Charles的技术博客

自己动手写分布式KV存储引擎（三）：网络框架中的客户端实现原理

2016-10-30 | 分布式

引言

自己动手写分布式KV存储引擎系列文章的目标是记录基于LevelDB(RockDB)构建一个分布式KV存储引擎实现过程，此系列文章对应的源码在DSTORE。

本文主要分析了网络框架中客户端的实现原理，全文分为如下两部分

- 客户端功能需求
- 客户端实现

本系列的其他文章还包括：

- 自己动手写分布式KV存储引擎（一）：设计和实现网络框架
- 自己动手写分布式KV存储引擎（二）：网络框架中的定时器原理和实现

客户端功能需求

在分布式系统中，一台服务器在与其他服务器交互的过程中，既扮演server端，也扮演client端，因此，网络框架中client端的实现也是至关重要的，一般地，网络框架中client端至少提供以下接口：

- connect: 连接其他服务器，因为是服务端程序，需要高性能，因此，此操作必须是非阻塞的
- read: 读server端发来的数据
- write： 往server端写数据

客户端实现

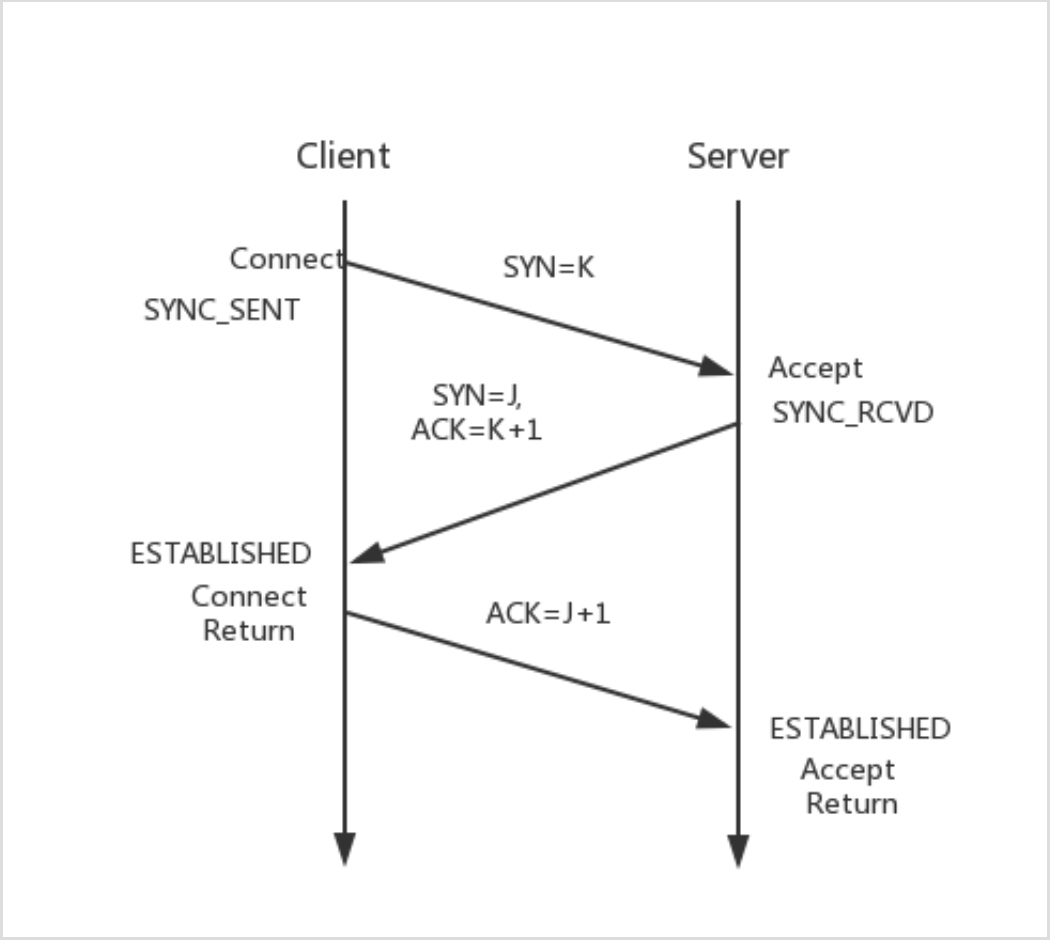
针对网络框架中的客户端需要的各种接口，本节讨论其功能实现。

connect

先来看看阻塞方式的connect，一般其用法如下

```
1  int ret = connect(fd, server_addr, server_len);
2  send(fd, data);
3  recv(fd, data);
```

