奶牛监控数据收集网关软件系统技术路线分析 V1.0

1. 系统简介

1.1 系统功能

本系统用于奶牛监控数据的收集和转发。系统首先从串口读入从无线通信模块接收到的传感数据，在本地进行初步的整合后，通过有线局域网，将数据批量的同步到远端的服务器进行深度的数据分析。

1.2 系统开发环境

宿主机环境

操作系统：Linux

软件库支持：XSI IPC

开发工具：VIM 等文件编辑软件

ARM 平台的交叉编译工具链

目标板环境

操作系统：Linux

软件库支持：rsync 命令、XSI IPC

2. 系统架构说明

2.1 系统框架

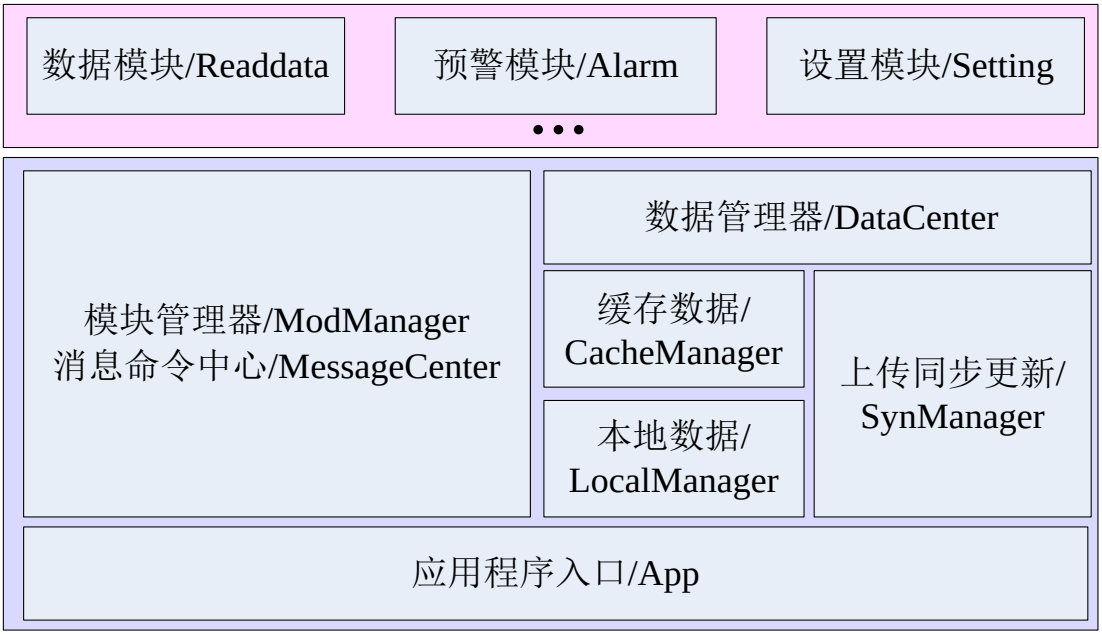

图 2.1 系统框架

参考 MVC 架构，本系统划分为数据层、控制层、应用层三大模块进行设计。

(1) 数据层是本系统设计的重点，负责对接收过来的传感数据进行缓存、整合、存储、同步等重要操作。本系统设计一个 DataCenter 统一对数据进行管理，保证数据的一致性和有效性。还要提供清晰的 API 供系统其他各模块调用。

(2) 控制层负责对系统进行启动初始化和运行的管理工作。控制层有两个主要的子模块，他们分别是模块管理器（ModManager）和消息命令中心（MessageCenter）：

a. 模块管理器是提供应用层调用的子系统，负责对各个应用子模块进行加载和管理，失各应用模块有序稳定的运行。

b. 消息命令中心是对系统添加的一个消息通信系统，实现系统内部的各子模块的数据通信和对远端服务器的命令响应。

(3) 应用层中包含各应用子模块，供网关使用者调用。他们统一继承应用层的模块基类，由控制层的模块管理器（ModManager）进行加载和启动。

对传感数据进行接收处理的数据模块就是作为一个应用子模块工作在这一层，提供最直接的系统应用。

2.2 启动流程

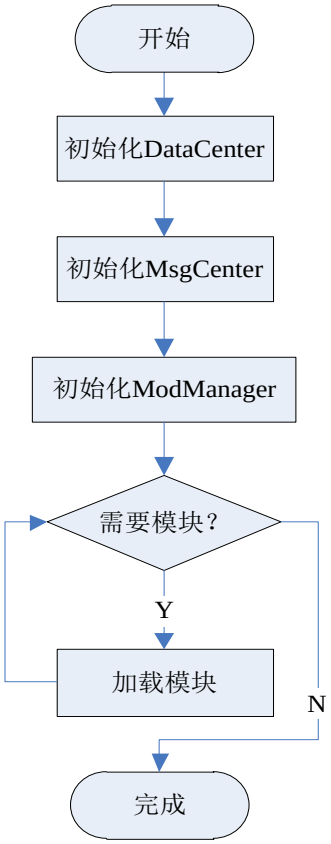

图 2.2 系统启动流程

系统启动的时候，首先启动并初始化 DataCenter，保证数据管理器加载成功。再进行后续的控制层加载，在加载控制层的时候，首先启动 MessageCenter，MessageCenter 加载成功后，就可以对系统注册各消息事件。接着启动 ModManger，ModManger 也加载成功后，就可以加载应用层的各个应用子模块。至此，网关系统启动完毕。

