编写人:赵佰枫

日期:2013-10-16

**DTU客户端管理与线程处理**

# 任务和目标

服务端对客户端连接进行高效的管理。定时访问客户端设备运行状态,并对设备控制指令进行及时的发送到客户端。

# 流程示意图

程序启动

开启服务,等待客户端连接

开启线程等待DTU进行注册

接收到连接

接收到注册数据后,保存socket到列表,并结束此线程

注册

启动线程2

循环socket列表,向客户端请求设备运行状态数据

启动线程3

循环socket列表,向客户端请求设备控制命令

启动线程1

启动线程4

循环socket列表,判断客户端是否连接正常,如果连接断开则删除

# 功能设计

**主要线程**

程序中有四个主要的线程，分别为：等待客户端连接线程（线程1），自动向DTU访问设备运行状态线程（线程2），向DTU发送工况配置线程（线程3）,检测socket连接是否正常的线程(线程4)。

**各线程的功能：**

**线程1**：主线程中创建socket服务后，开启线程1，在线程1中利用ServerSocket.Accept()方法监听客户端连接.每当接收到一个DTU连接,则开启一个临时线程,用来监听DTU注册信息,待DTU发送注册信息,调用clientManager类对注册的DTU进行管理,然后关闭此线程。

DTU连接管理类（ClientManager）

|  |  |  |
| --- | --- | --- |
| 方法名 | 参数 | 功能说明 |
| AddClient | (string tel, Socket socket) | 注册客户端 |
| UpdateLastVisitTime | (DTUClientInfo clientInfo, DateTime datetime) | 更新客户端的最后一次访问时间 |
| DeleteClient | (DTUClientInfo clientInfo) | 删除客户端注册信息 |

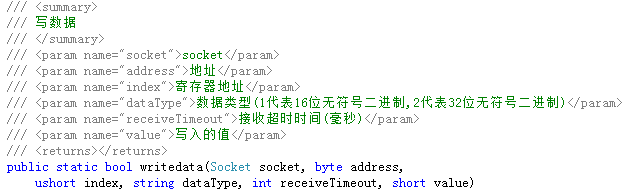
**线程2**: 应用程序启动后,则开启线程2,并设置成后台线程. 每隔一段时间(具体间隔时间在sysconfig.xml文件中配置),循环所在注册过的DTU连接，并调用modbus.readdata()方法读取设备运行状态,保存到数据库。

以下是modbus.readdata方法



生命周期:从应用程序启动到应用程序结束。

**线程3**：应用程序启动后,则开启线程3,并设置成后台线程。每隔一段时间(具体间隔时间在sysconfig.xml文件中配置),循环所在注册过的DTU连接，并根据DTU所对应的数据库，查询出control\_command表中的指令,调用modbus.read()或modbus.write()进行工况配置。



生命周期:从应用程序启动到应用程序结束。

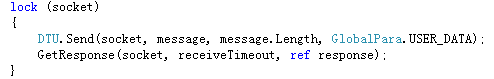
**线程4**:应用程序启动后,则开启线程,循环clinetmanager中的DTU连接,对DTU进行检测,判断连接是否正常.如果连接已断开,则删除这个DTU连接。

检测方法:调用socket.send();方法,如果出现异常,则表明socket连接已断开.

**多线程访问socket同步问题:**

由于有多条线程的存在,存在同时访问socket的现象,所以在发送和接收消息时给socket加锁.

具体需要在Modbus类的SendFc和SendFc16方法内的以下代码加锁:如下图



**存在的缺陷:**

一旦DTU客户端多起来.循环所有的客户端来读取设备运行状态可能需要很长的时间.但如果这个时间超出了DTU的心跳间隔时间,则DTU会向服务端发送心跳.而服务端socket则没在监听.导致DTU主动与服务端断开连接,最终影响设备运行状态的读取.

**补救措施:**

再开一个线程,循环所有的客户端连接,找出最后一次访问时间在两分钟之前的客户端,并给它们发送一次用户数据来保证DTU不向上发心跳数据.