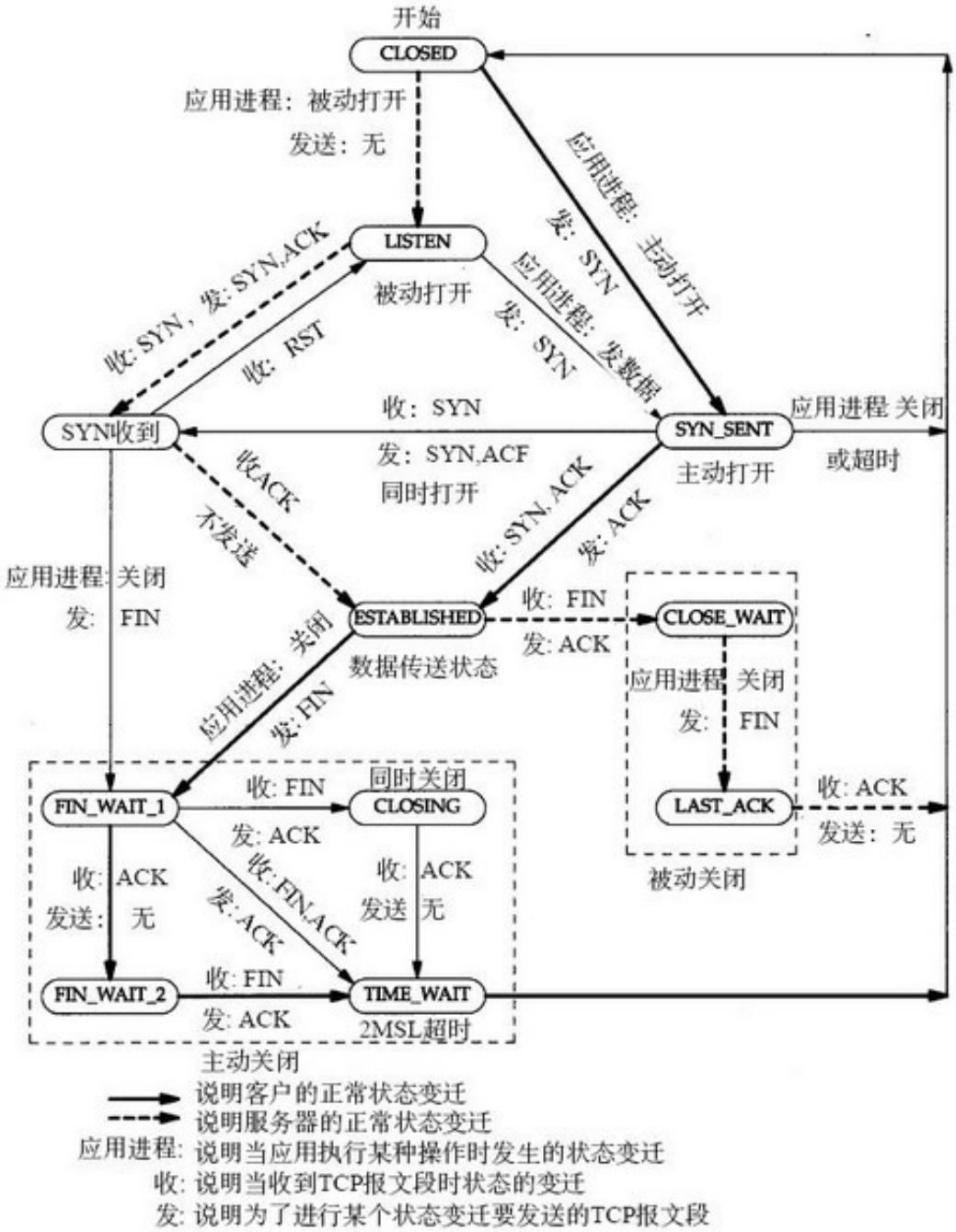
TCP/IP状态图的TIME_WAIT作用

在TCP/IP状态图中，有很多种的状态，它们之间有的是可以互相转换的，也就是说，从一种状态转到另一种状态，但是这种转换不是随便发送的，是要满足一定的条件。TCP/IP状态图看起来更像是自动机。下图即为TCP/IP状态。



由上图可以看出，一共有11种不同的状态。这11种状态描述如下：

1. CLOSED：关闭状态，没有连接活动或正在进行；
2. LISTEN：监听状态，服务器正在等待连接进入；

3. SYN_RCVD：收到一个连接请求，尚未确认；
4. SYN_SENT：已经发出连接请求，等待确认；
5. ESTABLISHED：连接建立，正常数据传输状态；
6. FIN_WAIT 1：（主动关闭）已经发送关闭请求，等待确认；
7. FIN_WAIT 2：（主动关闭）收到对方关闭确认，等待对方关闭请求；
8. TIME_WAIT：完成双向关闭，等待所有分组死掉；
9. CLOSING：双方同时尝试关闭，等待对方确认；
10. CLOSE_WAIT：（被动关闭）收到对方关闭请求，已经确认；
11. LAST_ACK：（被动关闭）等待最后一个关闭确认，并等待所有分组死掉。

