独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：孙秀云　　日期：2010、3、10

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：孙秀云　　日期：2010、3、10

导师签名：　　日期：2010、3、10

XXXX卫星实时传输子系统的设计与实现

摘 要

XXXX卫星地面通信系统承担着卫星原始资料、卫星测控和轨道数据、指挥调度信息等数据的远程传输任务，对我国的XXXX卫星事业有着举足轻重的作用。其中，通信分系统主要负责实时高效、安全稳定的卫星原始数据传输工作。通信分系统应用软件划分为两个子系统：实时传输子系统（XXXX01）、通信设备监控子系统（XXXX02）。实时传输子系统主要提供转发端至接收端的高效的可靠的数据文件的传输功能；通信设备监控子系统用于实现调制解调器/路由器、通信收发器和调制解调器等通信设备的远程监控和告警消息的发送功能。其中，实时传输子系统运行在转发端和接收端的UNIX通信服务器上，通信设备监控子系统运行在转发端和接收端的监控WINDOWS PC上。实时传输子系统能为整个通信分系统提供高效可靠的数据传输。

论文针对现有实时传输子系统的实际需求，在分析与研究通信模型的基本理论和方法的基础上，提出并建立了实时传输子系统的通信模型，给出了实时传输子系统的总体结构设计，并基于此通信模型，详细阐述了实时传输子系统发送端和接收端的功能设计和通信接口及通信协议的设计，包括模块划分、模块交互的内部接口、模块功能实现、系统间交互的外部接口等。根据以上的系统设计，将该实时传输子系统在UNIX平台下编程实现。

根据实际应用需求，对实时传输子系统进行了系统地测试，测试结果表明该系统能够满足整个卫星通信系统对其的功能需求及性能需求。论文还针对该实时传输子系统的优化提出了相关建议。

关键词：实时传输 通信模型 消息队列 多线程 UNIX

THE DESIGN AND IMPLEMENTATION OF XXXX SATELLITE REAL-TIME TRANSMISSION SUBSYSTEM

ABSTRACT

The XXXX satellite ground communication system is responsible for the transmission of original satellite data, the measurement and control data and the dispatch command. This system plays a vital role in the XXXX satellite career of our nation. As a part of the system, the communication subsystem aims at making real-time, efficient, secure and stable transmission of original satellite data between different sites. The communication subsystem is divided into two parts: the real-time transmission subsystem (XXXX01), which provides efficient and reliable transmission functionality from relay to receiver site; the communication equipment surveillance subsystem (XXXX02), which enables remote surveillance and notification message for modem, router, communication transceiver equipment. The real-time transmission subsystem is running on UNIX communication server installed in relay and receiver site, and the communication equipment surveillance subsystem takes Windows PC as its running environment. The real-time transmission subsystem can provide efficient data transmission for the whole communication subsystem.

Based on the analysis and research to the basic theory and method of communication models, according to the actual requirements of the real-time transmission subsystem, we bring forward and establish the communication model of the real-time transmission subsystem, give general structure design of the system, and based on the communication model, we explain the function design of the transmitter and receiver and the design of communication interfaces and communication protocols which include module devision, inner interface of module level interaction, implementation of the module function, outer interface of system level interaction and so on. According to above system design, we implement the real-time transmission subsystem under UNIX platform.

According to the actual application requirements, we implement a systematical

test to the real-time transmission subsystem whose result shows that the subsystem has fulfilled both functionality requirements and performance requirements required by the whole satellite communication system. Finally, we give some advice on the optimization of the real-time transmission subsystem.

