细数 TCP 的定时器们

TCP 为每条连接建立了 7 个定时器：

- 连接建立定时器
- 重传定时器
- 延迟 ACK 定时器
- PERSIST 定时器
- KEEPALIVE 定时器
- FIN*WAIT*2 定时器
- TIME_WAIT 定时器

一、连接建立定时器（connection establishment）

当发送端发送 SYN 报文想建立一条新连接时，会开启连接建立定时器，如果没有收到对端的 ACK 包将进行重传。

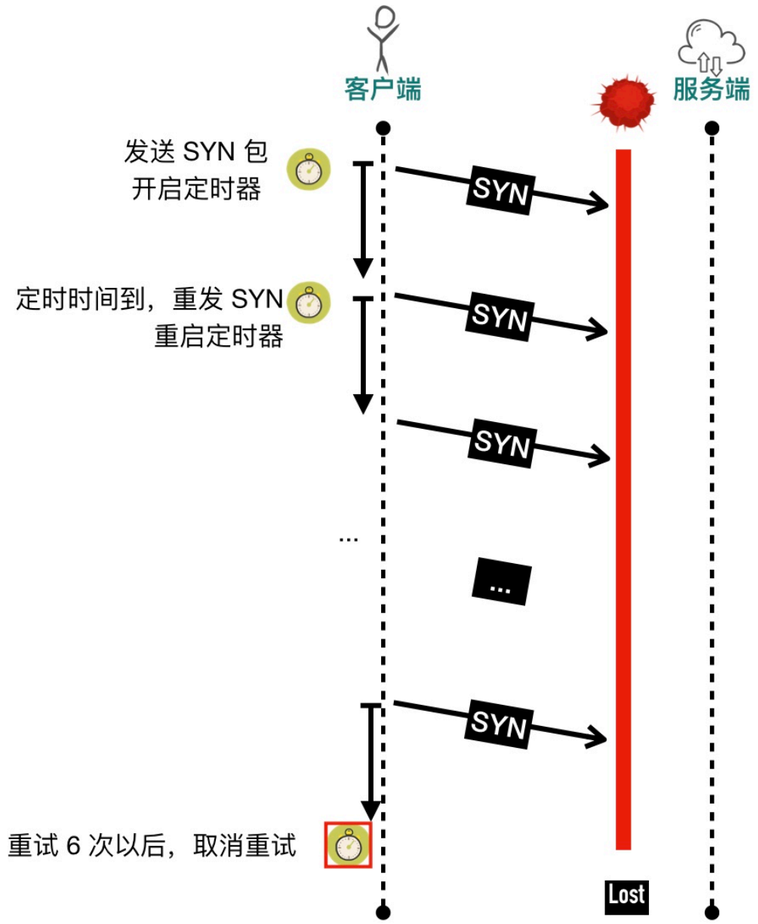
可以用一个最简单的 packetdrill 脚本来模拟这个场景

```
// 新建一个 server socket
+0    socket(..., SOCK_STREAM, IPPROTO_TCP) = 3

// 客户端 connect
+0 connect(3, ..., ...) = -1
```

抓包结果如下

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.176.50	192.0.2.1	TCP	43674 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
2	1.025288	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0
3	3.033040	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0
4	7.049193	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0
5	15.055395	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0
6	31.138640	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0
7	63.185337	192.168.176.50	192.0.2.1	TCP	[TCP Retransmission] 43674 → 8080 [SYN] Seq=0 Win=29200 Len=0

