同步阻塞线程的唤醒问题研究*)

The Research of Wakening up a Synchronic Blocking Thread

胥光辉[1,2] 徐永森[1]

(南京大学软件新技术国家重点实验室 南京 210093)[1] (解放军理工大学指挥自动化学院 南京 210016)[2]

Abstract Firstly, the definition and characters of blocking mode and non-blocking mode in socket communication are introduced, and the question of wakening up a synchronic blocking thread is proposed. Then, a typical blocking socket communication within mobile stock operations is given, and a method based on exception detecting to waken up synchronic blocking threads is proposed. Additionally, a phenomenon of vibrating is also analyzed and solved. And lastly, a conclusion is given.

Keywords Socket communication, Blocking mode, Non-blocking mode, Exception detecting, Vibrating

1 引言

Socket 通信是一种常用的网络编程方法,一个通信 Socket 可以工作在阻塞(同步)模式,也可以工作在非阻塞(异步操作)模式[1,2]。处于阻塞模式的 Socket 的函数被调用之后,在完成所有动作之前不会返回。之所以称之为"阻塞"是因为这时该 Socket 什么也不能做了,即被阻塞了,直到被调用的函数返回。例如,Socket 的 Receive 函数在接收数据时可能需要等待任意长的时间才能完成。而一个处于非阻塞模式的 Socket 的函数在调用之后立刻返回,当等待的条件满足时,由操作系统自动调用一个事先编写好的回调函数来完成所需功能。

阻塞方式的 Socket 编程比较简单,程序员不需了解底层的通信细节,适用于多任务的环境。需要为通信 Socket 单独设置一个线程,当因该 Socket 阻塞而引起线程阻塞时,该线程不再占用计算资源,也不会影响主线程和其他线程的行为。

非阻塞方式的 Socket 通信编程灵活,对于程序员的要求较高,需要了解底层的通信细节,如回调通知、网络字节顺序、字符编码转换等,适用于单任务和多任务的环境。

阻塞方式和非阻塞方式的 Socket 通信各有利弊和应用场合,对于阻塞方式的 Socket 通信还存在一个同步阻塞线程的唤醒问题,即当一个线程中的 Socket 处于阻塞状态因而引起该线程阻塞时,如何在需要时唤醒该线程的问题。

2 一个典型的同步阻塞线程的唤醒问题

2.1 应用背景

考虑移动证券业务中证券公司的一个通信平台[3],该平台由一个通信服务器和若干个营业部通信机组成,通信服务器与每个营业部通信机采用两条 TCP 连接进行通信,一条用于发送数据,另一条用于接收数据。通信平台的结构如图 1 所示。通信服务器负责接收手机上传的短消息,并将上传短消息转发给相应营业部去处理;同时接收各营业部发送的处理结果,并组织成短消息下发给手机。营业部通信机接收请求短消息,完成指定的证券交易,并将处理结果反馈给手机用户。

通信服务器的结构如图 2 所示。为了提高通信处理的并发性,系统设置了向营业部发送和从营业部接收两种通信线程。向营业部发送线程负责从接收队列中取出接收的手机短消息,并转发给相应的营业部通信机;从营业部接收线程负责接收营业部的处理结果,并存放到发送队列中等待下发手机。另外,为了避免与一个营业部的通信故障引起整个通信服务器的崩溃,通信服务器针对每个营业部分别设置了一对发送和接收通信线程。

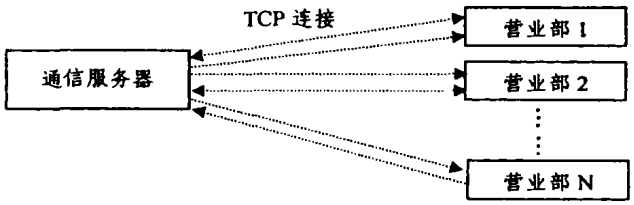

图 1 移动证券业务中证券公司的通信平台

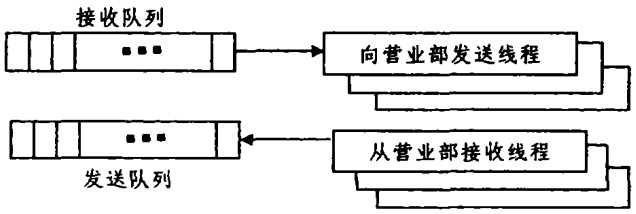

图 2 通信服务器的结构

营业部通信机的结构如图 3 所示。为了提高并发性,设置了一个接收线程和一个发送线程,接收线程负责接收通信服务器转发的手机短消息,并存放到接收队列中等待处理;发送线程从发送队列中取出处理结果,并发送给通信服务器。

