代码如诗

小楼一夜听春雨

muduo 的 shutdown() 没有直接关闭 TCP 连接？

http://blog.csdn.net/Solstice/article/details/6208634

今天收到一位网友来信：

在 simple 中的 daytime 示例中，服务端主动关闭时调用的是如下函数序列，这不是只是关闭了连接上的写操作吗，怎么是关闭了整个连接？

```
1: void DaytimeServer::onConnection(const muduo::net::TcpConnectionPtr& conn)
2: {
3:   if (conn->connected())
4:   {
5:     conn->send(Timestamp::now().toFormattedString() + "/n");
6:     conn->shutdown();
7:   }
8: }
9:
10: void TcpConnection::shutdown()
11: {
12:   if (state_ == kConnected)
13:   {
14:     setState(kDisconnecting);
15:     loop_->runInLoop(boost::bind(&TcpConnection::shutdownInLoop, this));
16:   }
17: }
18:
19: void TcpConnection::shutdownInLoop()
20: {
21:   loop_->assertInLoopThread();
22:   if (!channel_->isWriting())
23:   {
24:     // we are not writing
25:     socket_->shutdownWrite();
26:   }
27: }
28:
29: void Socket::shutdownWrite()
30: {
31:   sockets::shutdownWrite(sockfd_);
32: }
33:
34: void sockets::shutdownWrite(int sockfd)
35: {
36:   if (::shutdown(sockfd, SHUT_WR) < 0)
```

公告

昵称：小 楼 一 夜 听 春 雨
园龄：7年9个月
粉丝：391
关注：3
+加关注

‹ 2017年9月 ›

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

ACE(4)
BOOST(8)
C/C++(67)
C++(54)
D3D(114)
D3D1(3)
Game_dev(7)
GO
IOS(5)
Java(2)
linux(112)
Lua(3)
OGRE(44)
OpenGLES(4)
Python(28)
Windows(8)

```
37:   {
38:     LOG_SYSERR << "sockets::shutdownWrite";
39:   }
40: }
```

陈硕答复如下：

Muduo TcpConnection 没有提供 close，而只提供 shutdown ，这么做是为了收发数据的完整性。

TCP 是一个全双工协议，同一个文件描述符既可读又可写， shutdownWrite() 关闭了“写”方向的连接，保留了“读”方向，这称为 TCP half-close。如果直接 close(socket_fd)，那么 socket_fd 就不能读或写了。

用 shutdown 而不用 close 的效果是，如果对方已经发送了数据，这些数据还“在路上”，那么 muduo 不会漏收这些数据。换句话说，muduo 在 TCP 这一层面解决了“当你打算关闭网络连接的时候，如何得知对方有没有发了一些数据而你还没有收到？”这一问题。当然，这个问题也可以在上面的协议层解决，双方商量好不再互发数据，就可以直接断开连接。

等于说 muduo 把“主动关闭连接”这件事情分成两步来做，如果要主动关闭连接，它会先关本地“写”端，等对方关闭之后，再关本地“读”端。练习：阅读代码，回答“如果被动关闭连接，muduo 的行为如何？” 提示：muduo 在 read() 返回 0 的时候会回调 connection callback，这样客户代码就知道对方断开连接了。

Muduo 这种关闭连接的方式对对方也有要求，那就是对方 read() 到 0 字节之后会主动关闭连接（无论 shutdownWrite() 还是 close()），一般的网络程序都会这样，不是什么问题。当然，这么做有一个潜在的安全漏洞，万一对方故意不不关，那么 muduo 的连接就一直半开着，消耗系统资源。

完整的流程是：我们发完了数据，于是 shutdownWrite，发送 TCP FIN 分节，对方会读到 0 字节，然后对方通常会关闭连接，这样 muduo 会读到 0 字节，然后 muduo 关闭连接。（思考题，在 shutdown() 之后，muduo 回调 connection callback 的时间间隔大约是一个 round-trip time，为什么？）

另外，如果有必要，对方可以在 read() 返回 0 之后继续发送数据，这是直接利用了 half-close TCP 连接。muduo 会收到这些数据，通过 message callback 通知客户代码。

那么 muduo 什么时候真正 close socket 呢？在 TcpConnection 对象析构的时候。TcpConnection 持有一个 Socket 对象，Socket 是一个 RAII handler，它的析构函数会 close(sockfd_)。这样，如果发生 TcpConnection 对象泄漏，那么我们从 /proc/pid/fd/ 就能找到没有关闭的文件描述符，便于查错。

muduo 在 read() 返回 0 的时候会回调 connection callback，然后把 TcpConnection 的引用计数减一，如果 TcpConnection 的引用计数降到零，它就会析构了。

参考：

《TCP/IP 详解》第一卷第 18.5 节，TCP Half-Close。

《UNIX 网络编程》第一卷第三版第 6.6 节， shutdown() 函数。

分类: 网络

小 楼 一 夜 听 春 雨
关注 - 3
粉丝 - 391
+加关注

0 0

« 上一篇：socket shutdown和close的区别
» 下一篇：linux下的c++filt 命令

posted @ 2017-07-10 15:31 小 楼 一 夜 听 春 雨 阅读(26) 评论(0) 编辑 收藏

wordpress(1)
测试(2)
架构设计(6)
开源(25)
人生感悟(14)
数据结构(5)
数据库(3)
算法(8)
网络(43)
英语学习(6)
游戏开发(72)
杂谈(26)
职业规划(10)

随笔档案

2017年9月 (1)
2017年8月 (29)
2017年7月 (11)
2017年6月 (25)
2017年5月 (1)
2017年4月 (5)
2017年3月 (13)
2017年2月 (4)
2017年1月 (2)
2016年12月 (12)
2016年11月 (20)
2016年10月 (31)
2016年9月 (13)
2016年8月 (13)
2016年7月 (7)
2016年6月 (8)
2016年5月 (6)
2016年4月 (4)
2016年3月 (12)
2016年2月 (32)
2016年1月 (27)
2015年12月 (5)
2015年4月 (1)
2014年12月 (9)
2014年10月 (2)
2014年9月 (3)
2014年8月 (4)
2014年7月 (1)
2014年6月 (2)
2014年5月 (2)
2014年4月 (1)
2014年3月 (4)
2014年1月 (2)
2013年12月 (2)
2013年11月 (13)
2013年10月 (7)
2013年9月 (2)
2013年8月 (7)
2013年7月 (11)
2013年6月 (2)
2013年4月 (2)
2013年1月 (1)
2012年12月 (2)
2012年11月 (8)
2012年10月 (5)
2012年9月 (21)
2012年8月 (14)
2012年7月 (7)
2012年6月 (5)
2012年5月 (17)
2012年4月 (10)
2012年3月 (12)
2012年2月 (9)
2012年1月 (12)