在我的电脑上，将重传 6 次（间隔 1s、2s、4s、8s、16s、32s），6 次重试以后放弃重试，connect 调用返回 -1，调用超时，整个过程如下：



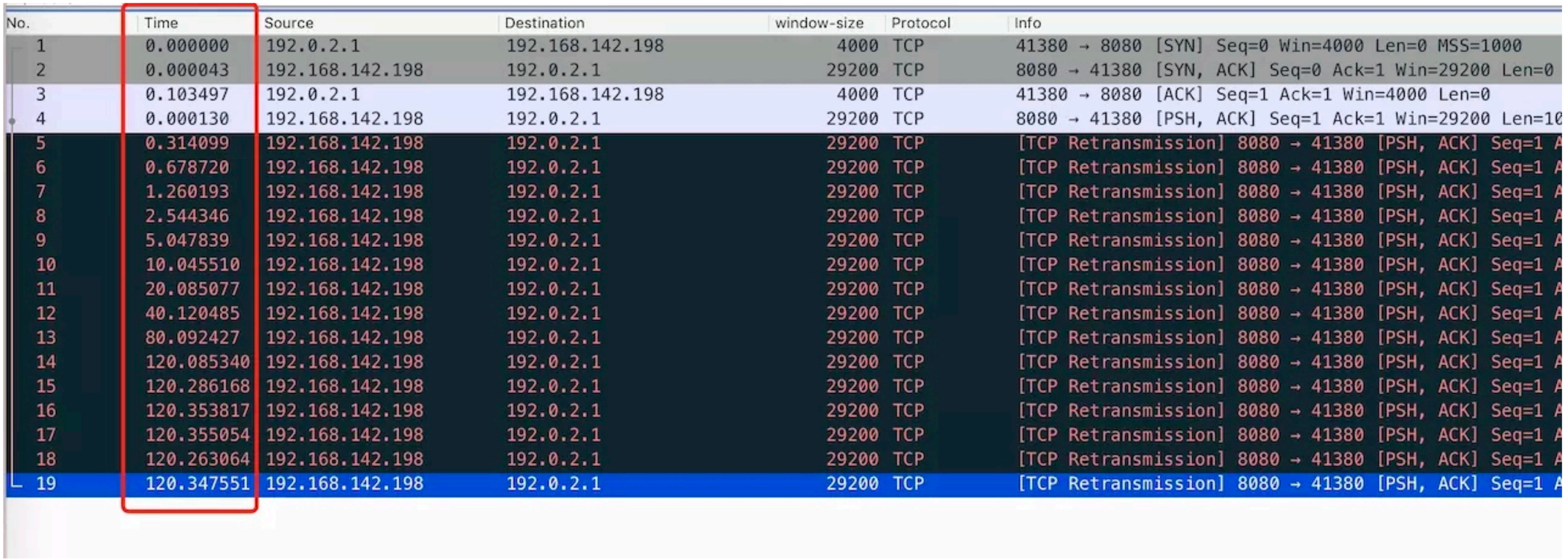
二、重传定时器（retransmission）

第一个定时器讲的是连接建立没有收到 ACK 的情况，如果在发送数据包的时候没有收到 ACK 呢?

