决胜网络协议

TCP 协议的首部字段

张师傅

TCP 协议的首部字段

<table>
<tr><td colspan="5">0</td><td colspan="5">1</td><td colspan="5">2</td><td>3</td></tr>
<tr><td colspan="10">源端口（Source port）</td><td colspan="5">目标端口（Destination port）</td></tr>
<tr><td colspan="15">序列号（Sequence number）</td></tr>
<tr><td colspan="15">确认号（Acknowledgment number）</td></tr>
<tr><td colspan="3">头部长度</td><td colspan="3">保留</td><td>NS</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="5">窗口大小（Window Size）</td></tr>
<tr><td colspan="10">校验和（Checksum）</td><td colspan="5">紧急指针（Urgent pointer）</td></tr>
<tr><td colspan="15">选项（Options）、填充（Padding）</td></tr>
</table>

初学 wireshark 抓包

"tcp.addr == 172.19.214.24"

VS

"ip.addr == 172.19.214.24"

源端口号、目标端口号

<table>
<tr><td colspan="4">0</td><td colspan="4">1</td><td colspan="2">2</td><td colspan="2">3</td></tr>
<tr><td colspan="8">源端口（Source port）</td><td colspan="4">目标端口（Destination port）</td></tr>
<tr><td colspan="12">序列号（Sequence number）</td></tr>
<tr><td colspan="12">确认号（Acknowledgment number）</td></tr>
<tr><td colspan="2">头部长度</td><td colspan="2">保留</td><td>NS</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">窗口大小（Window Size）</td></tr>
<tr><td colspan="8">校验和（Checksum）</td><td colspan="4">紧急指针（Urgent pointer）</td></tr>
<tr><td colspan="12">选项（Options）、填充（Padding）</td></tr>
</table>

