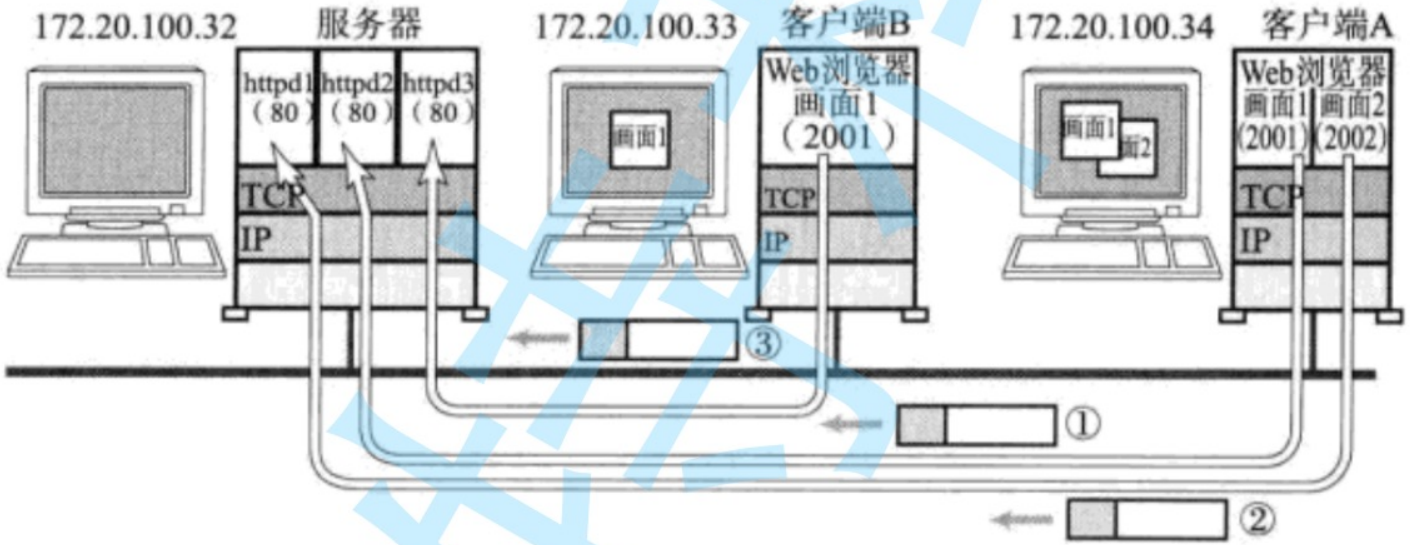
0307_Udp_Tcp

在TCP/IP协议中, 用 "源IP", "源端口号", "目的IP", "目的端口号", "协议号" 这样一个五元组来标识一个通信(可以通过netstat -n查看);