还是用 packetdrill 脚本的方式来模拟

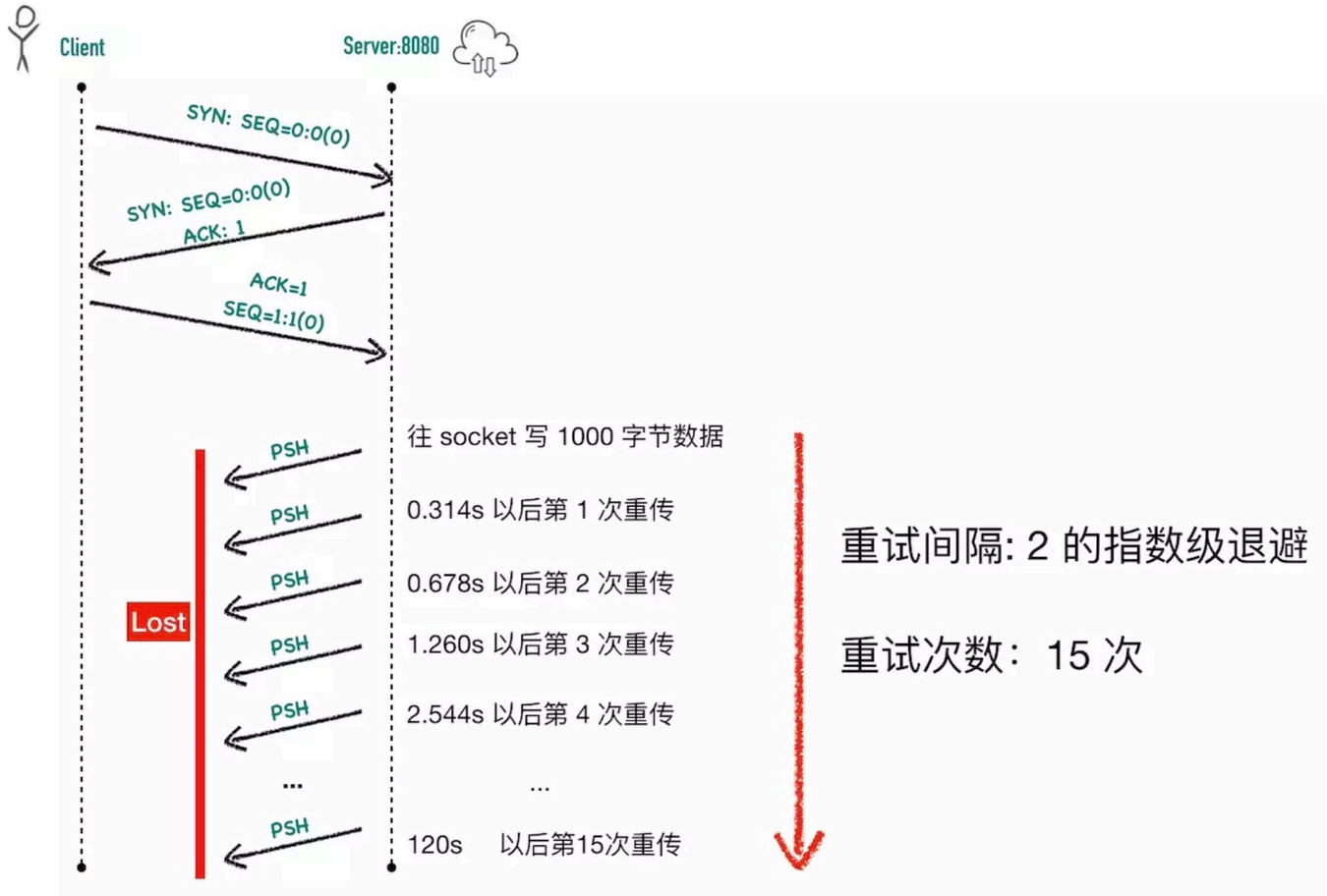
```
0   socket(..., SOCK_STREAM, IPPROTO_TCP) = 3
+0  setsockopt(3, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
+0  bind(3, ..., ...) = 0
+0  listen(3, 1) = 0

// 三次握手
+0  < S 0:0(0) win 4000 <mss 1000>
+0  > S. 0:0(0) ack 1 <...>
+.1 < . 1:1(0) ack 1 win 4000
+0  accept(3, ..., ...) = 4

// 往 fd 为 4 的 socket 文件句柄写入 1000 个字节数据（也即向客户端发送数据）
+0  write(4, ..., 1000) = 1000

// 注释掉 向协议栈注入 ACK 包的代码，模拟客户端不回 ACK 包的情况
// +.1 < . 1:1(0) ack 1001 win 1000

+0 `sleep 1000000`
```

