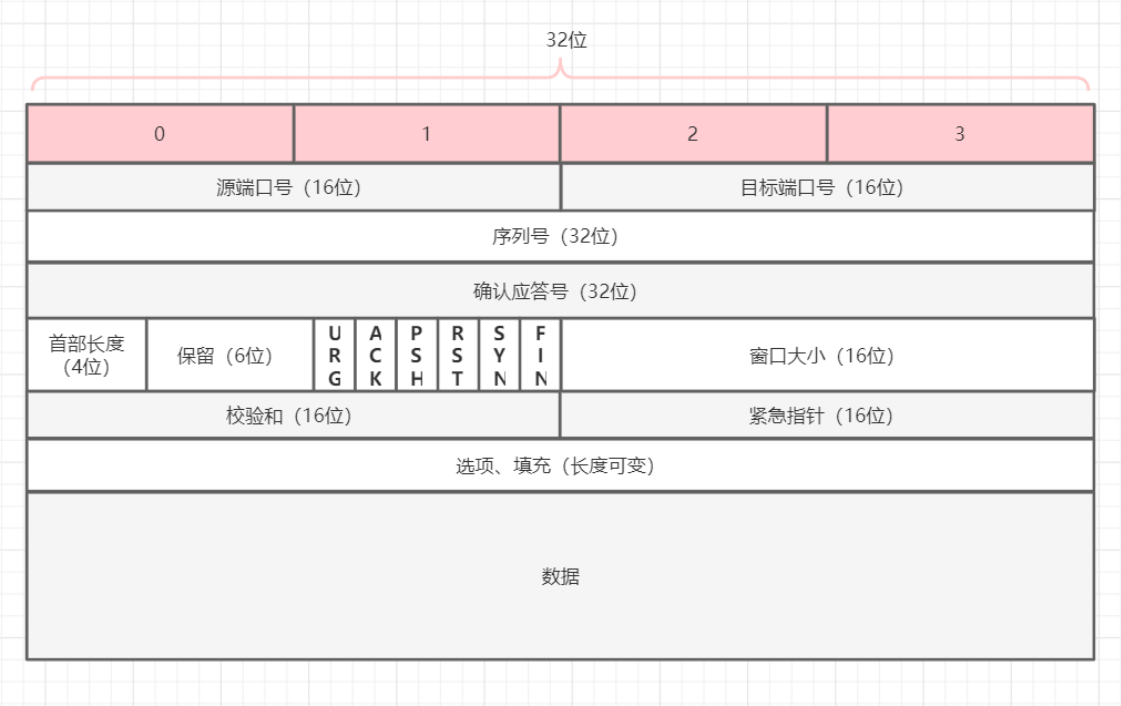
TCP虽然是面向字节流的，但TCP传送的数据单元却是报文段。一个TCP报文段分为首部和数据两部分，而TCP的全部功能体现在它首部中的各字段的作用。TCP首部比UDP复杂得多，完整的TCP头部如下图所示：

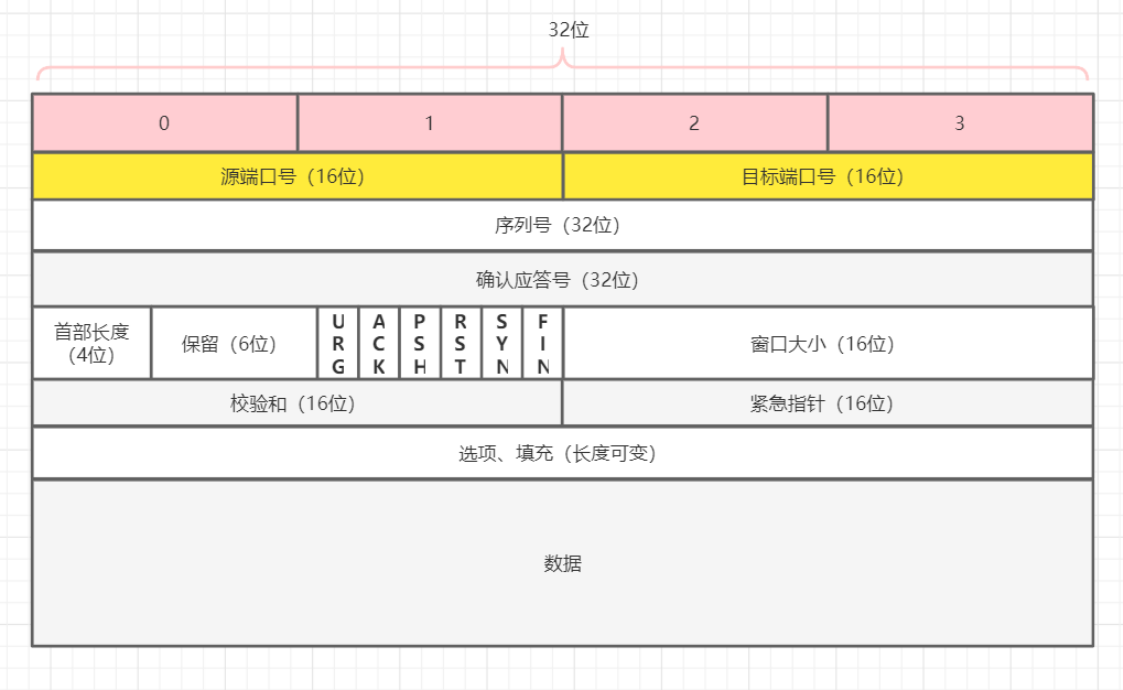


TCP首部类似于IP首部，前20个字节是固定的，后面可选长度最大为40字节，可以根据下面要说明的首部长度字段中计算出来，因此TCP报文段首部长度最小是20字节、最大是60字节，我们核心关注的是前20个字节的固定首部。