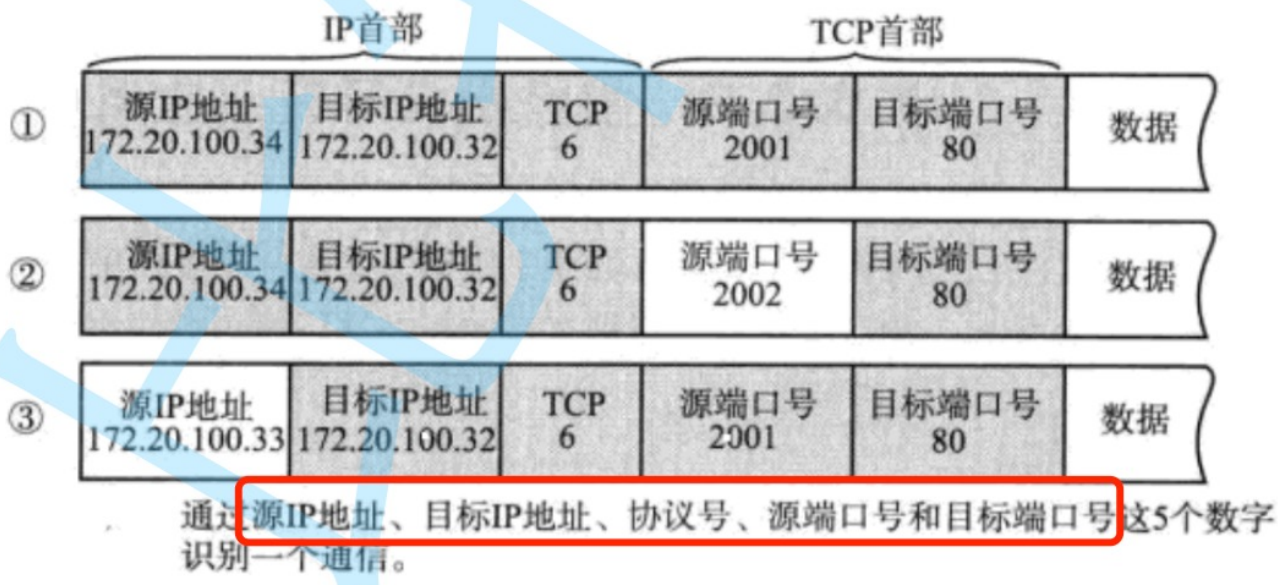


	IP首部			TCP首部		
①	源IP地址 172.20.100.34	目标IP地址 172.20.100.32	TCP 6	源端口号 2001	目标端口号 80	数据
②	源IP地址 172.20.100.34	目标IP地址 172.20.100.32	TCP 6	源端口号 2002	目标端口号 80	数据
③	源IP地址 172.20.100.33	目标IP地址 172.20.100.32	TCP 6	源端口号 2001	目标端口号 80	数据

通过源IP地址、目标IP地址、协议号、源端口号和目标端口号这5个数字识别一个通信。

```
(base) [yufc@ALiCentos7:~]$ netstat -ntlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:40477         0.0.0.0:*               LISTEN      9742/node
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      -
tcp6       0      0 ::1:25                  :::*                    LISTEN      -
tcp6       0      0 :::3306                 :::*                    LISTEN      -
tcp6       0      0 :::111                  :::*                    LISTEN      -
(base) [yufc@ALiCentos7:~]$
```

这里就是一个五元组



- ssh服务器, 使用22端口
- ftp服务器, 使用21端口
- telnet服务器, 使用23端口
- http服务器, 使用80端口
- https服务器, 使用443

ssh给我们提供服务，让我们可以远程登录我们的服务器，很明显，PID，PGID，SID一样，所以它是一个守护进程

```
(base) [yufc@ALiCentos7:~]$ ps axj | head -1 && ps axj | grep sshd
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
    1  1158  1158  1158 ?           -1 Ss       0   0:00 /usr/sbin/sshd -D
 1158  9660  9660  9660 ?           -1 Ss       0   0:00 sshd: yufc [priv]
 9660  9662  9660  9660 ?           -1 S     1000   0:00 sshd: yufc@notty
 9913 10090 10089  9913 pts/5    10089 S+    1000   0:00 grep --color=auto sshd
(base) [yufc@ALiCentos7:~]$
```

netstat

netstat是一个用来查看网络状态的重要工具.

语法：netstat [选项]
功能：查看网络状态
常用选项：

> - n 拒绝显示别名，能显示数字的全部转化成数字
> - l 仅列出有在 Listen (监听) 的服務状态
> - p 显示建立相关链接的程序名
> - t (tcp)仅显示tcp相关选项
> - u (udp)仅显示udp相关选项
> - a (all)显示所有选项，默认不显示LISTEN相关

pidof

在查看服务器的进程id时非常方便.

语法：pidof [进程名]
功能：通过进程名, 查看进程id

```
(base) [yufc@ALiCentos7:~/Src/BitCodeField/0226/http_demo]$ ./HttpServer 8080
[NORMAL] [1683616663] create socket success, sock: 3
[NORMAL] [1683616663] init TcpServer Success
```

我们先启动一个服务

```
(base) [yufc@ALiCentos7:~]$ pidof HttpServer
10222
(base) [yufc@ALiCentos7:~]$
```

用pidof指令可以查到我们刚刚启动的服务的pid

```
(base) [yufc@ALiCentos7:~]$ pidof HttpServer | xargs kill -9
(base) [yufc@ALiCentos7:~]$
```

