**网络总线设计文档**

| 日期 | 修改记录 | 修改人 |
| --- | --- | --- |
| 2015.8.28 |  | 李进 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[一、简介 3](#_Toc429297321)

[1．概述 3](#_Toc429297322)

[二、总体设计 3](#_Toc429297323)

[1．概述 3](#_Toc429297324)

[2．设计思想 4](#_Toc429297325)

[2.1 模块划分 4](#_Toc429297326)

[2.2 心跳子模块 4](#_Toc429297327)

[2.3 传输子模块 5](#_Toc429297328)

[三、结构说明 8](#_Toc429297329)

[四、接口说明 9](#_Toc429297330)

[1．总线接口 9](#_Toc429297331)

[1.1 StartNetBus 9](#_Toc429297332)

# 一、简介

## 1．概述

本系统即支持单机模式运行，也可支持分布式部署。在多机环境下，系统支持**双机**热备冗余，同时系统支持**双网**热备冗余。

本文档主要适用于SCADA与FES、SCADA之间、FES之间交换数据，单机模式时（SCADA与FES运行在一台机器上），通过本地IPC进行数据传输。分布式模式时，通过网络进行通信。

网络总线负责系统内部各节点间数据的传输，以及节点状态的判定、维护等。网络总线模块在系统架构中如下图所示：

图

# 二、总体设计

## 1．概述

系统中网络节点类型如下表：

|  |  |  |
| --- | --- | --- |
| 类型 | 缩写 | 备注 |
| 前置机 | F |  |
| 主服务器 | S |  |
| 工作站 | C | 运行操作工作站、维护员工作站、报表工作站等 |

每个节点的工作状态如下表：

|  |  |  |
| --- | --- | --- |
| 状态 | 说明 | 备注 |
| 启动态 |  |  |
| 从机 |  |  |
| 主机 |  |  |
| 申请切换 |  |  |
| 未知 |  |  |
|  |  |  |

## 2．设计思想

### 2.1 模块划分

网络总线部分分成两个子模块：

**网络心跳子模块：**

主要负责网络各节点状态的更新、判断、维护管理，各节点之间时间的同步管理。

**数据传输子模块：**

负责F\S\C之间的数据传输。

### 2.2 心跳子模块

系统每个节点定时（**1秒**）发送UDP广播报文作为心跳信号，心跳信号内容见下文。 同时，每个节点也从网络上接收其他节点发出的广播心跳报文，如果是同一项目内的对时从节点（如果采用其他对时方式，则忽略），则更新本地时间。此外，更新系统的节点状态表。

节点状态的管理。

通过心跳在本地内存中维持一份各个节点的存活状态。

心跳报文

|  |  |  |
| --- | --- | --- |
| 名称 | 长度 | 备注 |
| 特征码 |  |  |
| 报文序号 |  |  |
| 主从状态 |  |  |
| 节点名称 |  |  |
| 本机时间 |  |  |
| 健康状态 |  |  |
| 项目名称 |  |  |
| 校验码 |  | 和校验 |

1. 心跳流程图



### 2.3 传输子模块

网络程序启动后，首先创建接收和发送队列，然后确定本节点的角色，如服务器、工作站、前置机等以及是否需要判断主从状态。接着启动报文的接收、发送服务线程，以及“心跳”发送、接收的服务线程，接着主线程循环判断节点的通讯状态和主从状态。接收线程将收到的报文写到接收队列中，发送线程则从发送队列中取出报文发送到相应的节点。

报文

初始化

主线程

监视节点状态

接收线程

发送线程

“心跳”线程

接收队列

发送队列

报文

外部节点

网络通讯程序基本原理

总线启动后，各个节点之间的链路状态总线需要自动检测，以及各个节点的状态需要实时记录。

* 搜索网络节点表，确定是否存在主从节点
* 启动发送、接收、“心跳”等服务线程
* 如果是单机，直接注册为主
* 否则根据对方状态进行判断
  + 如果对方为启动态，则本机进入申请态
  + 如果对方为从，本机默认为主，则进入申请态
  + 如果对方为从，本机默认为从，则注册为从
  + 如果对方为申请态，则本机注册为从
* 下面进入循环
* 如果监听端口未启动，但该网络可用，则进行监听(SockListen)
* 判断本机网络状态
* 测试报文处理
* 如果单机，则返回
* 否则进入主从切换判断
  + 当本机为主时，若本机网络断开，或者对方为主且自身非默认主，则注册为从
  + 当本机为从，或者本机为启动态，而对方优先级比本机低时，则本机升为申请态
  + 当本机为申请态时，如果判断时间满足，则注册为主
  + 若对方为主，则自身注册为从
  + 若对方为也为申请态，且对方为默认主，则本机注册为从
  + 若本机网络断开，注册为从