KEY WORDS: real-time transmission, communication model, message queue, multi-thread, UNIX

目 录

第一章　绪论

1.1 课题背景

XXXX卫星地面应用系统是一个实时、多功能的遥感卫星的业务运行系统。其主要目的是完成对XXXX卫星（和/或其他遥感卫星）遥感数据接收、数据远程传输、数据预处理、产品制作、数据存档和产品分发、定标和真实性检验、产品应用和示范等业务处理工作，为各类用户提供XXXX卫星遥感数据产品与服务。

该系统由以下几个分系统组成：接收预处理分系统；资料处理分系统；资料存档与产品分发分系统；辐射定标和真实性检验分系统；产品应用与示范分系统；通信分系统；运控分系统。

通信分系统承担着卫星遥感数据传输、网络监控以及传输信道切换的任务，以保证XXXX地面站遥感数据高可靠、实时传输和网络监控，该分系统是整个地面应用系统的“通信中枢”。该分系统由若干台高可用性计算机系统、高性能网络和各种通信设备以及“XXXX卫星地面应用系统通信分系统应用软件子系统”组成。

通信分系统应用软件划分为两个子系统：实时传输子系统（XXXX01）、通信设备监控子系统（XXXX02）。实时传输子系统主要提供发送端至接收端的高效的可靠的数据文件的传输功能；通信设备监控子系统用于实现调制解调器/路由器、通信收发器和调制解调器等通信设备的远程监控和告警消息的发送功能。其中，实时传输子系统运行在发送端和接收端的UNIX通信服务器上，通信设备监控子系统运行在发送端和接收端的监控WINDOWS PC上。

发送端-接收端之间的通信信道采用光纤和VSAT两条专线的主备工作方式。

实时传输子系统分为发送端实时传输子系统和接收端实时传输子系统。接收端实时传输子系统主要通过光纤链路和发送端实时传输子系统进行通信，在光纤链路故障状态下采用备用的卫星链路进行通信。每个实时传输子系统需要与本地的通信设备监控子系统通信。实时传输子系统将传输状态、告警信息等传递给通信设备监控子系统并加以显示。

