计网题目

jask

09/09/2024

Socket 编程

服务端和客户端初始化 socket ，得到文件描述符；

服务端调用 bind ，将绑定在 IP 地址和端口；

服务端调用 listen ，进行监听；

服务端调用 accept ，等待客户端连接；

客户端调用 connect ，向服务器端的地址和端口发起连接请求；

服务端 accept 返回用于传输的 socket 的文件描述符；

客户端调用 write 写入数据；服务端调用 read 读取数据；

客户端断开连接时，会调用 close ，那么服务端 read 读取数据的时候，就会读取到了 EOF ，待处理完数据后，服务端调用 close ，表示连接关闭。

这里需要注意的是，服务端调用 accept 时，连接成功了会返回一个已完成连接的 socket，后续用来传输数据。

所以，监听的 socket 和真正用来传送数据的 socket，是「两个」socket，一个叫作监听 socket，一个叫作已完成连接 socket。

成功连接建立之后，双方开始通过 read 和 write 函数来读写数据，就像往一个文件流里面写东西一样。

listen 时候参数 backlog 的意义?

Linux 内核中会维护两个队列：

未完成连接队列 (SYN 队列)：接收到一个 SYN 建立连接请求，处于 SYN_RCVD 状态；

已完成连接队列 (Accpet 队列)：已完成 TCP 三次握手过程，处于 ESTABLISHED 状态；

```
int listen(int sockfd,int backlog);
```

参数一 socketfd 为 socketfd 文件描述符

参数二 backlog，这参数在历史版本有一定的变化

早期 Linux 内核 backlog 是 SYN 队列大小，也就是未完成的队列大小。

在 Linux 内核 2.2 之后,backlog 变成 accept 队列，也就是已完成连接建立的队列长度，所以现在通常认为 backlog 是 accept 队列。

但是上限值是内核参数 somaxconn 的大小，也就说 accpet 队列长度 = min(backlog, somaxconn)。

accept 发生在三次握手的哪一步?

客户端的协议栈向服务器端发送了 SYN 包，并告诉服务器端当前发送序列号 client_isn，客户端进入 SYN_SENT 状态；

服务器端的协议栈收到这个包之后，和客户端进行 ACK 应答，应答的值为 client_isn+1，表示对 SYN 包 client_isn 的确认，同时服务器也发送一个 SYN 包，告诉客户端当前我的发送序列号为 server_isn，服务器端进入 SYN_RCVD 状态；

客户端协议栈收到 ACK 之后，使得应用程序从 connect 调用返回，表示客户端到服务器端的单向连接建立成功，客户端的状态为 ESTABLISHED，同时客户端协议栈也会对服务器端的 SYN 包进行应答，应答数据为 server_isn+1；

应答包到达服务器端后，服务器端协议栈使得 accept 阻塞调用返回，这个时候服务器端到客户端的单向连接也建立成功，服务器端也进入 ESTABLISHED 状态。