抓包结果如下



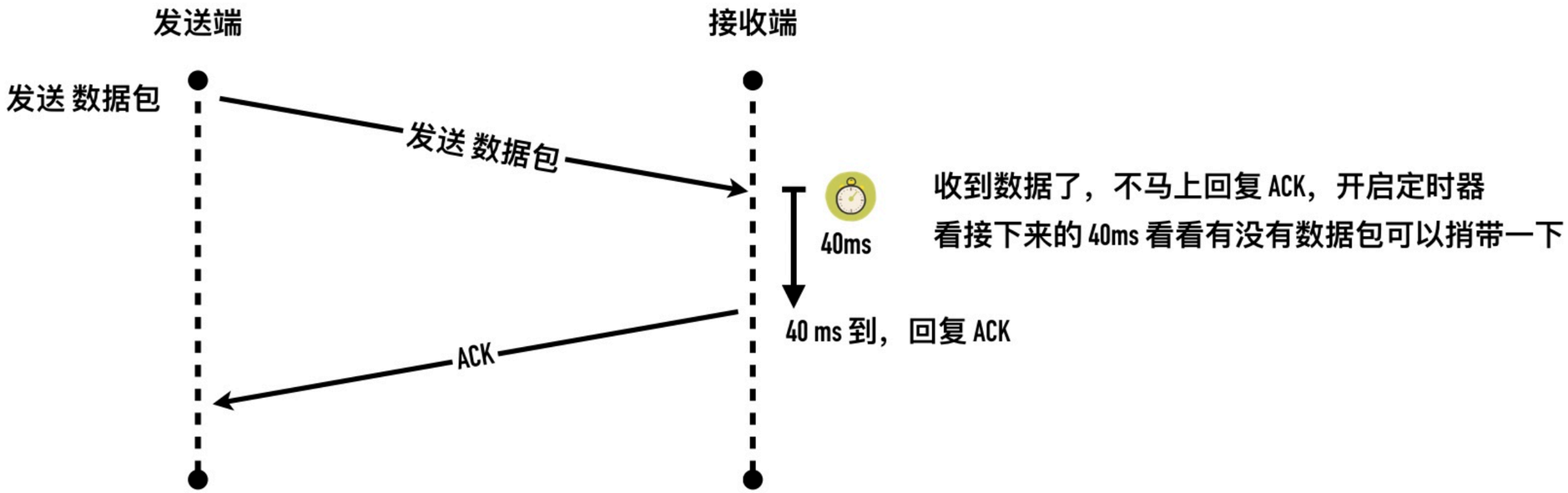
No.	Time	Source	Destination	window-size	Protocol	Info
1	0.000000	192.0.2.1	192.168.142.198	4000	TCP	41380 → 8080 [SYN] Seq=0 Win=4000 Len=0 MSS=1000
2	0.000043	192.168.142.198	192.0.2.1	29200	TCP	8080 → 41380 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
3	0.103497	192.0.2.1	192.168.142.198	4000	TCP	41380 → 8080 [ACK] Seq=1 Ack=1 Win=4000 Len=0
4	0.000130	192.168.142.198	192.0.2.1	29200	TCP	8080 → 41380 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=10
5	0.314099	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
6	0.678720	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
7	1.260193	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
8	2.544346	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
9	5.047839	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
10	10.045510	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
11	20.085077	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
12	40.120485	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
13	80.092427	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
14	120.085340	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
15	120.286168	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
16	120.353817	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
17	120.355054	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
18	120.263064	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A
19	120.347551	192.168.142.198	192.0.2.1	29200	TCP	[TCP Retransmission] 8080 → 41380 [PSH, ACK] Seq=1 A

重传时间间隔是指数级退避，直到达到 120s 为止，重传次数是15次（这个值由操作系统的 /proc/sys/net/ipv4/tcp_retries2 决定)，总时间将近 15 分钟。