在这11中状态当中，TIME_WAIT这种状态是最重要的，也是最难理解的。

为什么需要 TIME_WAIT 状态？

假设最终的 ACK 丢失 ， server 将重发 FIN ， client 必须维护 TCP 状态信息以便可以重发最终的 ACK ，否则会发送RST ，结果 server 认为发生错误。 TCP 实现必须可靠地终止连接的两个方向 (全双工关闭) ， client 必须进 TIME_WAIT状态，因为 client 可能面临重发最终 ACK 的情形。先调用 close() 的一方会进入 TIME_WAIT 状态

为什么 TIME_WAIT 状态需要保持 2MSL 这么长的时间？

如果 TIME_WAIT 状态保持时间不足够长 (比如小于 2MSL) ，第一个连接就正常终止了。
第二个拥有相同相关五元组的连接出现，而第一个连接的重复报文到达，干扰了第二个连接。
TCP 实现必须防止某个连接的重复报文在连接终止后出现，所以让 TIME_WAIT 状态保持时间足够长 (2MSL) ，连接相应方向上的 TCP 报文要么完全响应完毕，要么被丢弃。建立第二个连接的时候，不会混淆。

根据《TCP/IP详解》中的TCP的建立和终止中有关"TCP的终止"的讲解
TCP的终止通过双方的四次握手实现。发起终止的一方执行主动关闭，响应的另一方执行被动关闭。

1. 发起方更改状态为FIN_WAIT_1，关闭应用程序进程，发出一个TCP的FIN段；
2. 接收方收到FIN段，返回一个带确认序号的ACK，同时向自己对应的进程发送一个文件结束符EOF，同时更改状态为CLOSE_WAIT，发起方接到 ACK后状态更改为FIN_WAIT_2；
3. 接收方关闭应用程序进程，更改状态为LAST_ACK，并向对方发出一个TCP的FIN段；
4. 发起方接到FIN后状态更改为TIME_WAIT，并发出这个FIN的ACK确认。ACK发送成功后(2MSL内)双方TCP状态变为CLOSED。

我们不难看出上面的显示的结果的意思。根据TCP协议，主动发起关闭的一方，会进入TIME_WAIT状态(TCP实现必须可靠地终止连接的两个方向(全双工关闭))，持续2*MSL (Max Segment Lifetime)，缺省为240秒.

TIME_WAIT状态的作用

主动关闭的Socket端会进入TIME_WAIT状态，并且持续2MSL时间长度，MSL就是maximum segment lifetime(最大分节生命期），这是一个IP数据包能在互联网上生存的最长时间，超过这个时间将在网络中消失。MSL在RFC 1122上建议是2分钟，而源自berkeley的TCP实现传统上使用30秒，因而，TIME_WAIT状态一般维持在1-4分钟。

TIME_WAIT状态存在的理由：

1）可靠地实现TCP全双工连接的终止

在进行关闭连接四路握手协议时，最后的ACK是由主动关闭端发出的，如果这个最终的ACK丢失，服务器将重发最终的FIN，因此客户端必须维护状态信息允 许它重发最终的ACK。如果不维持这个状态信息，那么客户端将响应RST分节，服务器将此分节解释成一个错误（在java中会抛出connection reset的SocketException)。因而，要实现TCP全双工连接的正常终止，必须处理终止序列四个分节中任何一个分节的丢失情况，主动关闭
的客户端必须维持状态信息进入TIME_WAIT状态。

2）允许老的重复分节在网络中消逝

TCP分节可能由于路由器异常而“迷途”，在迷途期间，TCP发送端可能因确认超时而重发这个分节，迷途的分节在路由器修复后也会被送到最终目的地，这个 原来的迷途分节就称为lost duplicate。在关闭一个TCP连接后，马上又重新建立起一个相同的IP地址和端口之间的TCP连接，后一个连接被称为前一个连接的化身（incarnation)，那么有可能出现这种情况，前一个连接的迷途重复分组在前一个连接终止后出现，从而被误解成从属于新的化身。为了避免这个情 况，TCP不允许处于TIME_WAIT状态的连接启动一个新的化身，因为TIME_WAIT状态持续2MSL，就可以保证当成功建立一个TCP连接的时 候，来自连接先前化身的重复分组已经在网络中消逝。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）