我们可以得知客户端 connect 成功返回是在第二次握手，服务端 accept 成功返回是在三次握手成功之后。

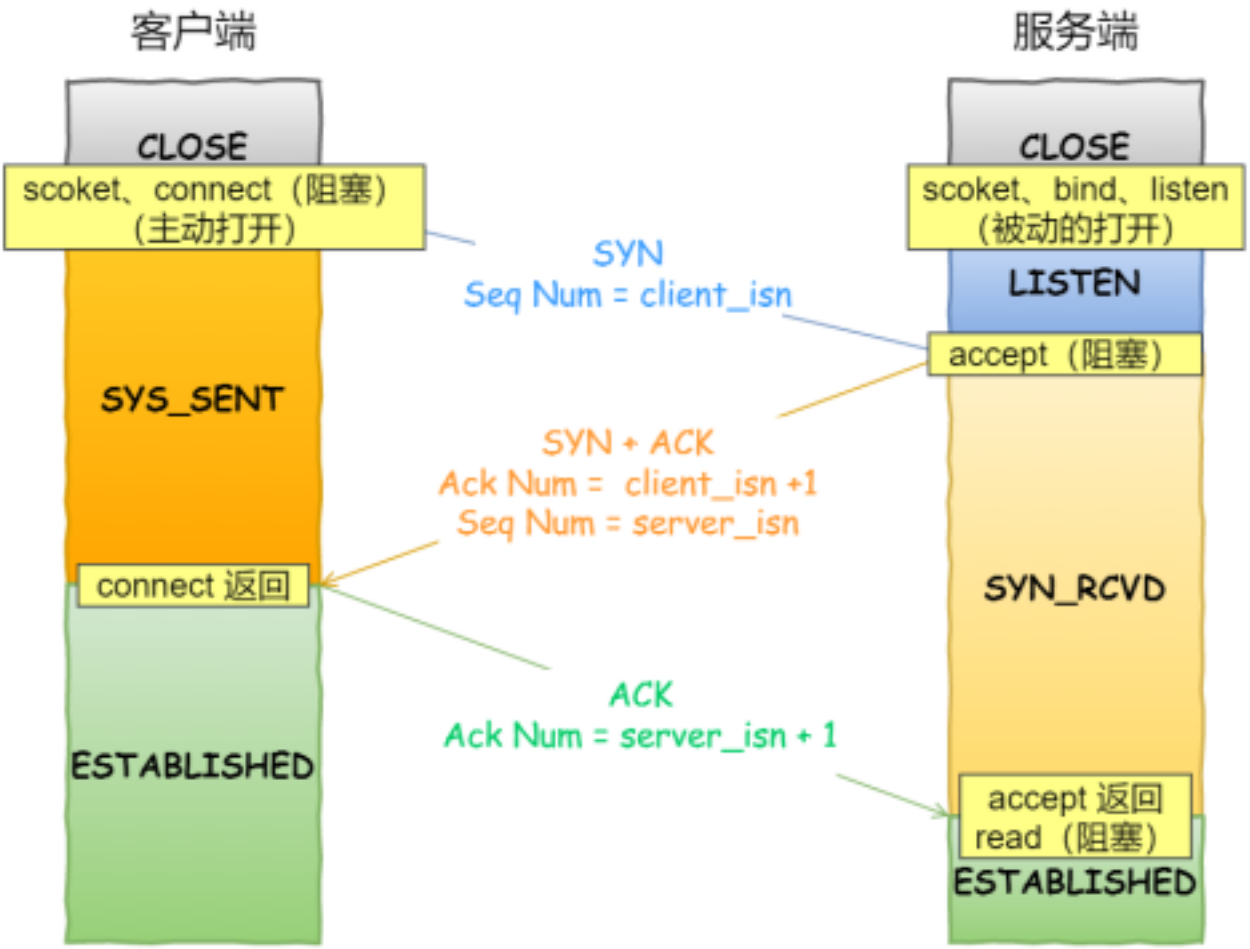
客户端
服务端
CLOSE
scoket、connect（阻塞）
（主动打开）
SYS_SENT
connect 返回
ESTABLISHED
CLOSE
scoket、bind、listen
（被动的打开）
LISTEN
accept（阻塞）
SYN_RCVD
accept 返回
read（阻塞）
ESTABLISHED
SYN
Seq Num = client_isn
SYN + ACK
Ack Num = client_isn +1
Seq Num = server_isn
ACK
Ack Num = server_isn + 1