3. DataCenter

3.1 模块定义

为了系统管理和组织收集回来的传感数据，需要添加一个数据管理中心（DataCenter），以统一对数据进行缓存、存储、同步等操作。

3.2 模块框架

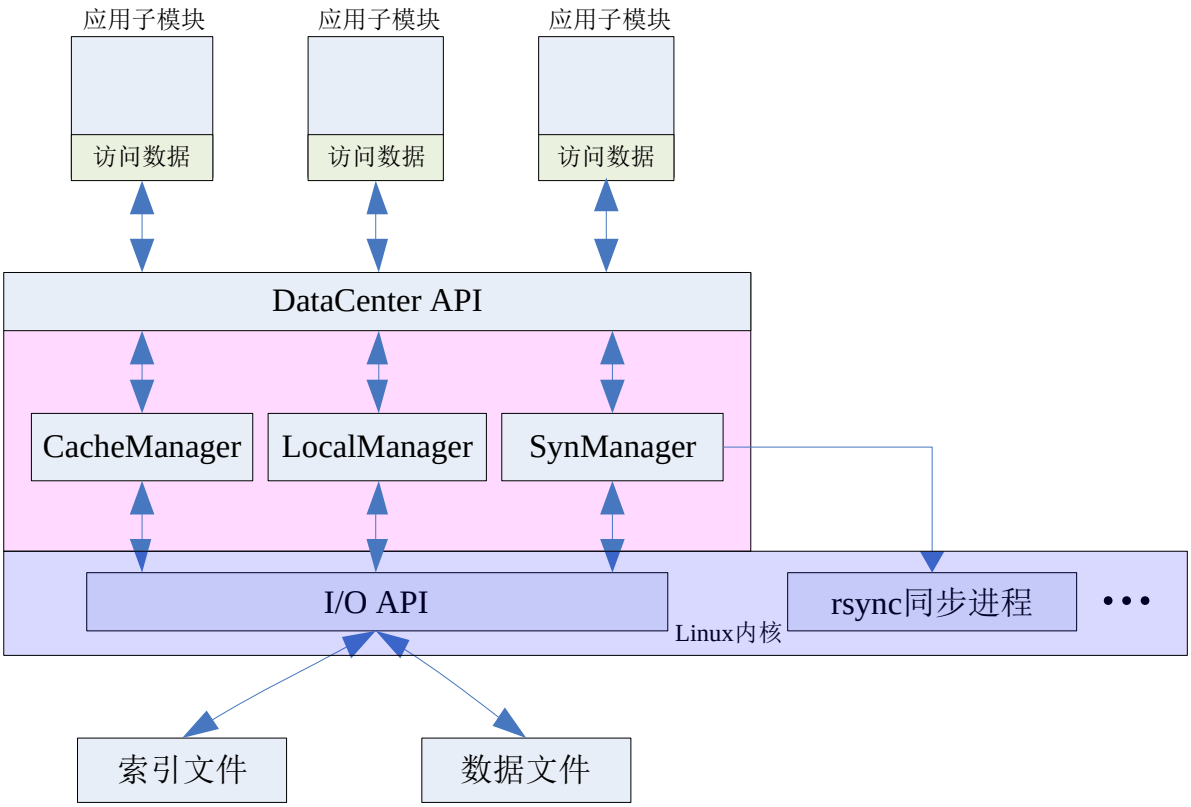

图 3.1　DataCenter 模块框架

1、 DataCenter 启动：

a) 启动 CacheManager
b) 启动 LocalManager
c) 启动 SynManager

2、 访问数据

a) 当应用子模块需要对网关系统的数据进行操作（包括存取数据、保存文件、同步到远端服务器等）的时候，就会调用 DataCenter 提供的 API 进行访问。
b) DataCenter 根据实际功能调用 CacheManager、LocalManager 和 SynManager 的接口
c) CacheManager、LocalManager 和 SynManager 调用 Linux 内核提供的 I/O API 进行实际的数据操作。而 SynManager 还会调用 rsync 将数据同步到远端服务器。