端口号 65535 个应该够了吧？

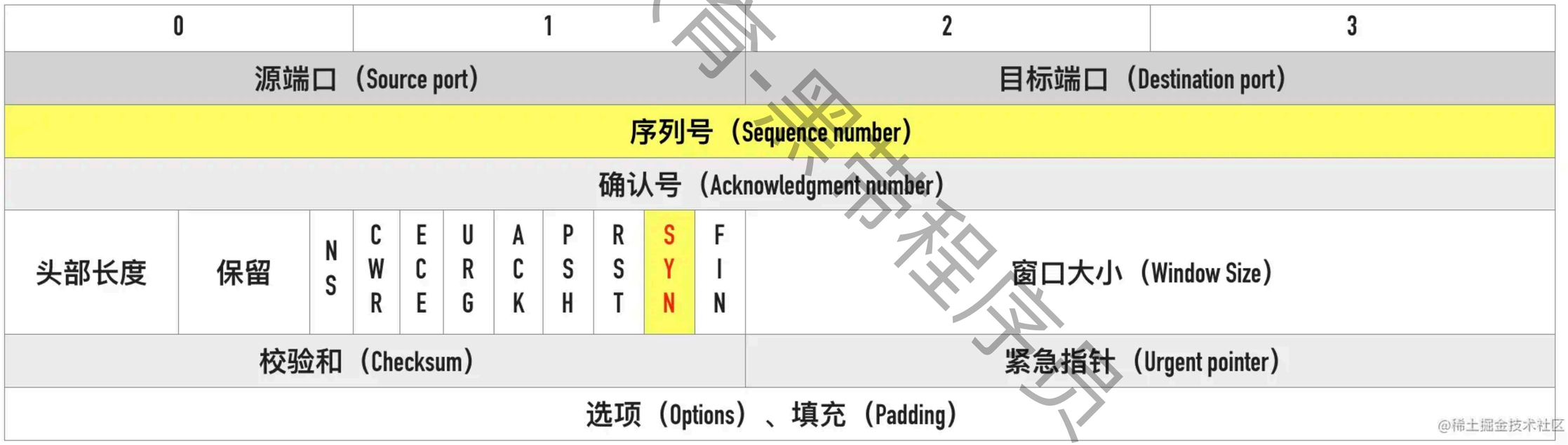
序列号（Sequence number）

TCP 是一种字节流（byte-stream）协议，流的含义是没有固定的报文边界。

序列号（Sequence number）

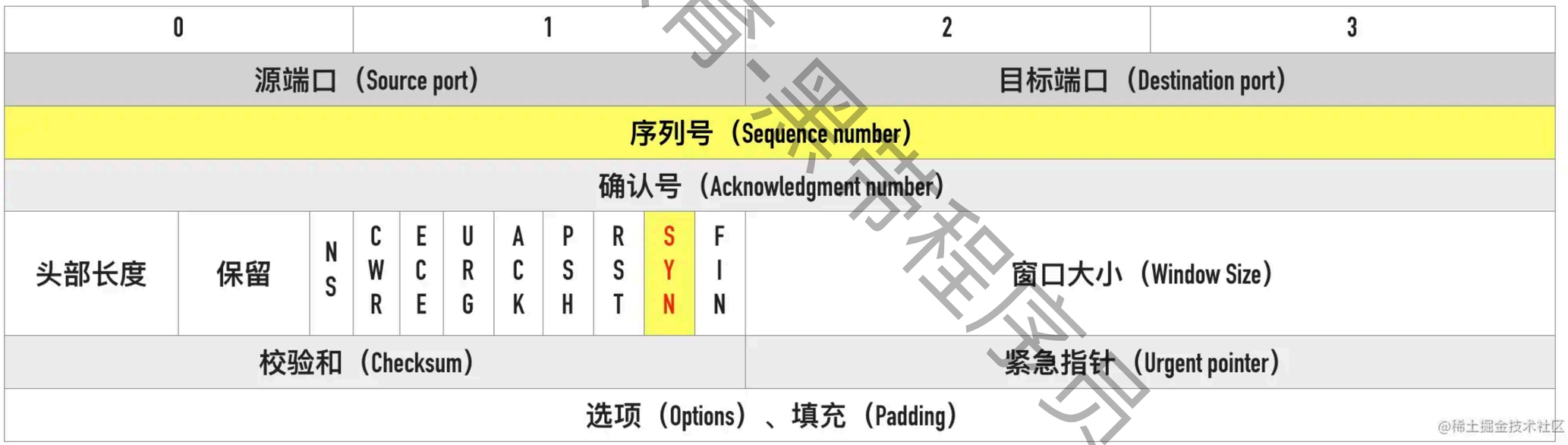
TCP 是一种字节流（byte-stream）协议，流的含义是没有固定的报文边界。