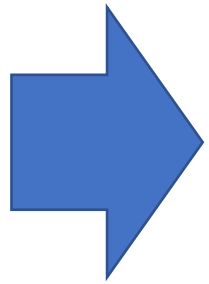
用这个命令，可以杀掉我们刚才启动的服务

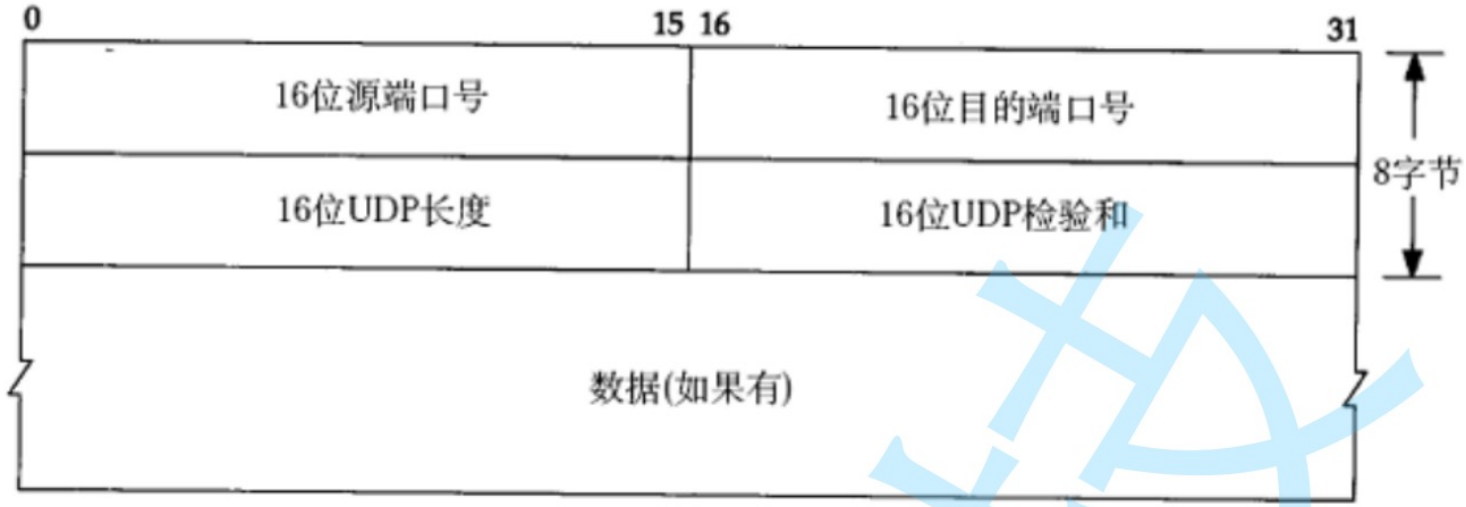
Udp

几乎任何协议都要首先解决两个问题

1. 如何分离（封装）
2. 如何交付

这是一个udp的报文





- 16位UDP长度, 表示整个数据报(UDP首部+UDP数据)的最大长度;
- 如果校验和出错, 就会直接丢弃;

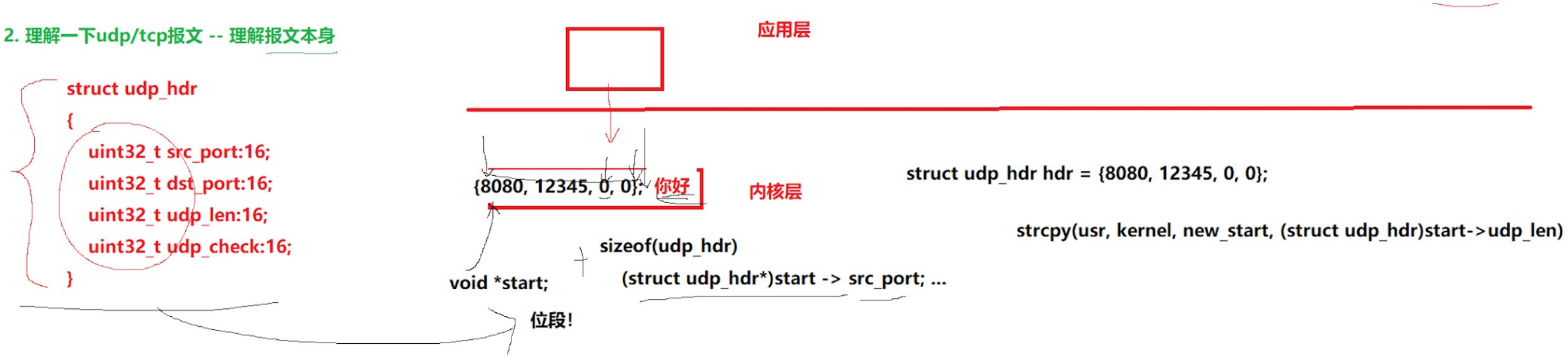
几乎任何协议都要首先解决两个问题

1. 如何分离（封装）
2. 如何交付

Udp是固定长度的报头！

有了固定的报头，那么将报头和有效载荷分离就可以容易做到
然后，根据报头中的16位端口号，进行向上交付就行了！

udp是具有将报文一个一个接收的能力的！
所以->udp是面向数据报的！



面向数据报

应用层交给UDP多长的报文, UDP原样发送, 既不会拆分, 也不会合并;