**工作站与主服务器**

工作站作为客户端主动连接主服务器（主服务器作为服务端）。

**主服务器与前置机**

各前置作为客户端与主服务器连接（主服务器作为服务端）。

**工作站与前置机**

工作站不与前置机直接连接。

**主从机之间**

主从机的状态一旦确定，从机就连接主机

①、前置 和 scada 之间

总线运行在前置时作为客户端（C1），运行在scada作为服务端。如果有多个scada，需要根据配置确认前置是不是向这些scada全部发送数据。

客户端和服务端需要2个通道，一个是normal（普通），即普通遥测遥信信息。一个是urgent（紧急），遥控命令，事项，告警，SOE。Scada向fes发送遥控命令时，走的是urgent通道。

②、前置主备之间

此模式下，总线启动时，即是主又是从。建立双向通道。

③、scada和工作站之间

待讨论

④、双网卡

双网卡时，总线同时open（）2个服务，一个为主一个为从。主断链后，从自动切换为主。之前的主变成从，并且自动进行链路修复。

主备切换

根据接收到的主机名，判断接收到的主从状态信息，如果放生改变，通知主从切换管理程序进行主备切换。

2.4 主备切换管理

1. 流程说明
2. 流程图



网络报文格式如下表：

|  |  |  |
| --- | --- | --- |
| 名称 | 长度 | 备注 |
| 消息类型 |  | 见下表 |
| 消息优先级 |  | ？ |
| 报文序号 |  |  |
| 目标节点名称 |  |  |
| 目标模块名称 |  |  |
| 源节点名称 |  |  |
| 源进程名称 |  |  |
| 功能码 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

消息类型

|  |  |  |
| --- | --- | --- |
| 码 | 功能 | 备注 |
| 3 | 到后台 |  |
| 1 | 到对侧前置 |  |
| 2 | 到其他主前置（网络服务用） |  |
| 11 | 发往所有后台 |  |
| 12 | 发往对侧节点 |  |
| 13 | 发往别名为主 需目的节点别名 |  |
| 14 | 发往主服务器 |  |
| 15 | 发往指定节点（包括主从） 需目标节点别名 |  |
| 16 | 发往指定节点（点对点发送） 需目标节点主机名称（发送文件时不需要） | 11-16 均需要填写目的进程名称，透明通道用 |
|  |  |  |
| 17 | 发往主服务器SCADA |  |
| 30 | 传输文件（仅支持点对点传输） |  |
|  |  |  |

程序通过“心跳”报文判断当前正在运行的网络节点，并采用一定的连接规则使各个节点之间进行相互连接，在完成连接后，为防止网络的意外中断或者可恢复的中断发生，每个节点在套接字的两端发送测试报文。在意外中断或暂时中断的情况下仅周期性地进行尝试连接而不进行数据的接收和发送，并等待进一步的网络节点判断。

# 三、结构说明

1、节点信息定义：

struct

{

int nNodeID; //节点在数据库中的ID

int nNodeType; //节点类型

int nHostSlave; //主从状态

char chIP[2][NAME\_SIZE+1]; //AB网IP

int nCardStatus[2]; //AB网状态

char chHostName[NAME\_SIZE + 1]; //主机名称

char chNickName[NAME\_SIZE + 1]; //别名

char chNodeName[NAME\_SIZE + 1]; //节点名

BOOL bDefaultHost; //是否默认为主

} NODE\_CONF;

2、消息头：

Struct

{

Char chBegin[2]; // 消息头 0x5a 0xa5

int nMsgType; // 消息类型

int msg\_len; // 消息长度

char chEnd[2]; // 消息尾 0xa5 0x5a

} MSG\_HEAD;

3、端口宏定义

#define NETBUS\_DGRAM\_ LISTEN\_PORT 21001 // 广播监听端口

#define NETBUS\_DGRAM\_ SEND\_PORT 21002 // 广播发送端口

#define NETBUS\_TCP \_LISTEN\_PORT 21003 // 多机TCP监听端口

#define NETBUS\_TCP \_SEND\_PORT 21004 // 多机TCP发送端口

# 四、接口说明

## 1．总线接口

此类接口由其他模块调用。

### 1.1 StartNetBus

BOOL StartNetBus (NetConf \*pConf);

**描述：**

启动总线

**参数：**

pConf 网络表

pVal 获取到的模拟量返回值

**返回：**

IO\_OK 成功

IO\_ERROR 失败

**约束：**

无

// 断开连接

BOOL stopNetBus();

// 初始化

int Init();

// 结束

Void Fini();

// 发送接口

Long Send(const void\* msg, unsigned int len, unsigned long ulTimeout);

// 接收接口，回调的形式实现接收接口

bool Receive(const void\* msg, unsigned int& len, unsigned long ulTimeout);

// 同步文件接口

Long sendFile();

// 进程间通信接口

接口形式和socket一样，可以加个mode，或者另外提供接口。

Long ipcSend();

Long ipcRecv();

**五、内部设计**