序列号是一个 32 位的无符号整数，达到 2^32-1 后循环到 0

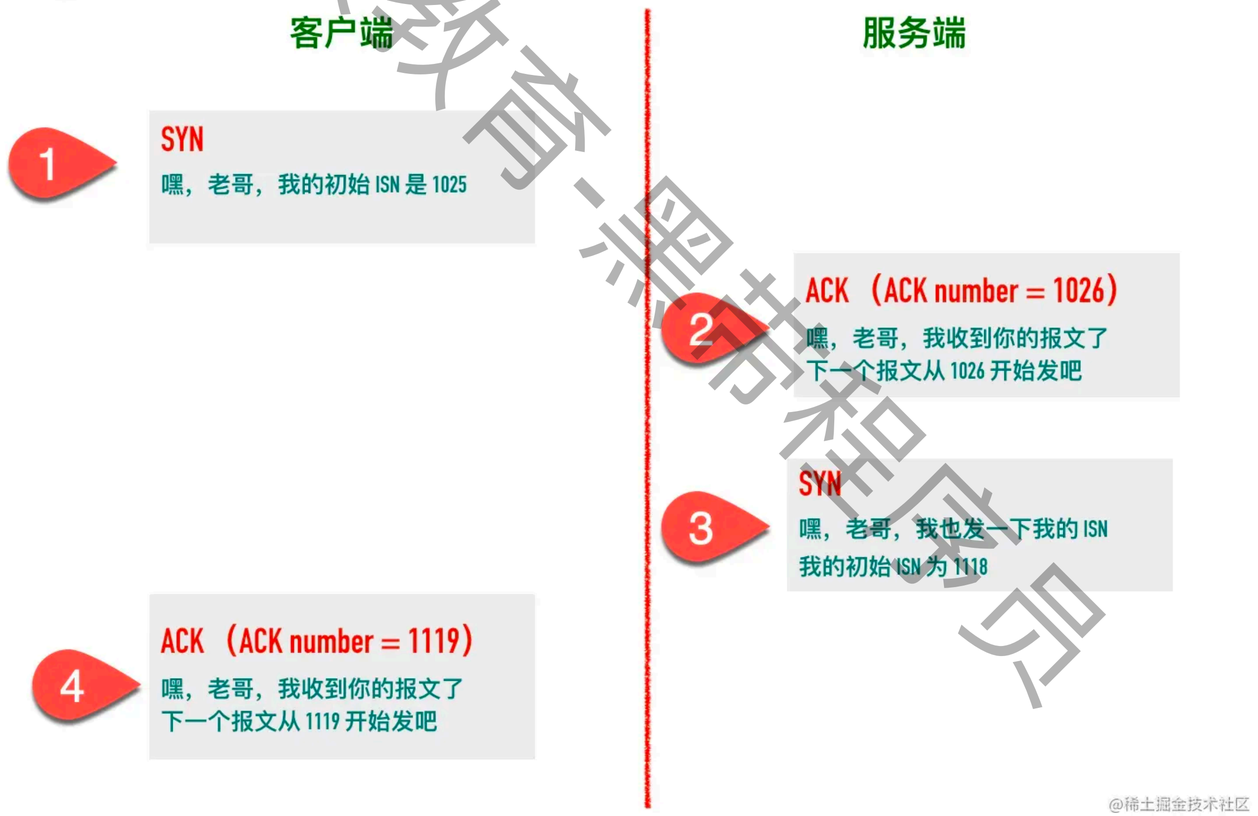


初始序列号（Initial Sequence Number, ISN）

在建立连接之初，通信双方都会各自选择一个序列号，称之为初始序列号



三次握手的 ISN 交换



初始序列号的生成算法

```
__u32 secure_tcp_sequence_number(__be32 saddr, __be32 daddr,
                                 __be16 sport, __be16 dport)
{
    u32 hash[MD5_DIGEST_WORDS];

    net_secret_init();
    hash[0] = (__force u32)saddr;
    hash[1] = (__force u32)daddr;
    hash[2] = ((__force u16)sport << 16) + (__force u16)dport;
    hash[3] = net_secret[15];

    md5_transform(hash, net_secret);

    return seq_scale(hash[0]);
}

static u32 seq_scale(u32 seq)
{
    return seq + (ktime_to_ns(ktime_get_real()) >> 6);
}
```

序列号回绕

sequence wraparound

```
static inline bool before(__u32 seq1, __u32 seq2)
{
        return (__s32)(seq1-seq2) < 0;
}
```

```
seq1 = 255, seq2 = 1
seq1 = 1111 1111
seq2 = 0000 0001
seq1 - seq2 = 1111 1110 < 0 --> seq1 < seq2

seq2 = 128
seq1 = 1111 1111
seq2 = 1000 0000
seq1 - seq2 = 0111 1111 > 0 --> seq1 > seq2
```