用UDP传输100个字节的数据:

> - 如果发送端调用一次sendto, 发送100个字节, 那么接收端也必须调用对应的一次recvfrom, 接收100个字节; 而不能循环调用10次recvfrom, 每次接收10个字节;

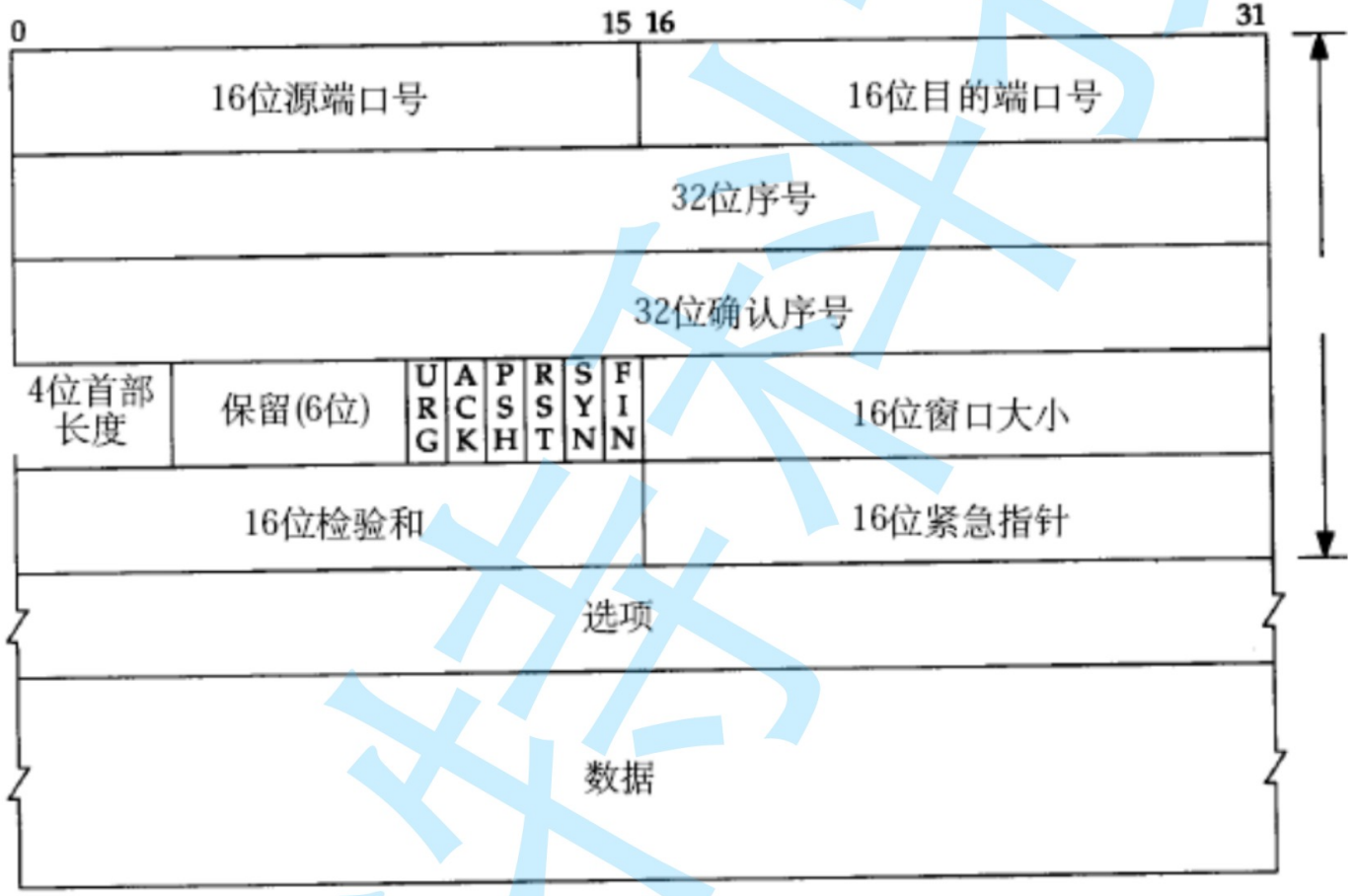
UDP的缓冲区

> - UDP没有真正意义上的 **发送缓冲区**. 调用sendto会直接交给内核, 由内核将数据传给网络层协议进行后续的传输动作;
> - UDP具有接收缓冲区. 但是这个接收缓冲区不能保证收到的UDP报的顺序和发送UDP报的顺序一致; 如果缓冲区满了, 再到达的UDP数据就会被丢弃;