一、源端口号和目标端口号

黄色高亮的字段就是首部中第一个要看的字段：端口号，分为源端口号（Src Port）和目标端口号（Dst Port），分别占2个字节长度，即一台主机最大允许65536个端口号。

- 源端口号用来标识发送该TCP报文段的应用进程
- 目标端口号用来标识接收该TCP报文段的应用进程

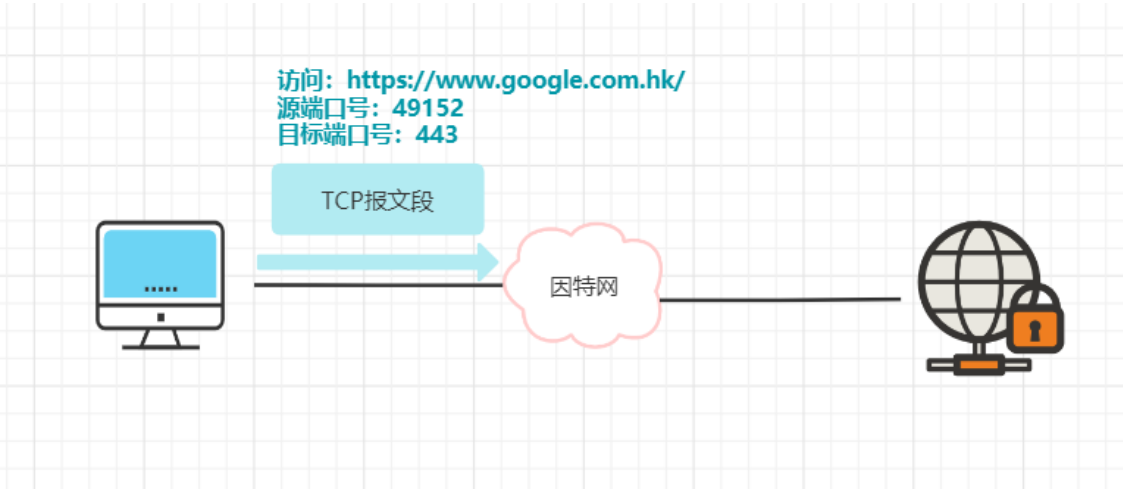


在TCP报文首部中，没有源IP地址和目标IP地址，因为那是IP层协议的事情，只有源端口号和目标端口号。

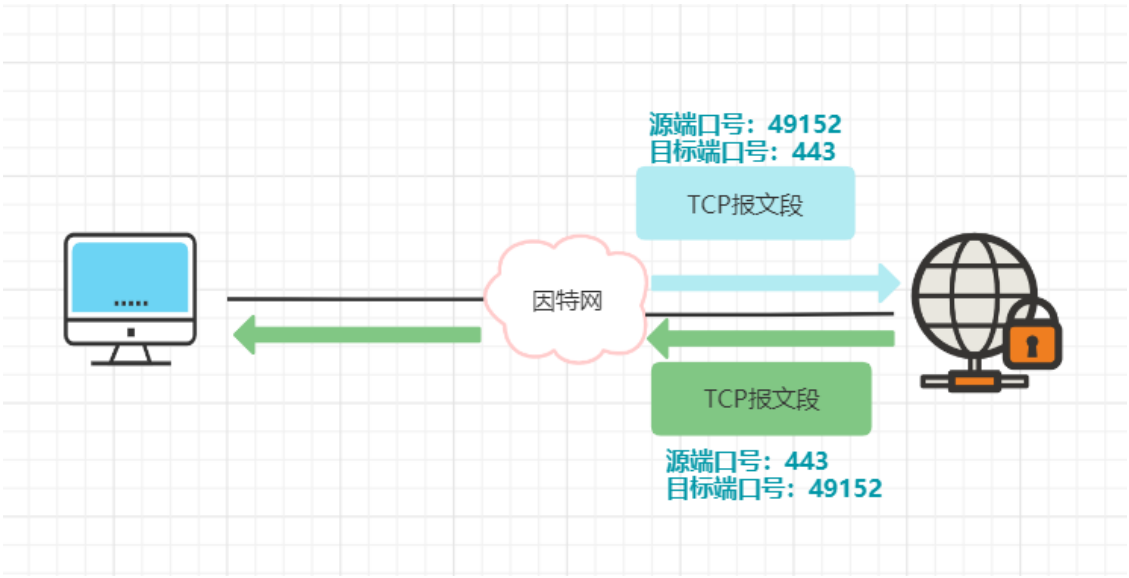
源IP、源端口、目标IP、目标端口构成了TCP连接的「四元组」。一个四元组可以唯一标识一个连接。

下面举例说明源端口号和目标端口号的作用，假设主机中的浏览器进程要访问Web服务器中的Web进程，忽略所有其他细节，仅考虑端口号。

当在浏览器地址栏中输入了 `https://www.google.com.hk/` 域名后，浏览器会构建一个封装有HTTP请求报文的TCP报文段（这里为了聚焦问题就不提及网络层的IP数据报首部和MAC帧首部了），该报文段首部中的源端口字段会填写一个临时端口号例如49152，用来标识发送该报文段的浏览器进程，目标端口号是熟知端口号443端口，因为使用HTTPS协议的Web进程默认监听该端口号。