确认号

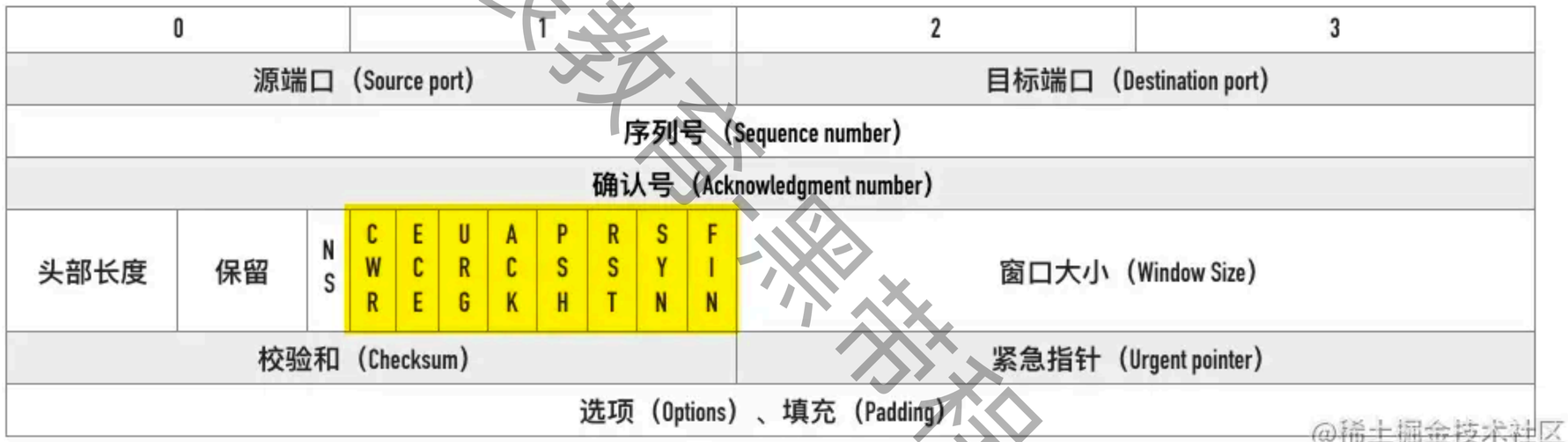
Acknowledgment number

告知对方下一个期望接收的序列号，小于此确认号的所有字节都已经收到。

一些唠叨

- 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

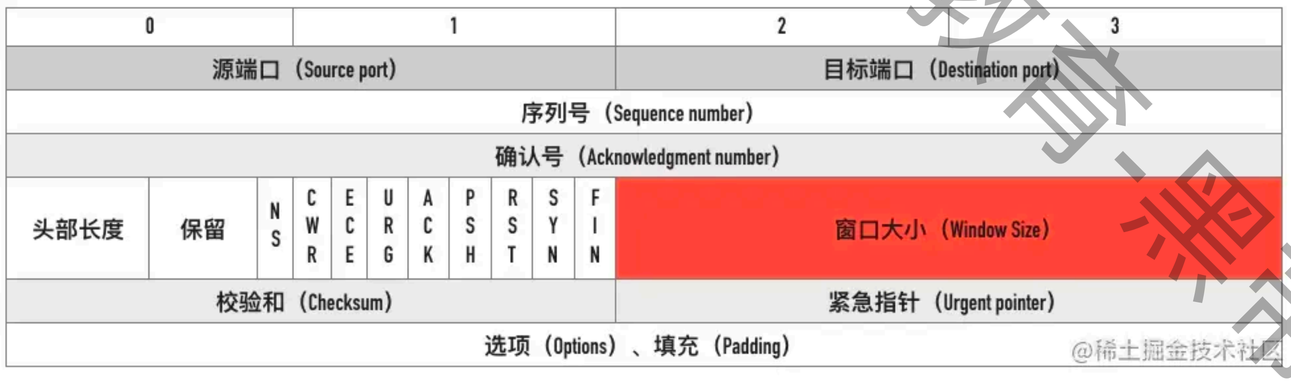
TCP Flags




```
1011 .... = Header Length: 44 bytes (11)
▼ 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·]
```