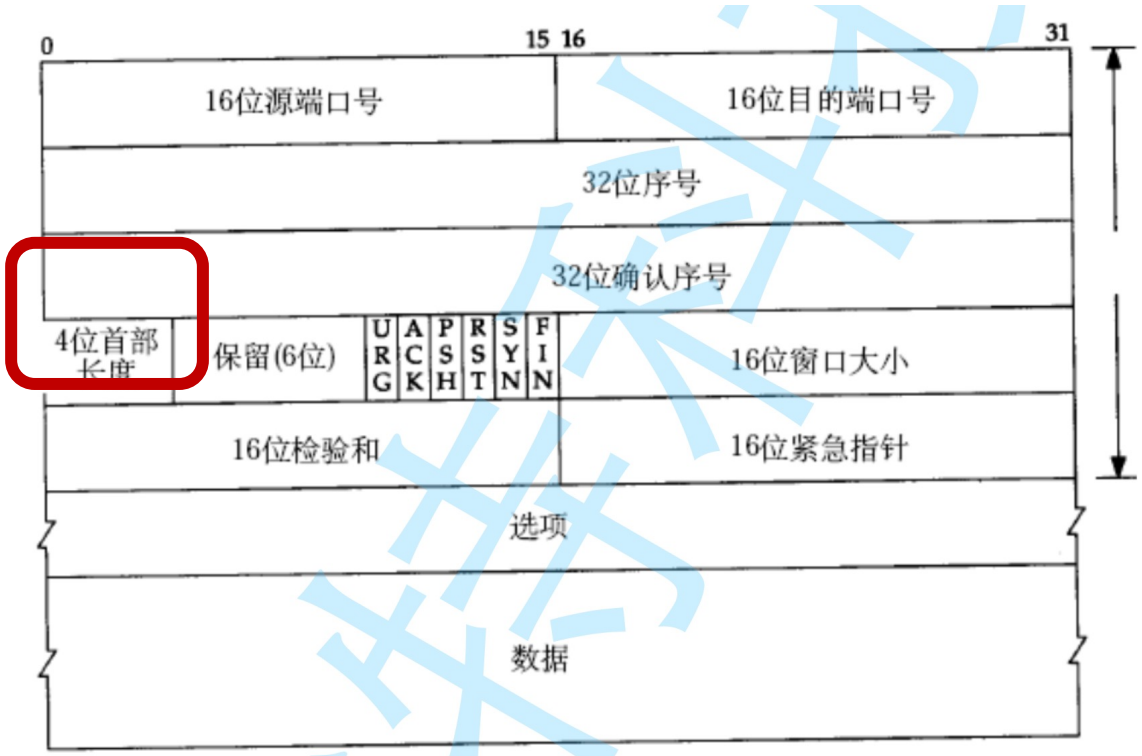
UDP的socket既能读, 也能写, 这个概念叫做 **全双工**

TCP

1. 如何解包？如何向上交付
2. 如何理解tcp报头
3. 挑一个重要的可靠性属性，作为切入点，理解什么叫做可靠
4. 系统学习tcp报头
5. tcp可靠性的其他策略



tcp是变长报头！有20个字节是固定的，然后有选项！



TCP是如何解包的?

1. 提取20字节

2. 根据标准报头，提取4位首部长度 * 4 =20 - done

3. 读取[提取4位首部长度 * 4 - 20]字节数据，选项

4. 读完了报头，剩下的都是有效载荷

[20, 60]