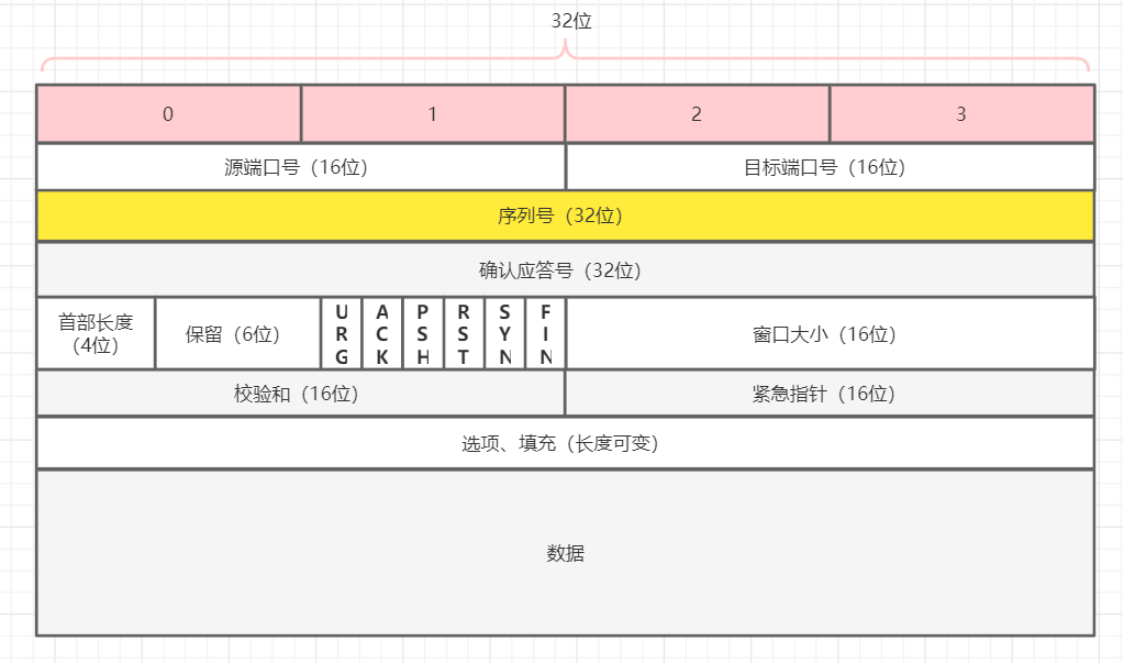
Web服务器收到该TCP报文段后，从中解封出HTTP请求报文，并根据TCP报文段首部中的目标端口号的值443，将HTTP请求报文上交给Web服务器进程，Web服务器进程根据HTTP请求报文的内容进行相应处理，并构建一个HTTP响应报文，HTTP响应报文同样需要封装成TCP报文段进行发送，该报文段首部中的源端口号为443，目标端口号为49152。



访问网站的主机收到TCP报文段后，从中解封出HTTP响应报文，并根据TCP报文段首部中的目的端口值49152，将HTTP响应报文上交给浏览器进程，浏览器根据HTTP响应报文进行解析并显示。

二、序列号

下面一个字段来看序列号字段。



我们需要对每个网络包进行编号来解决网络包乱序、重复的问题，以保证数据包以正确的顺序组装传递给上层应用。

如果发送方发送的是四个报文序列号分别是1、2、3、4，但到达接收方的顺序是2、4、3、1，接收方就可以通过序列号的大小顺序组装出原始的数据。

TCP是面向字节流的协议，通过TCP传输的字节流的每个字节都分配了序列号，序列号（Sequence number）指的是本TCP报文段数据载荷的第一个字节的序号。

我们来举例说明下，如下图所示，有三个TCP报文段，每个报文段的序列号就是数据载荷部分的第一个字节的序号。（请注意，9、10、11等这些数字是字节数据的序号而不是内容）

序列号占用32位，序号增加到最后一个后，下一个序号就又回到0。

三、初始序列号ISN

接着序列号继续说下初始序列号。

建立TCP连接后的第一个报文序列号我们称为初始序列号ISN。

每个新的TCP连接中，ISN不会从0或1开始，要求ISN是随机生成的，通过SYN包传给接收端主机，为的就是让攻击者更难以猜测序列号sequence number，因为伪造的序列号sequence number不在合法范围内，而被接收方丢弃，增加安全性。