3.3 模块功能

3.3.1 缓存管理

CacheManager 提供对缓存数据的各项操作，包括数据写、数据读已经后续扩展的数据分析。

接口 API

write_cache_data 将数据写入缓存

read_cache_data 根据索引读取指定的缓存数据

delete_cache_data 根据索引删除指定的缓存数据

modify_cache_data 根据索引修改指定的缓存数据

fliter_cache_data 对缓存数据进行过滤（主要对无效数据或错误数据的检查）

3.2.2 文件管理

localManager 提供对数据进行本地存储和管理的操作

接口 API

load_data 将文件数据导入内存

save_data 将指定的缓存数据保存

delete_data 删除指定的数据文件

copy_data 复制指定的数据文件

move_data 移动指定的数据文件

backup_data 备份指定的数据文件（打成 tar 包，放进 backup/目录）

3.2.3 数据同步

SynManager 提供对数据进行同步到远端服务器的操作。当本地存储文件多于一定数目或者在指定的时间间隔后，执行数据同步操作，也提供同步接口供调用者使用同步。

接口 API

set_setting 设置 setting 数据（包括同步时间间隔和文件数量阈值等）

get_setting 获取 setting 数据（包括同步时间间隔和文件数量阈值等）

sync 同步数据到远端服务器

4. MessageCenter

4.1 模块定义

为了组合各个模块协同工作，需要对整个系统添加通信机制。各个需要处理消息的模块，都需要注册事件处理回调函数给 MessageCenter。

4.2 模块框架

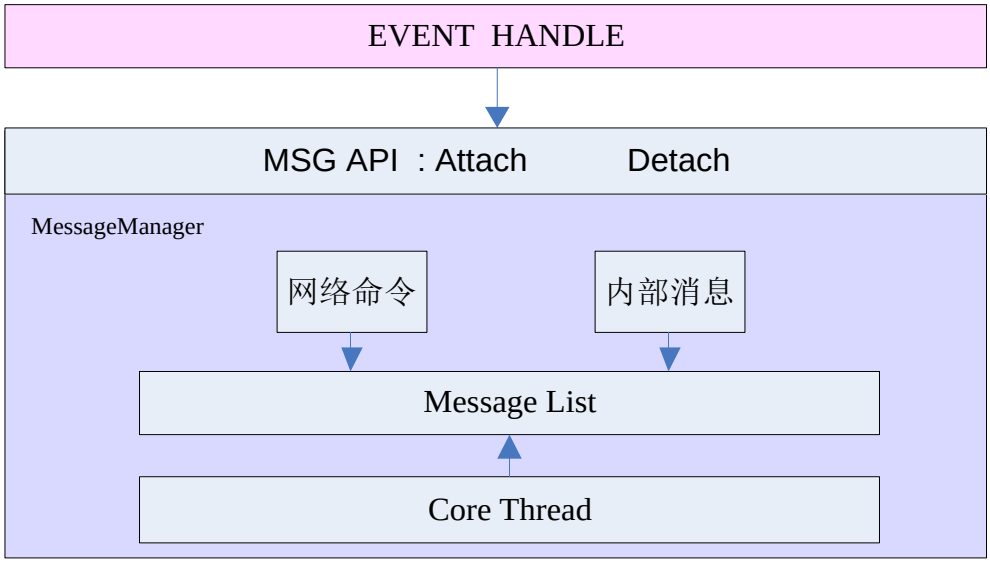

图 4.1 MessageManager 模块框架

1、 MessageManager 启动：

a) 初始化 Message List

b) 开启网络命令监听，创建网络 socket 服务器进程，接收从远端服务器发送过来的 socket 数据包。

c) 开启内部消息监听，创建 domain socket 服务器进程，接收从系统内部各子模块发送过来的消息数据包。

d) 启动 Core Thread 从 Message List 读取 message

2、 EVENT HANDLE 关联对应的事件

a) 添加新的消息处理函数，调用 MessageManager 的 API：Attach

b) 对不再需要处理的消息，调用 MessageManager 的 API：Detach

3、 封装消息

a) 接收到网络命令，封装为消息对象，压入 Message List

b) 接收到系统内部消息，封装为消息对象，压入 Message List

4、 事件处理

a) Core thread 从 Message List 获取到消息对象

b) 将事件与特征比对找到对应的 Handler

c) 调用 message 的信息处理函数

4.3 模块功能

4.3.1 接受系统内部消息

MessageCenter 调用 Linux 内核 domain socket 机制，创建系统内部消息监听服务器，能接收系统内部各个模块发来的消息。

接口 API

open_inner_server　创建系统内部消息监听服务器

inner_listen　开启系统内部消息监听进程

inner_accept　从系统内部消息监听服务器中读取一个消息。如果没有消息发来，则阻塞进程；如果有数据发来，则返回消息类型（如果调用时传入一个非空结构体 param，则返回带上消息的相关参数）