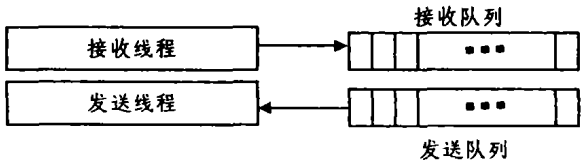

图 3 营业部通信机的结构

2.2 问题描述

由于阻塞方式的 Socket 通信编程比较简单,因此通信服务器和营业部通信机之间的 TCP 连接采用了阻塞方式的 Socket 通信。

首先,通过 Socket 通信,通信服务器的每个发送线程与某个营业部的接收线程之间建立一个 TCP 连接;通信服务器

*)本文受到国家 863 课题资助(编号:2001AA112030)。胥光辉 博士后,研究方向为计算机网络管理、软件测试等。徐永森 教授,研究方向为软件理论、软件工程等。

的每个接收线程与某个营业部的发送线程之间建立一个TCP连接。

然后，通信服务器的发送线程调用Socket的Send函数将收到的手机短消息发送给对方，对方(营业部接收线程)调用Socket的Receive函数接收手机短消息数据。营业部发送线程调用Socket的Send函数将短消息处理结果发送给对方，对方(通信服务器接收线程)调用Socket的Receive函数接收短消息处理结果数据。

处于阻塞模式的Socket的函数被调用之后，在完成所有动作之前不会返回。即，Socket的Send函数只有将数据发送完毕之后才会返回，而Socket的Receive函数只有在收到了所需数据之后才会返回。当通信服务器和营业部通信机的接收线程调用了Socket的Receive函数之后，该接收线程就会处于阻塞状态，等待对方的发送线程发送数据。由于不知道对方何时发送数据，因此所需等待的时间将是不确定的。

现在的问题是当发生了某种特殊情况，例如管理员发出通信中止命令，需要发送和接收线程处理时，如何通知(唤醒)处于阻塞状态的通信线程？因为发送线程只是在发送数据准备好之后，才会调用Socket的Send函数，而一旦发送完毕，Socket的Send函数就会立即返回。因此，发送线程处于阻塞状态的时间是短暂的，唤醒发送线程的问题不大。困难在于如何唤醒调用了Socket的Receive函数而处于阻塞状态的接收线程，通过发送消息来唤醒它是不可行的，因为处于阻塞状态的线程不参与处理机调度，不可能接收和处理消息。

3 唤醒方法

只有当线程正在等待的条件满足时，操作系统才会将线程的状态由阻塞状态转变为就绪状态。因此，唤醒阻塞线程只能从引起线程阻塞的等待条件入手。具体到上面的问题，考虑通信服务器和营业部通信机之间的两对发送和接收线程，如图4所示。

假设通信服务器的管理员发出服务停止命令，这条命令引起向通信服务器的发送线程和接收线程发出服务停止事件(1)。

发送线程很快能够处理这个事件，释放占用的资源，关闭与营业部接收线程之间的TCP连接，发送线程终止；而接收线程这时可能处于同步阻塞状态，因此，服务停止事件被保存在接收线程的消息缓冲队列中等待处理。

通信服务器发送线程关闭TCP连接的动作使得对应营业部通信机的接收线程产生一个通信异常事件(2)。

营业部通信机接收线程的异常处理部分发现这个异常事件之后，关闭当前的TCP连接，然后重新打开TCP端口，等待通信服务器发送线程的下一次连接请求(请注意，该线程不能被终止，因为连接中断是由于对方停止服务引起的，对方随时可能重新启动服务而请求与本线程再次建立TCP连接)；同时向本营业部通信机的发送线程发送一个连接中断事件(3)。