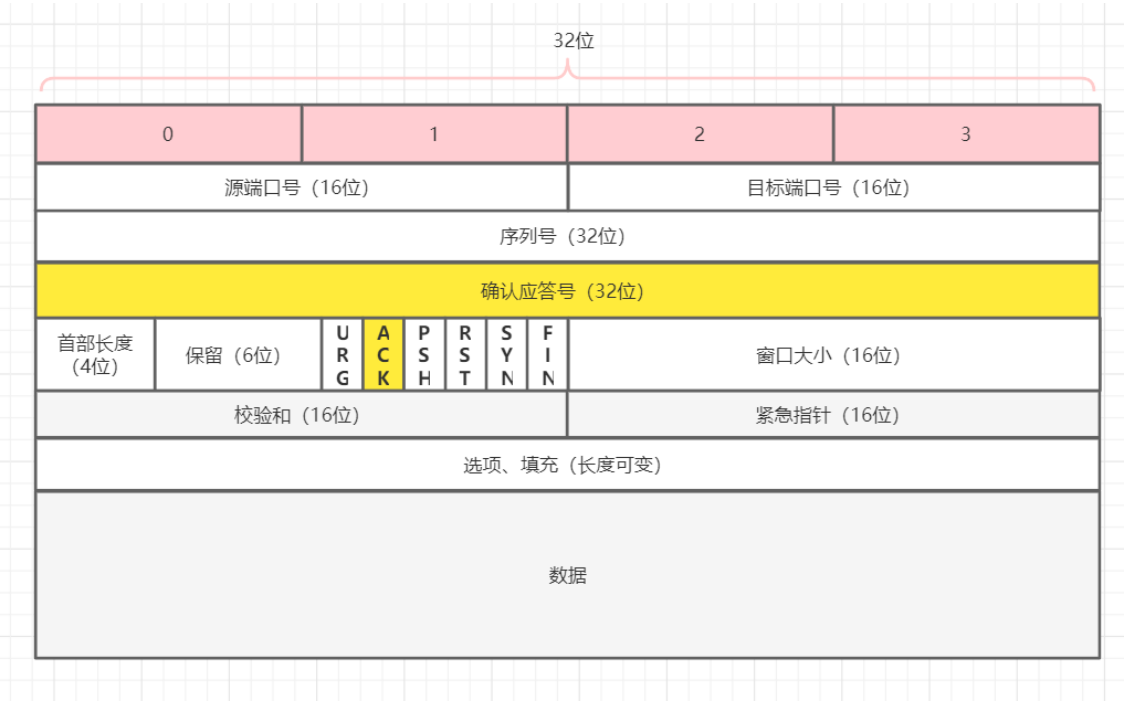
如果ISN被知晓会发生什么？假设A和B之间有一个TCP连接，C想搞破坏比如发送一个RST中断A和B的连接，C可以伪造一个合法TCP报文，最关键是TCP字段里的sequence number 、acknowledged number，只要这两项位于接收者滑动窗口内，就是合法的，对方可以接收，C可以伪造A的IP给B发一个Reset报文释放A和B之间的连接。

当然，这里的随机也不是完全随机，而是随着时间流逝而线性增长，到了2^32尽头再回滚。并且在Windows的不同版本，或者Linux的不同版本，这个随机的算法都不太一样。

RFC793 提到初始化序列号 ISN 随机生成算法：ISN = M + F(localhost, localport, remotehost, remoteport)。

- M 是一个计时器，这个计时器每隔 4 微秒加 1。
- F 是一个 Hash 算法，根据源 IP、目的 IP、源端口、目的端口生成一个随机数值。要保证 Hash 算法不能被外部轻易推算得出，用 MD5 算法是一个比较好的选择。

四、确认号和确认标志位ACK



指出期望收到对方下一个TCP报文段的数据载荷的第一个字节的序号，同时也是对之前收到的所有数据的确认。

若确认号等于N，则表明到序号N-1为止的所有数据都已正确接收，期望接收序号为N的数据。

请注意，只有当确认标志位ACK取值为1时，确认号字段才有效，取值为0时确认号字段无效。

不过TCP也规定了，在连接建立完成后，所有传送的TCP报文段都必须把ACK置为1。

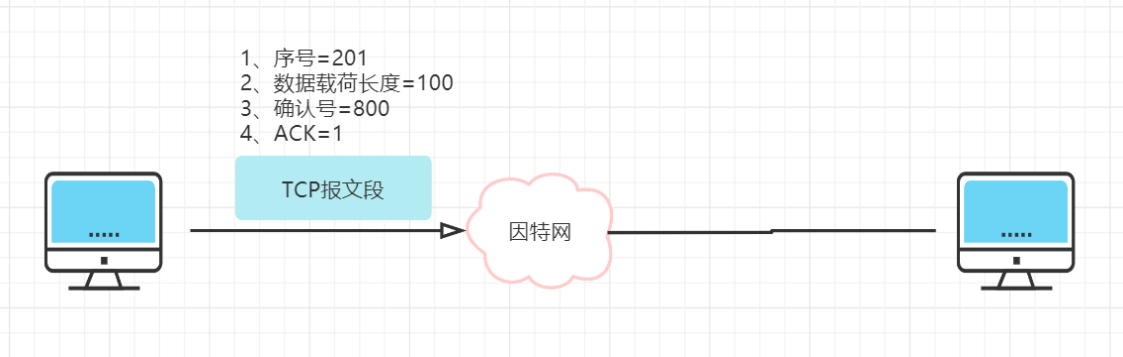
确认号含义很简单，不过也是有几个注意点的：

- 不是所有的包都需要确认的
- 不是收到了数据包就立马需要确认的，可以延迟一会再确认
- ACK 包本身不需要被确认，否则就会无穷无尽死循环了
- 确认号永远是表示小于此确认号的字节都已经收到