x * 4 = 20 => x = 5

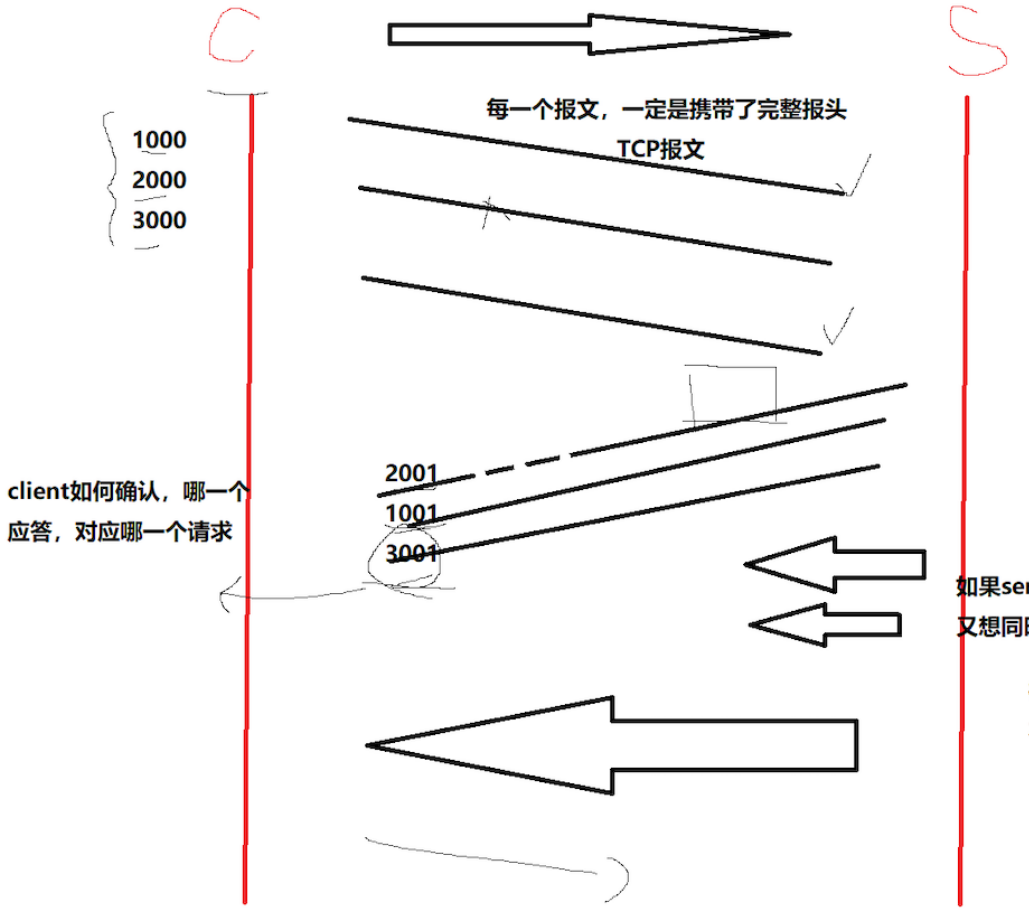
x * 4 = 60 => x = 15

[5, 15] => [0101, 1111]

这个4位首部长度，单位是4字节！
所有0101代表5*4=20字节
代表没有选项

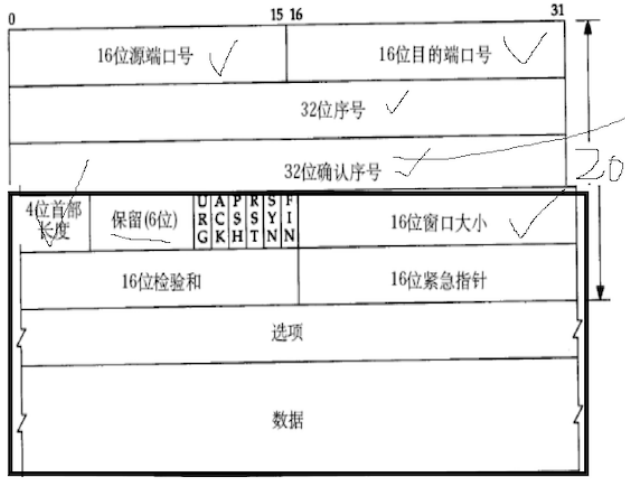
所以选项最多有40个字节

这个4位首部长度的范围就是[0101, 1111]



TCP是全双工的！，任何一方，既可以收，又可以发！

client一次可能向服务端发送多个报文，就有一个问题，发送的顺序，不一定是接受顺序



表示，确认序号对应的数字，之前的所有的报文已经全部收到了！
告诉对方，下次发送，从确认序号指明的序号发送

序号和确认序号：

1. 将请求和应答进行一一对应
2. 确认序号，表示的含义：确认序号之前的数据已经全部收到
3. 允许部分确认丢失，或者不给应答
4. 为什么要有两个字段数字？任何通信的一方，工作方式都是全双工的，在发送确认的时候，也可能携带新的数据
5. 1000 2000 3000 -》 2000 1000 3000 -》 乱序 因为任何一方都会收到报文，报文中会携带序号