营业部通信机的发送线程很快可以处理这个连接中断事件，处理过程与接收线程类似，关闭当前的TCP连接，然后重新打开TCP端口，等待通信服务器接收线程的下一次连接请求(该线程同样不能被终止)。

营业部通信机发送线程关闭TCP连接的动作将引起通信服务器的接收线程产生一个通信异常事件(4)。

这个通信异常事件引起处于同步阻塞状态的通信服务器的接收线程被激活，该接收线程发现消息缓冲队列中等待处理的服务停止事件，就关闭当前的TCP连接，释放占用的资源，然后接收线程终止。

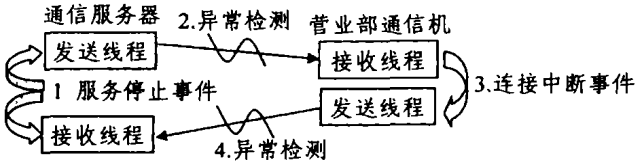

图4 通信服务器接收线程的唤醒流程

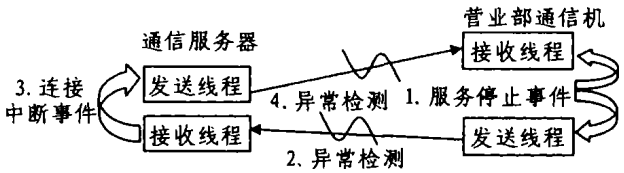

图5 营业部通信机接收线程的唤醒流程

当营业部通信机的管理员发出服务停止命令时，处理流程如图5所示，可以看出图5恰是图4的对称处理过程。

TCP连接的一端通知另一端关闭连接的方法除了采用上述的异常检测外，还可以设计一种应用层协议(如FTP协议)，将传输的报文分成命令报文和数据报文两种，通过发送"关闭连接请求"命令报文通知对方关闭TCP连接。由于目前流行的编程语言，如Java，Visual C++等，都支持异常处理，因此采用异常检测的方法更为方便。

4 震荡现象

采用上述方法可以解决同步阻塞线程的唤醒问题，但在实际运行中却出现了令人费解的震荡现象。即，如果停止服务后再次启动服务，就会出现发送线程和接收线程的撤销、创建、撤消、创建的震荡现象。即使增加了延时机制，震荡现象也不能彻底消除，只是震荡次数有所减少，一般需要经过2～3次震荡才能稳定下来，建立起正常的通信。

经过仔细地跟踪调试，我们发现震荡现象是由于接收线程向发送线程发送了不必要的连接中断事件而引起的。图4给出的是通信服务器停止服务的处理流程，图5给出的是营业部通信机停止服务的流程。两个处理流程都是正确的，但是叠加起来却出现了一个Bug。这个Bug就是任一方的接收线程在检测出通信异常事件时，都会向本方的发送线程发送一个连接中断事件，而不管是否收到过服务停止事件。实际上，如果收到过服务停止事件，按照处理流程，本方的发送线程这时已经终止了，接收线程不应该再向它发送连接中断事件。

震荡的真正原因找到之后，相应的解决方法是简单的，即，接收线程在发送连接中断事件之前，首先检查是否收到了服务停止事件，如果收到了，则不发送连接中断事件；否则向本方的发送线程发送一个连接中断事件。

结论 本文针对移动证券业务中通信服务器和营业部通信机之间的Socket通信，研究了同步阻塞线程的唤醒问题。提出了一种基于异常检测的同步阻塞线程的唤醒方法，这种方法对于唤醒若干对相互通信的线程非常有效。最后，分析了出现震荡现象的原因，并给出了解决办法。

参考文献

1 Comer D E, Stevens D L 著，赵刚，等译. 用TCP/IP进行网际互连第3卷：客户机-服务器编程和应用(第2版). 北京：电子工业出版社，1998
2 Tanenbaum A S. Distributed Operating Systems(分布式操作系统). 北京：清华大学出版社，1997
3 中国移动通信集团公司. 移动股票交易接口规范(V1.1). 2000