五、序列号和确认号如何一起工作

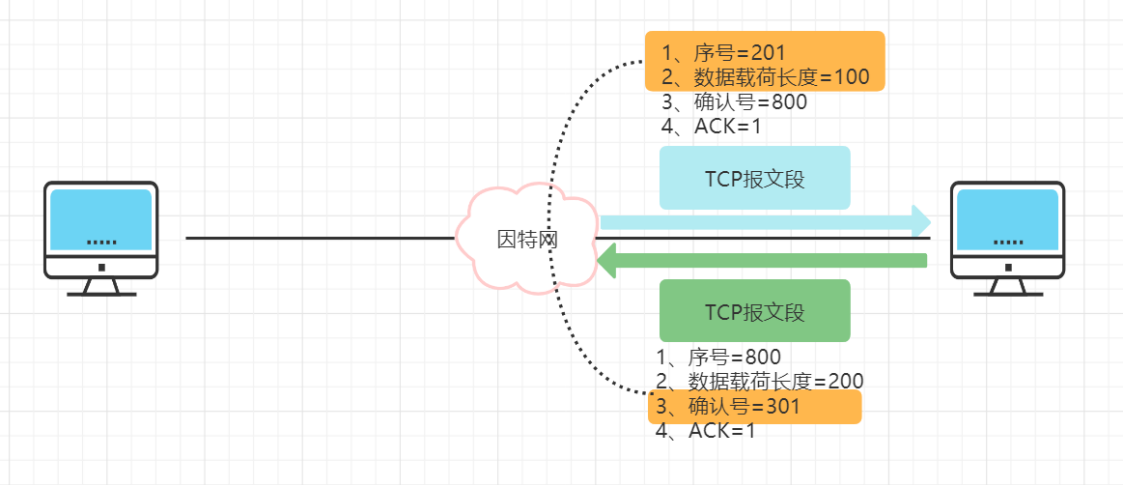
假设现在有一个TCP客户进程发送一个TCP报文段，该报文段首部中的序号字段取值为201，这表示该TCP报文段数据载荷的第一个字节的序号为201，假设数据载荷的长度为100字节。

首部中确认号字段的取值为800，这表示TCP客户端进程收到了TCP服务器进程发来的序号到799为止的全部数据，现在期望收到序号从800开始的数据，为了使确认号有效，首部中的确认标志位ACK的值必须为1。

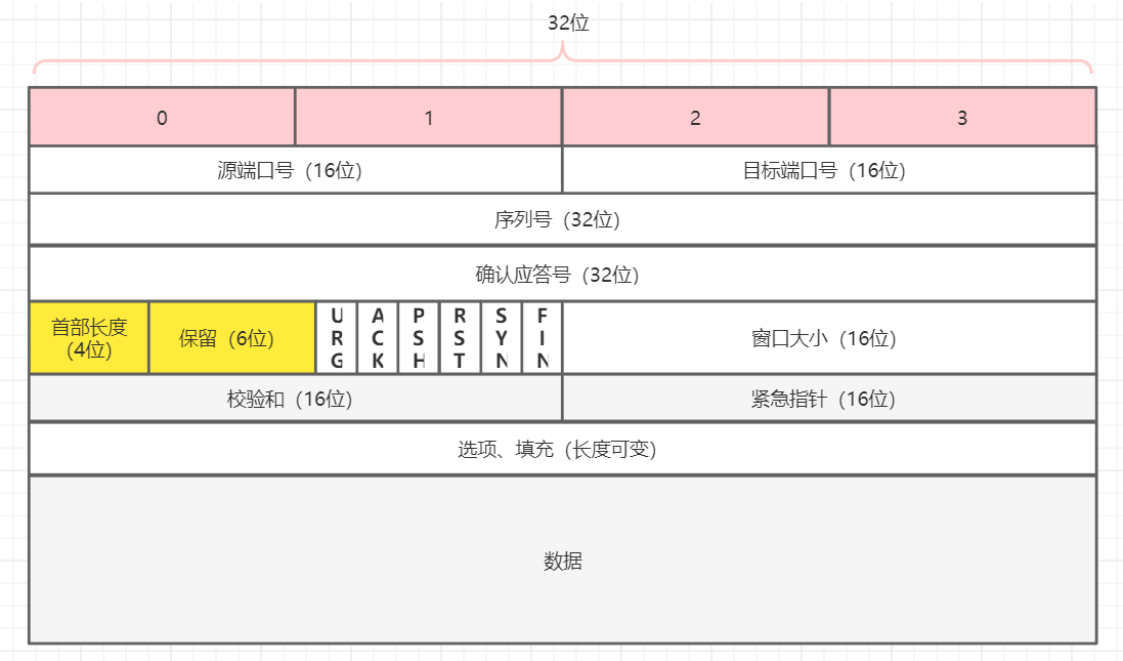


TCP服务器进程收到该报文段后，也给TCP客户端进程发送TCP报文段，该数据报首部中的序号值为800，这表示该TCP报文段数据载荷的第一个字节的序号为800，正好与TCP客户进程的确认相匹配。

首部中确认号取值假设为301，这表示TCP服务器进程收到了TCP客户进程发来的序号到300为止的全部数据，现在期望收到序号从301开始的数据。



六、首部长度字段、保留字段



首部长度往往也被称为数据偏移，占4个比特，并且单位是4字节。

用来指出TCP报文段的数据载荷部分的起始处距离TCP报文段的起始处有多远。

换句话说，实际上是指出了TCP报文段的首部长度。

一开始我们就说过，TCP的首部分为固定首部部分和可变首部部分，固定首部长度为20字节，因此TCP报文段首部长度最小值为0101(二进制)，对应十进制的值是5，再乘以单位4字节，即20字节。

