延迟信道传输，0表示正常延时。

c. 高吞吐量位T(Throughput)：该位置1时，数据报请求以高吞吐量信道传输，0表示普通。

d. 高可靠位R(Reliability)：该位置1时，数据报请求以高可靠性信道传输，0表示普通。

e. 保留位。

2.3.2 常见应用程序的TOS值

应用程序	短延迟位D	高吞吐量位T	高可靠性位	低成本位	十六进制值	特性
Telnet	1	0	0	0	0x10	短延迟
FTP控制	1	0	0	0	0x10	短延迟
FTP数据	0	1	0	0	0x08	高吞吐量
TFTP	1	0	0	0	0x10	短延迟
SMTP命令	1	0	0	0	0x10	短延迟
SMTP数据	0	1	0	0	0x08	高吞吐量
DNS UDP查询	1	0	0	0	0x10	短延迟
DNS TCP查询	0	0	0	0	0x00	普通
DNS 区域传输	0	1	0	0	0x08	高吞吐量
ICMP差错	0	0	0	0	0x00	普通
ICMP查询	0	0	0	0	0x00	普通
SNMP	0	0	1	0	0x04	高可靠性
IGP	0	0	1	0	0x04	高可靠性
NNTP	0	0	0	1	0x02	低成本

2.4 总长度（16bit）

IP报文总长度=IP报文头长度+数据部分，单位为字节。

2.5 标识（16bit）

当IP报文的长度大于当前网络的MTU（Maximum Transmission Unit）将会进行分片。该报文标识赋给所有分片片段，表示它们同属一个报文，以便接收端拼成原报文。

2.6 标志（3bit）

只有后两位有意义。第二位为DF，置1时表示不能分片，置0表示可以分片。

2.7 片偏移（13bit）

表示较长分组在分片后，某分片在原分组的相对位置。以8个字节为单位。

2.8 生存时间（TTL）(8bit)

指定了数据报可以在网络中传输的最长时间。实际应用中把生存时间字段设置成了数据报可以经过的最大路由器数。每经过一个路由，该值减一。

2.9 协议（8bit）

表示承载的上层协议，常见协议如下

十进制编号	协　议	说　　明
0	无	保留
1	ICMP	网际控制报文协议
2	IGMP	网际组管理协议
3	GGP	网关—网关协议
4	无	未分配
5	ST	流
6	TCP	传输控制协议
8	EGP	外部网关协议
9	IGP	内部网关协议
11	NVP	网络声音协议
17	UDP	用户数据报协议

2.10 首部检验和（16bit）

占用16位二进制数，用于协议头数据有效性的校验，可以保证IP报头区在传输时的正确性和完整性。

原理：发送端首先将检验和字段置0，然后对头部中每16位二进制数进行反码求和的运算，并将结果存在校验和字段中。由于接收方在计算过程中包含了发送方放在头部的校验和，因此，如果头部在传输过程中没有发生任何差错，那么接收方计算的结果应该是全1。

2.11 地址项(64bit)

源地址（32bit）：发送端ip地址

目的地址(32bit)：接收端ip地址