窗口大小

Window Size

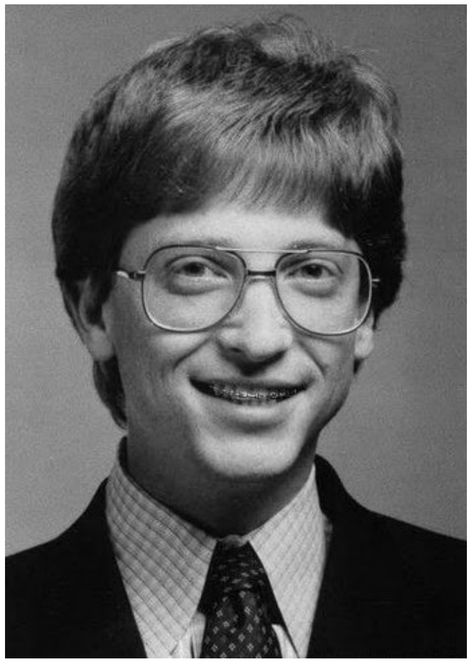


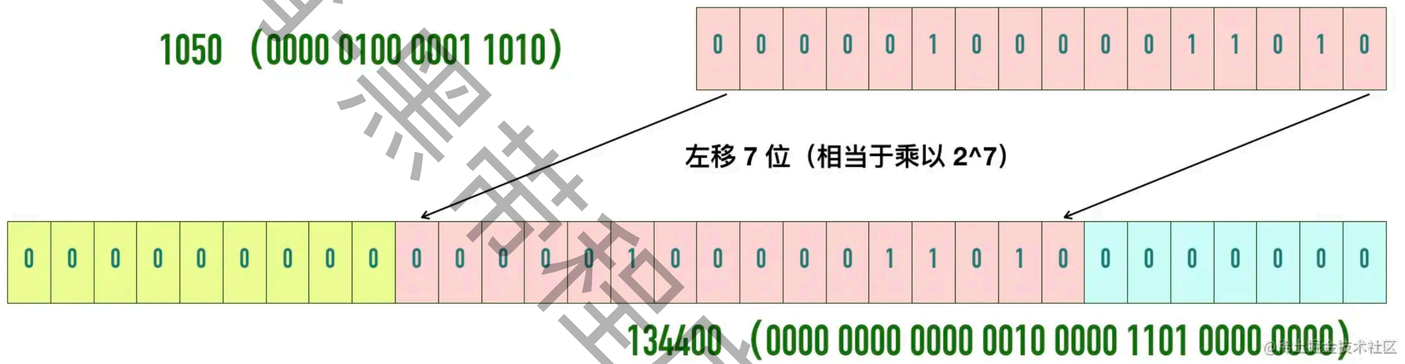
- 最大只有 65535 字节（64KB）

TCP 窗口缩放

Window Scale

“640K ought to be enough for anyone





窗口大小缩放前为 1050，缩放因子为 7

真正的窗口大小为 1050 * 128 = 134400

TCP 窗口缩放

Window Scale

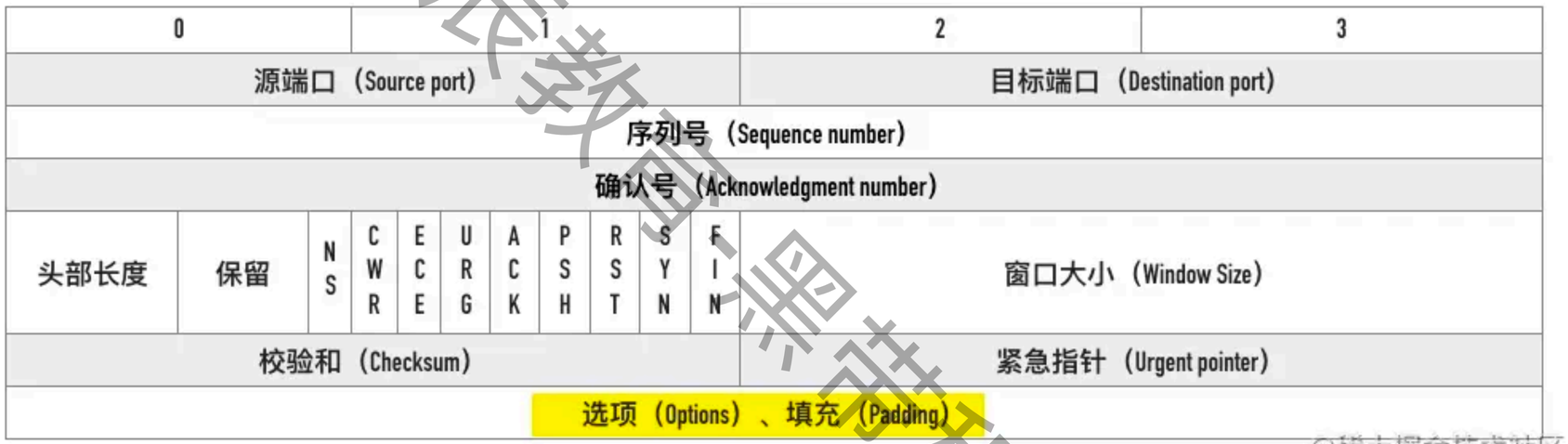

```
  3    0.006704    192.168.31.240          14.215.177.38    TCP      61024 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
  4    0.006841    192.168.31.240          14.215.177.38    HTTP     GET / HTTP/1.1
  5    0.012816    14.215.177.38           192.168.31.240   TCP      80 → 61024 [ACK] Seq=1 Ack=78 Win=24832 Len=0
```

```
▶ Frame 4: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits) on interface 0
▶ Ethernet II, Src: Apple_2f:dc:ab (38:f9:d3:2f:dc:ab), Dst: XiaomiCo_f1:a3:28 (f0:b4:29:f1:a3:28)
▶ Internet Protocol Version 4, Src: 192.168.31.240, Dst: 14.215.177.38
▼ Transmission Control Protocol, Src Port: 61024, Dst Port: 80, Seq: 1, Ack: 1, Len: 77
    Source Port: 61024
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 77]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 78    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 4096
    [Calculated window size: 262144]
    [Window size scaling factor: 64]
    Checksum: 0x11cc [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

window size = 4096 * 64 = 262144

可选项

Options、Padding



- MSS：最大段大小选项，是 TCP 允许的从对方接收的最大报文段
- SACK：选择确认选项
- Window Scale：窗口缩放选项

小结