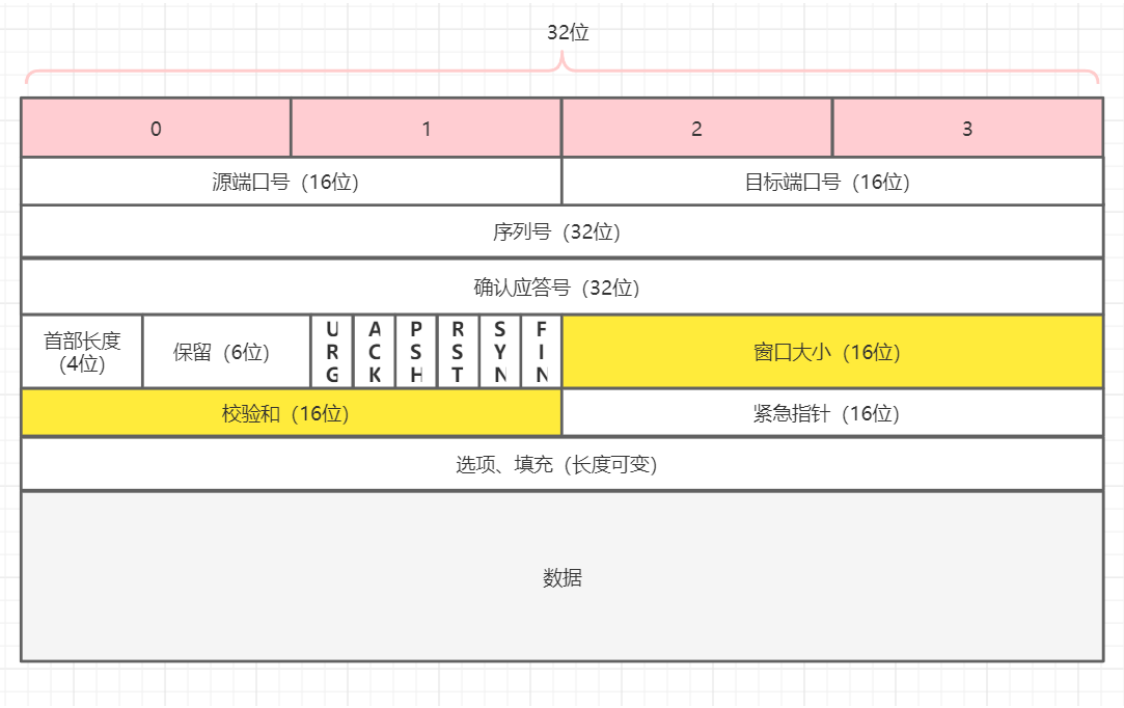
加上最大可选首部40字节，TCP报文段首部最大长度一共为60字节，那么TCP报文段首部首部长度最大值应该是1111(二进制)。

总之，TCP的首部长度一定是4字节的整数倍，不够则需要进行长度填充。

下面一个字段是保留字段，占用6比特，保留为进后使用，目前应置为0。

七、窗口大小和校验和字段

再来看看窗口大小字段和校验和字段。



窗口大小占16比特，以字节为单位，指出发送本报文段这一方的接收窗口。

接收方参考此接收窗口，可动态调整其能力的大小，这种以接收方的接收能力来控制发送方的发送能力，称为流量控制，后续我们将针对流量控制进行详细说明。

我们看到，窗口大小只有16比特，可能 TCP 协议设计者们认为 16 位的窗口大小已经够用了，也就是最大窗口大小是 65535 字节（64KB），不过显然对于如今的网速来说已经不能满足要求了。

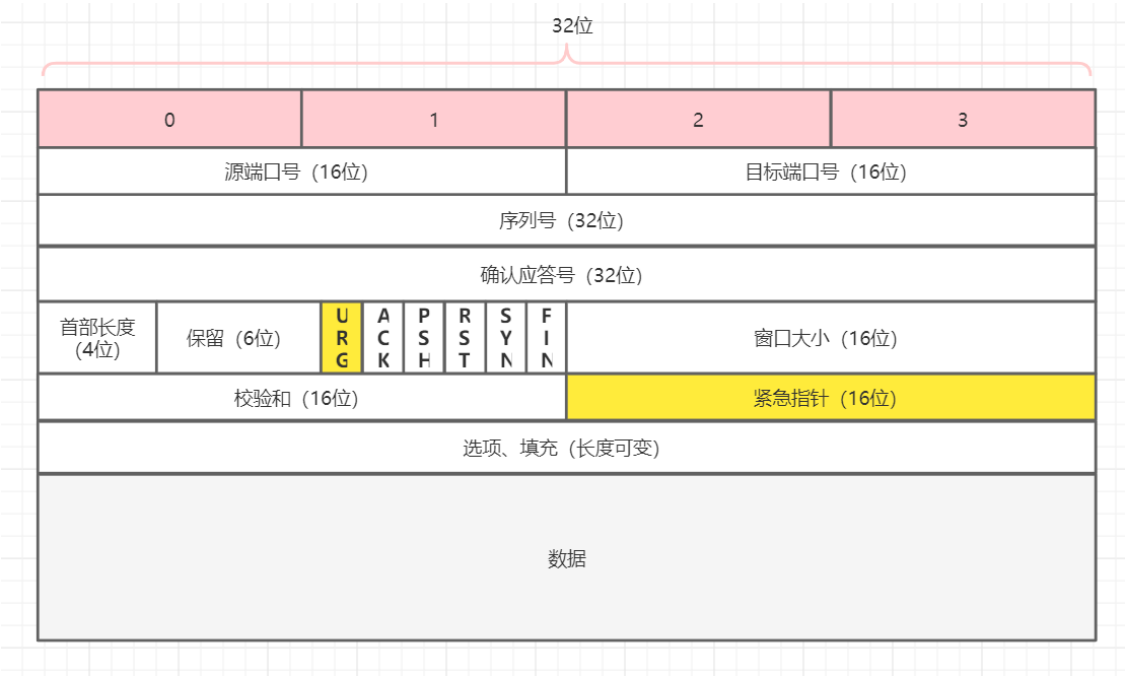
自己挖的坑当然要自己填，因此TCP 协议引入了「TCP 窗口缩放」选项 作为窗口缩放的比例因子，比例因子值的范围是 0 ~ 14，其中最小值 0 表示不缩放，最大值 14。比例因子可以将窗口扩大到原来的 2 的 n 次方，比如窗口大小缩放前为 1050，缩放因子为 7，则真正的窗口大小为 1050 * 128 = 134400。