send_msg　向系统内部消息服务器发送一个消息（提供给系统各模块调用）

close_inner_server　关闭系统内部消息监听服务器

4.3.2 接受网络通信命令

MessageCenter 调用 Linux 内核 socket 机制，创建网络端口监听服务器，接收远端服务器的网络通信命令。

接口 API

open_web_server　创建网络监听服务器

web_listen　开启网络端口监听进程

web_accept　从网络端口读取一个 socket 数据包。如果没有消息发来，则阻塞进程；如果有数据发来，则返回消息类型（如果调用时传入一个非空结构体 param，则返回带上消息的相关参数）

close_web_server　关闭网络监听服务器

4.3.3 统一封装消息对象

无论接收到的是系统内部的消息还是网络通信命令，MessageCenter 都会统一封装为一个消息对象，压入消息等待队列。

接口 API

parse_message　根据传入的消息类型和相关参数，创建一个消息对象并返回

push_message　将消息对象插入 Message List

pop_message　读取 Message List 的对头消息对象

5. ModManager

5.1 模块定义

为了统一管理各个应用子模块，需要对整个系统添加应用模块管理器（ModManager）。在数据层和控制层都加载完毕后，由 ModManager 根据实际需求加载各个应用子模块。如果在系统运行过程中不再需要某个应用子模块，也可以调用 ModManager 进行卸载。

5.2 模块框架

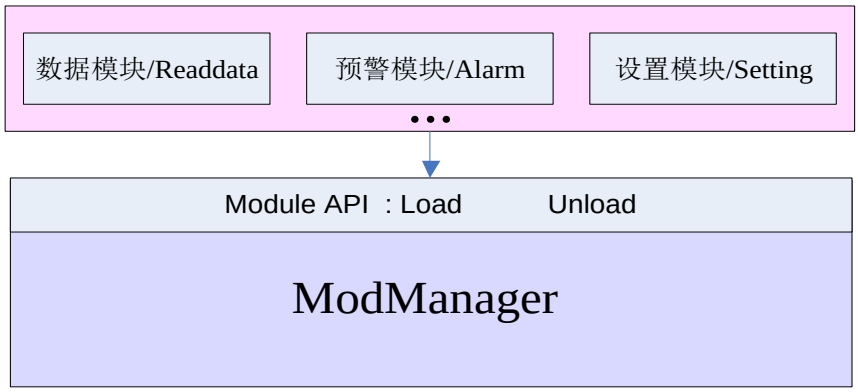

图 5.1　ModManager 模块框架

1、 ModManager 启动：

a) 加载数据模块

b) 根据实习需求加载其他应用子模块

2、 运行过程中对各应用子模块的管理

a) 加载新的应用子模块，调用 ModManager 的 API：Load

b) 卸载不再需要处理的应用子模块，调用 ModManager 的 API：Unload

5.3 应用子模块

应用子模块提供一个模版基类，其中定义一些由 ModManager 加载、卸载的管理机制。所有新定义的应用子模块都需要继承这个基类。

5.3.1 数据模块/Readdata

由于需要监控串口端是否有数据发来，需要创建一个应用子模块作为一个独立进程启动，不停轮询端口是否有数据可读。如果扫描到有数据可读，数据模块就会创建一个子进程单独与这个串口进行数据接收，并将接收到的数据写入系统缓存。工作原理如图 5.2 所示：

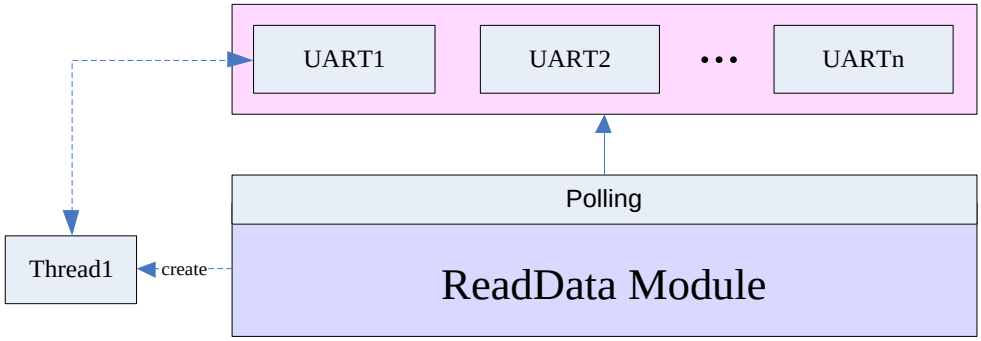

图 5.2 Readdata Module 工作原理

5.3.2 预警模块/Alarm

处理预警工作……

5.3.3 设置模块/Setting

处理网关控制命令的发布……

……