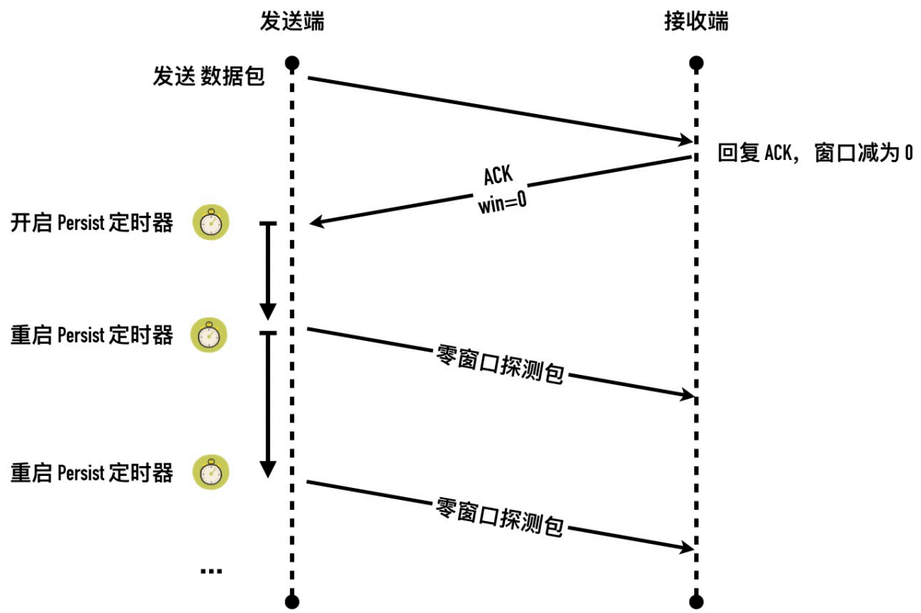
三、延迟 ACK 定时器

在 TCP 收到数据包以后在没有数据包要回复时，不马上回复 ACK。这时开启一个定时器，等待一段时间看是否有数据需要回复。如果期间有数据要回复，则在回复的数据中捎带 ACK，如果时间到了也没有数据要发送，则也发送 ACK。在 Centos7 上这个值为 40ms。



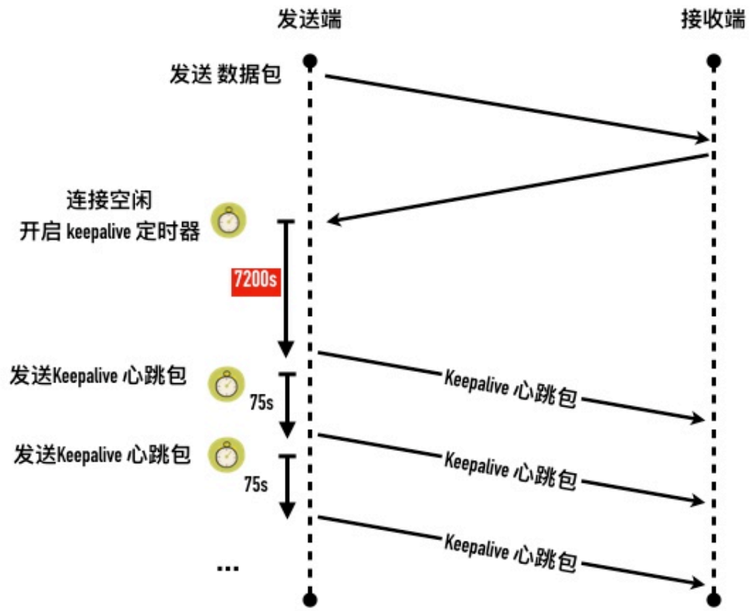
四、persist 定时器

Persist 定时器是专门为零窗口探测而准备的。我们都知道 TCP 利用滑动窗口来实现流量控制，当接收端 B 接收窗口为 0 时，发送端 A 此时不能再发送数据，发送端此时开启 Persist 定时器，超时后发送一个特殊的报文给接收端看对方窗口是否已经恢复，这个特殊的报文只有一个字节。