关于「TCP 窗口缩放」选项是在下面要介绍的选项字段中体现，值得注意的是，窗口缩放值在三次握手的时候指定。

校验和字段占用16比特，检查范围包括TCP报文段首部和数据载荷两部分，关于校验和字段我们不展开说明，感兴趣可以详细去了解下，只用知晓其字段就是用来检查TCP报文段在传输过程中是否出现了误码。

八、紧急指针字段和紧急标志位URG

接着说说紧急指针字段和紧急标志位URG。



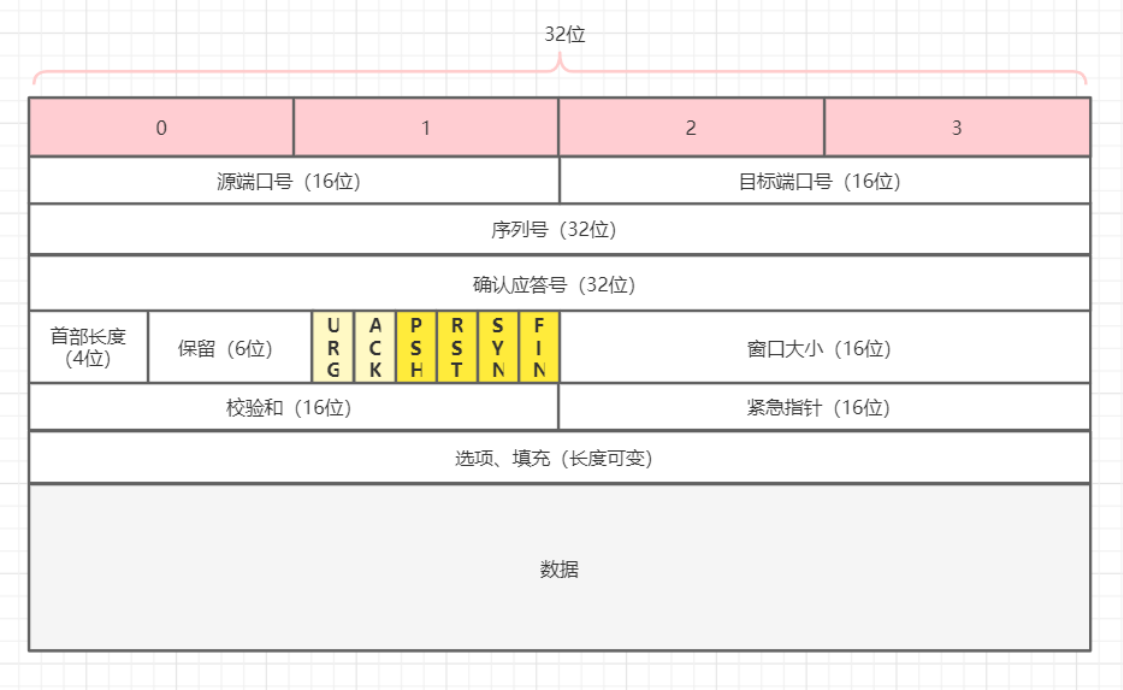
紧急指针：占16比特，以字节为单位，用来指明紧急数据的长度。

紧急标志位URG：取值为1时紧急指针字段有效；取值为0时紧急指针字段无效。

当发送方有紧急数据时，可将紧急数据插队到发送缓存的最前面，并立刻封装到一个TCP报文段中进行发送。紧急指针会指出本报文段数据载荷部分包含了多长的紧急数据，紧急数据之后是普通数据。