Figure 1: 流程

客户端调用 close 了，连接是断开的流程是什么？

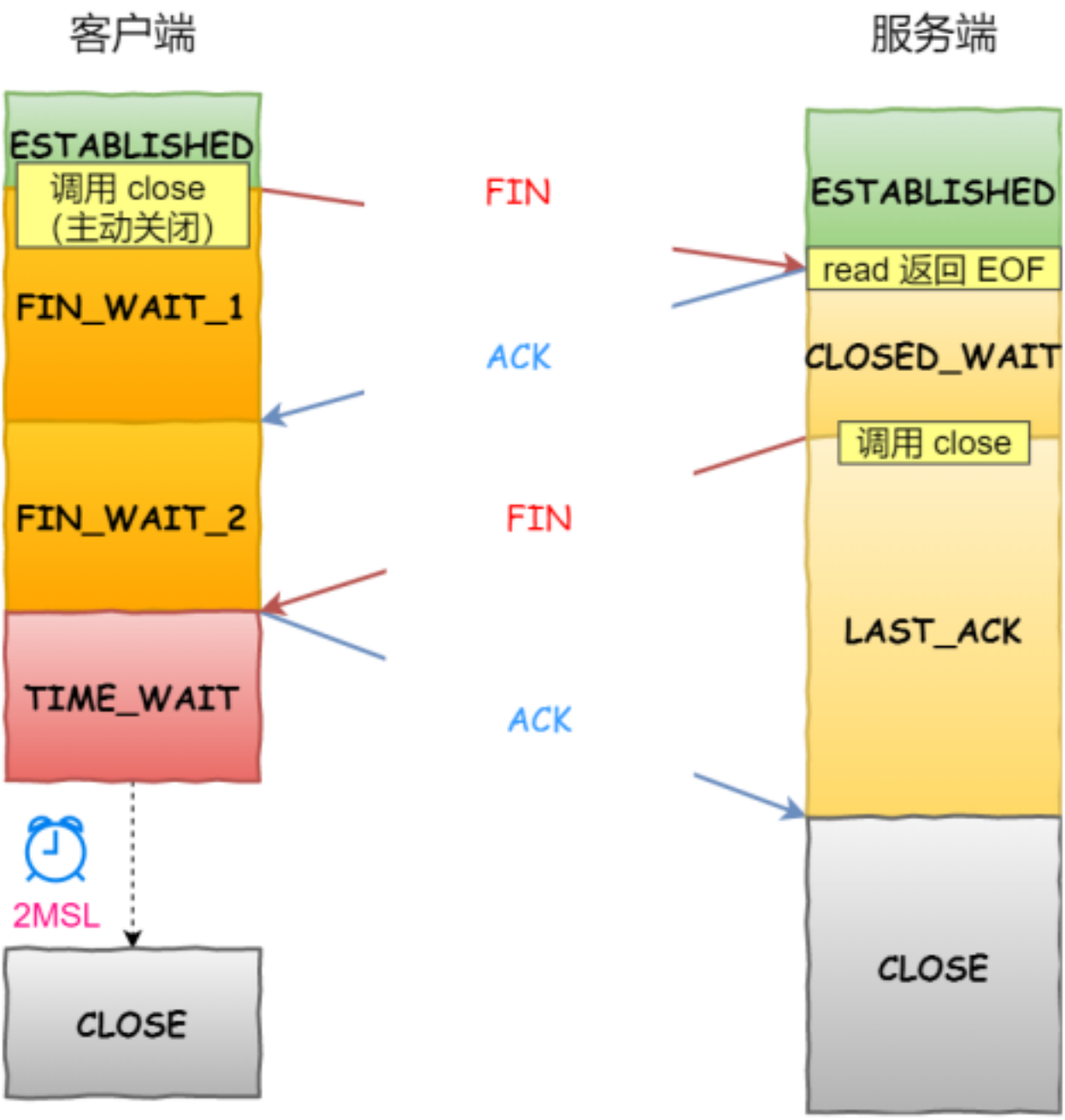

Figure 2: 流程

客户端调用 close ，表明客户端没有数据需要发送了，则此时会向服务端发送 FIN 报文，进入 FIN_WAIT_1 状态；

服务端接收到了 FIN 报文,TCP 协议栈会为 FIN 包插入一个文件结束符 EOF 到接收缓冲区中，应用程序可以通过 read 调用来感知这个 FIN 包。这个 EOF 会被放在已排队等候的其他已接收的数据之后，这就意味着服务端需要处理这种异常情况，因为 EOF 表示在该连接上再无额外数据到达。此时，服务端进入 CLOSE_WAIT 状态；

接着，当处理完数据后，自然就会读到 EOF ，于是也调用 close 关闭它的套接字，这会使得客户端会发出一个 FIN 包，之后处于 LAST_ACK 状态；

客户端接收到服务端的 FIN 包，并发送 ACK 确认包给服务端，此时客户端将进入 TIME_WAIT 状态；

服务端收到 ACK 确认包后，就进入了最后的 CLOSE 状态；

客户端经过 2MSL 时间之后，也进入 CLOSE 状态；

如果客户端第四次挥手 ack 丢失，服务端超时重发的 fin 报文也丢失，客户端 timewait 时

间超过了 2msl，这个时候会发生什么？认为连接已经关闭吗？

当客户端 timewait 时间超过了 2MSL，则客户端就直接进入关闭状态。

服务端超时重发 fin 报文的次数如果超过 tcp_orphan_retries 大小后，服务端也会关闭 TCP 连接。