阻塞的connect函数调用如下，其TCP状态转换如下图：



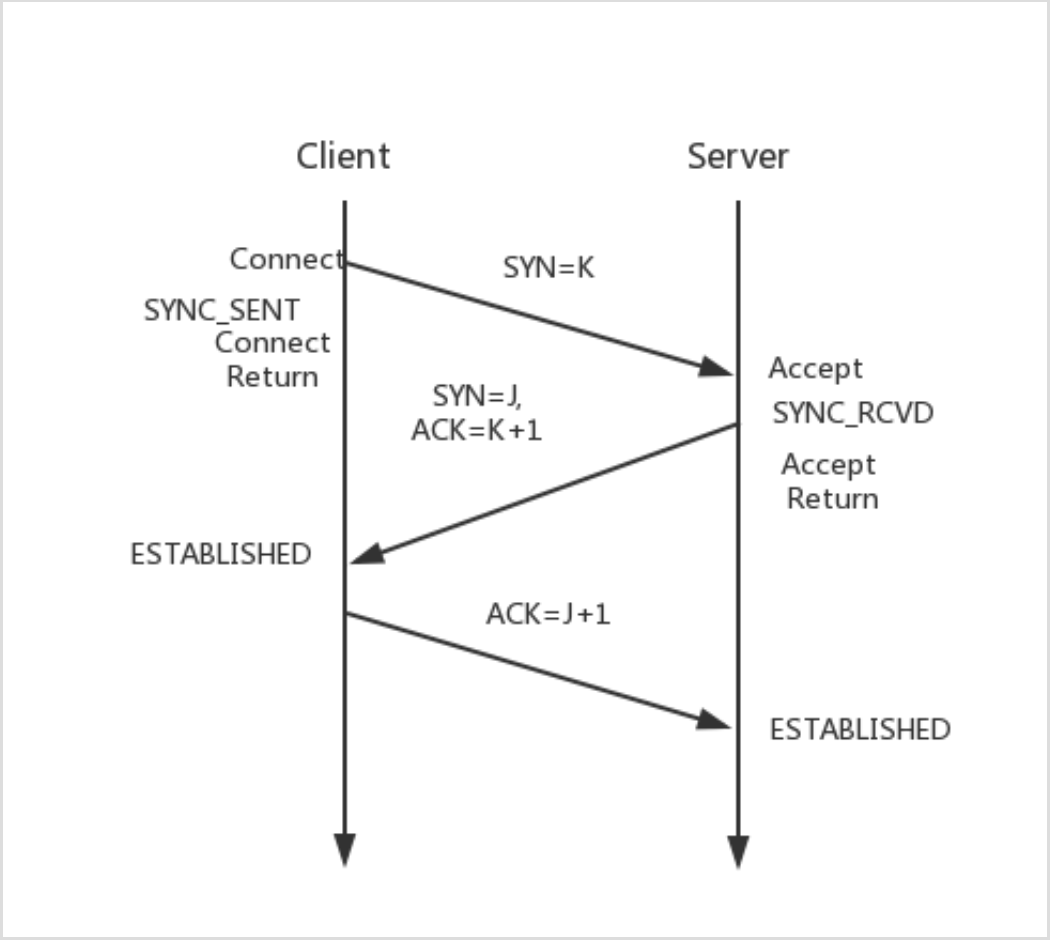
如上图所示，整个流程如下：

1. client端调用阻塞式connect，操作系统会向server端发送SYN数据包，并且client端的TCP状态会变成SYNC_SENT
2. server端调用accept接受connect请求，首先设置其TCP状态为SYNC_RCVD，然后会发送对server端的SYN包的确认包
3. client端接收到server端的确认包，操作系统将TCP状态设置成ESTABLISHED，这时候阻塞的connect函数返回，并且，操作系统会向服务端发送确认包

4. 服务端在收到客户端的确认包之后，操作系统将TCP连接状态设置成ESTABLISHED，接着，服务端的accept调用返回

从上述调用流程可以看出，阻塞式的connect会等待收到服务端的确认包之后，才会返回，这中间的等待时间是一次网络包的往返时间，对于服务端程序来讲，阻塞在等待连接建立上是不能接受的，因此，必须采用非阻塞的connect。

非阻塞的connect的调用如下，其TCP状态转换如下图：



如上图所示，整个流程如下：

1. client端调用非阻塞的connect，操作系统会向server端发送SYN数据包，并且client端的TCP状态会变成SYNC_SENT，然后client端返回
2. server端调用accept接受connect请求，其首先设置其TCP状态为SYNC，发送对client端的SYN包的确认包，接着accept函数返回
3. client端收到server端的确认包之后，操作系统将其状态设置成ESTABLISHED，然后，向server端发送确认包
4. server端收到确认包之后，操作系统将其状态设置成ESTABLISHED

对于非阻塞的connect，没有了等待server端回确认包的过程，但是，网络框架需要处理的是，连接真正建立的时候需要通知应用程序来处理。

根据linux mannual文档，说明如下

> EINPROGRESS
> The socket is nonblocking and the connection cannot be completed immediately. It is possible to select(2) or poll(2) for completion by selecting the socket for writing. After select(2) indicates writability, use getsockopt(2) to read the SO_ERROR option at level SOL_SOCKET to determine whether connect() completed successfully (SO_ERROR is zero) or unsuccessfully (SO_ERROR is one of the usual error codes listed here, explaining the reason for the failure).

如上说明，非阻塞connect之后，如果返回值是EINPROGRESS，那么需要用select或者epoll监听可写事件，然后，使用getsockopt来获取是否有错误，如果没有错误，说明连接建立成功，否则，连接建立失败。当然，如果返回值是0，说明在非阻塞的connect返回时，连接已经建立成功，这时候，跟处理阻塞式的connect是一样的。

整个非阻塞connect的实现在tcp_client，其中非阻塞connect调用是connect中完成。

read

read的实现与server端的read一致，这里就不再赘述了。

write

write的实现与server端的不同，需要根据不同情况来做不同的处理：

- 如果连接是处于连接中的状态，那么收到可写事件时，需要通过getsockopt函数来检查连接是否正确建立，并设置连接为已连接状态
- 如果连接是已连接的状态，那么收到可写事件时，则会尝试调用write往内核buffer中写数据

自己动手写分布式KV存储引擎的前三篇描述了如何设计和实现网络框架，接下来的文章将会关注如何基于网络框架设计和实现RPC库，敬请期待。

PS:

本博客更新会在第一时间推送到微信公众号，欢迎大家关注。

参考文献

- Unix网络编程

#C++ #网络编程

‹自己动手写分布式KV存储引擎（二）：网络框架中的定时器原理和实现

golang实现Raft（一）：选主›

Charles的技术博客正在使用多说