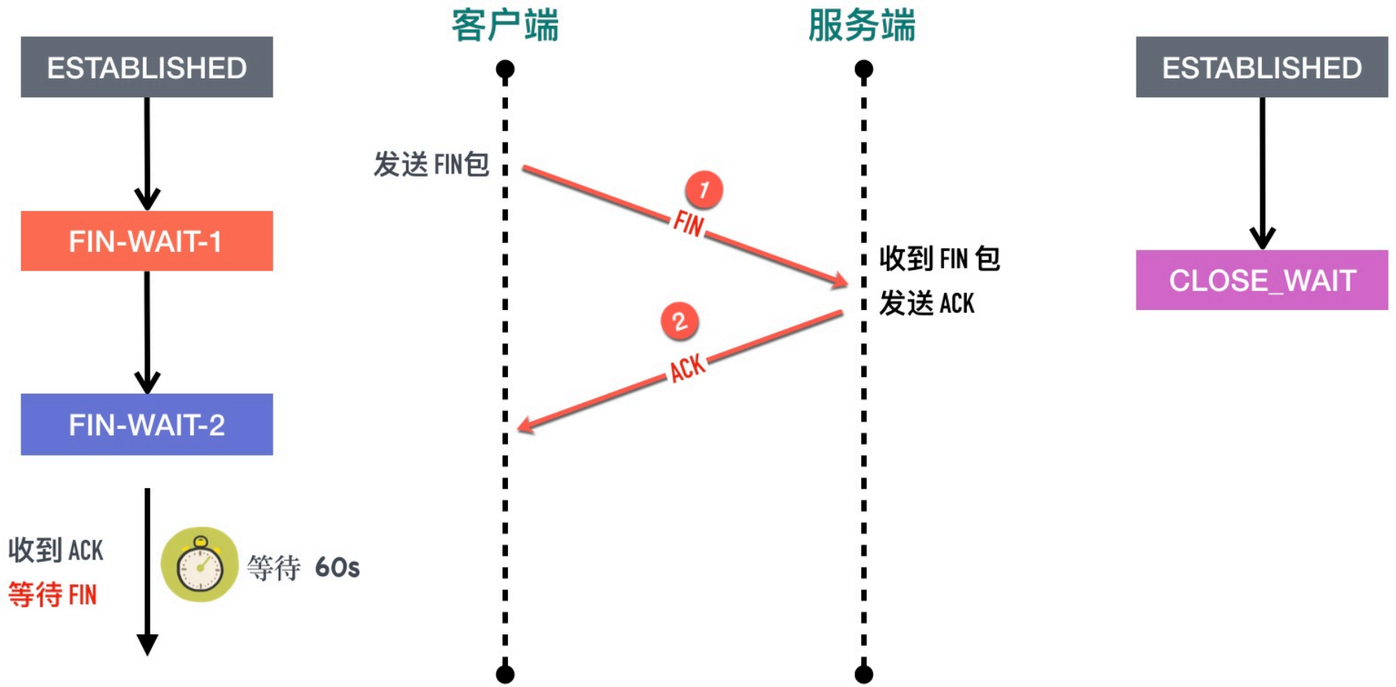
五、保活定时器（keepalive timer）

如果通信以后一段时间有再也没有传输过数据，怎么知道对方是不是已经挂掉或者重启了呢？于是 TCP 提出了一个做法就是在连接的空闲时间超过 2 小时，会发送一个探测报文，如果对方有回复则表示连接还活着，对方还在，如果经过几次探测对方都没有回复则表示连接已失效，客户端会丢弃这个连接。



六、FIN_WAIT_2 定时器

四次挥手过程中，主动关闭的一方收到 ACK 以后从 FIN*WAIT*1 进入 FIN*WAIT*2 状态等待对端的 FIN 包的到来，FIN*WAIT*2 定时器的作用是防止对方一直不发送 FIN 包，防止自己一直傻等。这个值由/proc/sys/net/ipv4/tcp_fin_timeout 决定，默认值为 60s



七、TIME_WAIT 定时器

TIME*WAIT* 定时器也称为 *2MSL* 定时器，可能是这七个里面名气最大的，主动关闭连接的一方在 *TIME*WAIT 持续 2 个 MSL 的时间，超时后端口号可被安全的重用。