接收方收到紧急标志位为1的报文段，会按照紧急字段的值，从报文段数据载荷部分取出紧急数据，并直接上交应用进程，而不必在接收缓存中排队。

九、其他标志位



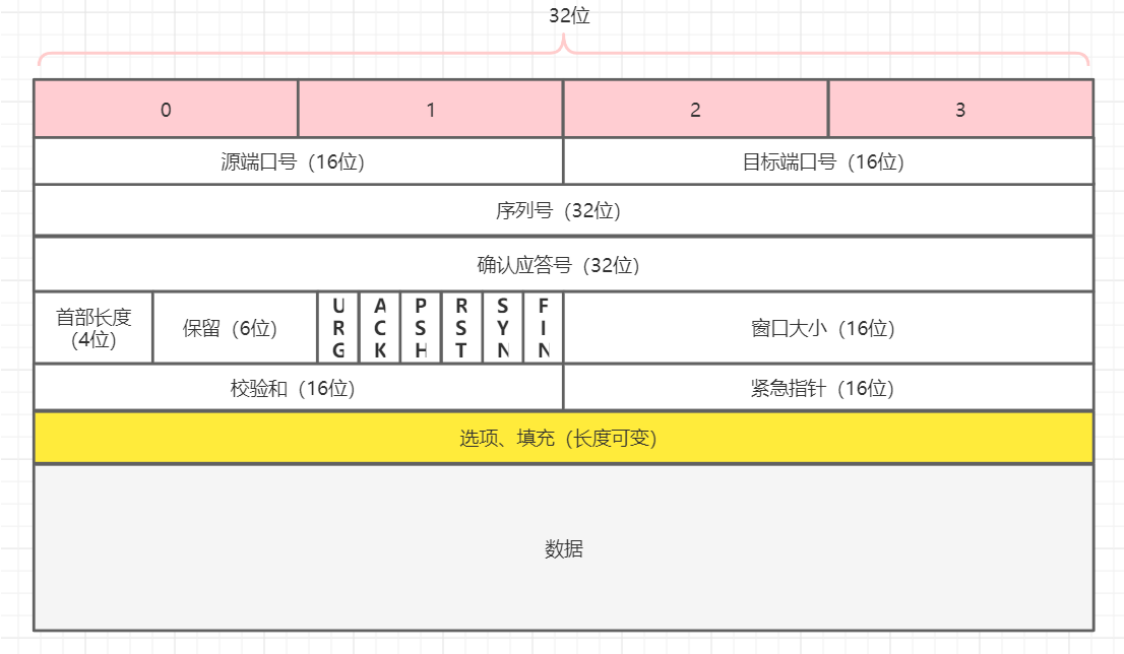
实际上应该有8个标记位，但是我们核心关心的是标黄的6个标记位，每个标记位占用1比特，URG和ACK字段已说明，剩下四个标记位：

- PSH：告知对方这些数据包收到以后应该马上交给上层应用，不必等到接收缓存都填满后再向上交付
- RST：表明TCP连接出现了异常，必须释放连接，再重新建立连接；RST置1还可用来拒绝一个一个非发报文或拒绝打开一个TCP连接
- SYN：用于发起连接数据包同步双方的初始序列号
- FIN：通知对方我发完了所有数据，准备断开连接，后面我不会再发数据包给你了

当某个字段生效时，只需要将其置为1，否则置为0表示失效，这些标记可以组合使用，比如SYN+ACK、FIN+ACK等。

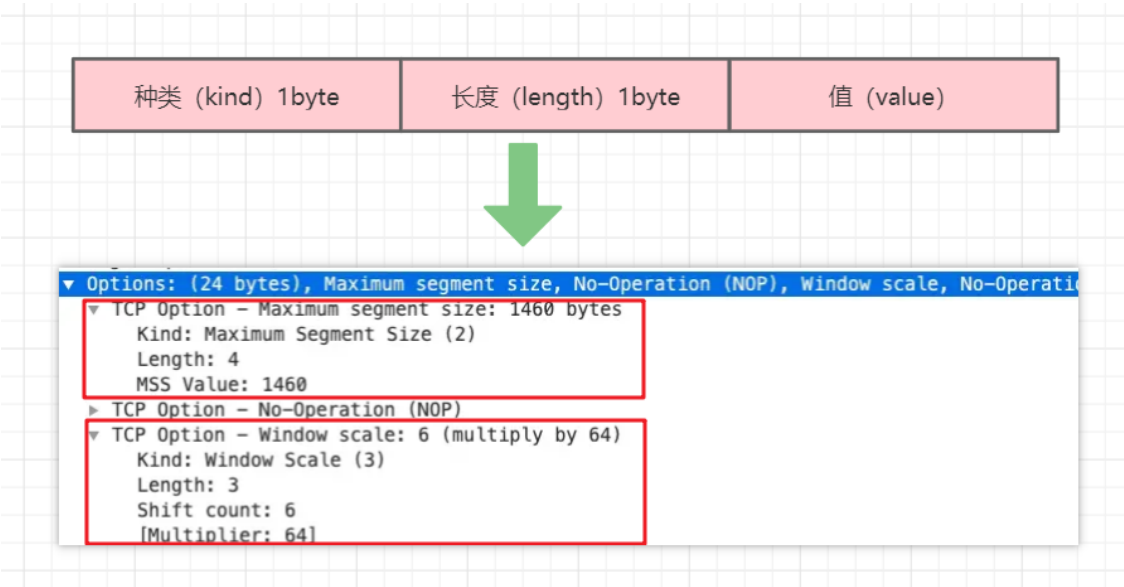
后续的三次握手和四次挥手过程中将会用到这些标记位，现在我们暂时先有个印象即可。

十、选项字段和填充字段



TCP报文段首部中，除了20字节固定首部外，还有最大40字节的选项部分，增加选项可以增加TCP功能。

可选项的格式如下所示：



比较常见的类型有：

- 最大报文段长度MSS选项：TCP报文段数据载荷部分的最大长度
- 窗口扩大选项：扩大窗口，提高吞吐量，TCP首部窗口字段只有16比特，因此一次最大只能发送64K字节的数据（2^16 个字节，对应到KB单位则为2^6KB，即64KB），如果采用了该选项，窗口的最大值可扩展到1G字节。
- ...

后续文章还会探讨到，这里也不展开说明了。

填充字段：由于选项字段的长度可变，因此使用填充来确保TCP报文段首部能被4整数，这个是由于TCP首部长度字段是以4字节为单位的。

对以上内容可能还存疑，不过具备整体印象即可，因为后面的内容都将跟这里的首部字段息息相关，从整体到局部切入我认为是比较好的一种学习方式，本文完、下篇见。