根据本系统的设计需求，概括出实时传输子系统的设计目标如下：

1、在光纤链路正常的情况下，实现高效的实时传输，并按优先级分配各个实时卫星数据的传输带宽；

2、在光纤链路断开的情况下，采用卫星链路实现高效的数据文件的传输；

3、有效判断链路是否拥塞，能在重新启动后实现断点续传；

4、按要求实现传输状态和异常状态的即时报告；

5、整个系统能高效、稳定、可靠地运行。

实时传输子系统（XXXX01）拓扑图如下：

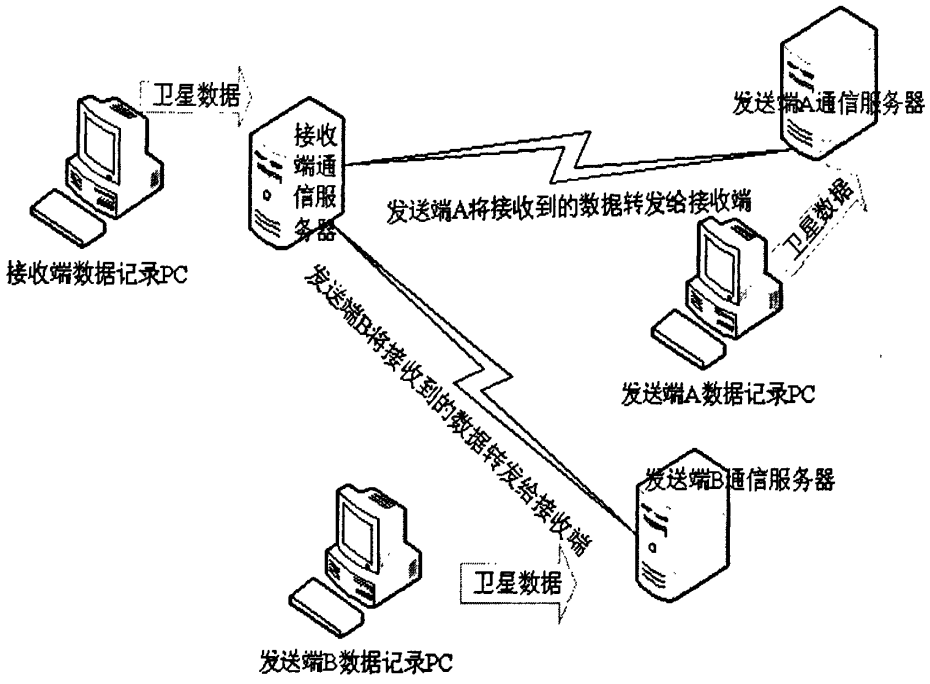

图1-1 实时传输子系统拓扑图

1.2 课题研究的目的和意义

本次研究具有重要的意义，可以设计并实现实时传输子系统，实现传输任务调度、实时数据静态备份、数据传输状态报告、数据传输状态监视、工作日志等功能，使光纤和卫星链路共同协作，保证卫星数据的高效可靠的传输，为整个XXXX卫星通信分系统起到中间桥梁的作用，最终为各类用户提供卫星遥感数据产品与服务，并为以后的系统扩展打好基础。

1.3 课题的主要研究内容

为了实现XXXX卫星实时传输子系统网络数据的高效可靠传输，我们分析现有模型及技术，并结合项目需求，设计适合本系统的模型，并在UNIX平台下实现，最后，对系统的功能和性能等进行了测试。

本论文一共由六章组成：

第一章绪论，介绍了课题背景，提出了本课题研究的目的和意义，并介绍了课题的主要研究内容。

第二章通信模型及相关技术，该章对通信模型的通信模式（对称通信和非对称通信）、传输方式（面向连接和无连接）和对话模式等进行了比较分析，详细了解它们的原理及优缺点。还对本项目中应用到的技术进行了分析。

第三章需求分析，对本项目的用户需求进行了介绍和分析，包括功能和性能的需求等。

第四章系统设计与实现，该章根据实时传输子系统的需求，对其进行交互模型的分析，设计发送端和接收端的交互协议，设计交互接口以及各端对数据的处理方法。对系统进行设计，包括基本组成要素的设计，系统框架的概要设计，以及系统的详细设计。根据以上的系统设计，将此实时传输子系统在UNIX平台下编程实现。

第五章系统测试，利用各类数据，对实时传输子系统进行测试，经过分析，它已经满足预期的功能和性能要求，以此对系统的设计进行验证。

第六章总结与展望，该章对本课题的研究工作和论文进行了总结评价，并对后续工作进行了展望。对以上系统的设计和实现进行分析，不足的地方有待以后改进。另外，一些在当前项目中尚未实现的功能，也有待在以后的开发工作中在已实现系统的基础上增加并实现，逐步完善整个系统。

第二章　通信模型及相关技术

2.1 通信模型分析

2.1.1 实体工作模式

1、主从模式

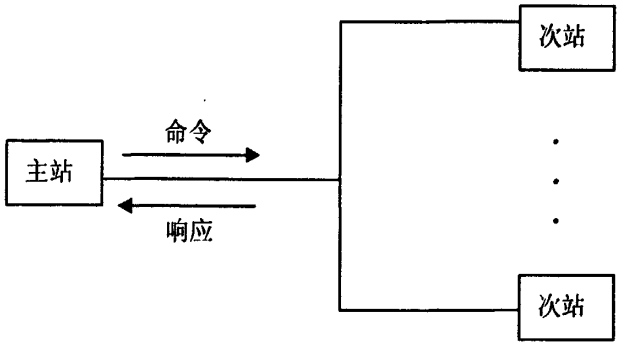

图2-1 主从模式

如上图，主从模式有一个主站、一个或多个次站，主站和次站的地位不对等，双方的信息流是不一样的，主站只可以作为主站，不可以转化为次站，次站亦不可以转化为主站。主站可以向次站发送命令消息，而次站则不可向主站发送命令消息，只可以发送响应消息。对于C/S结构的应用，就应该采用这种工作模式。

2、平衡模式

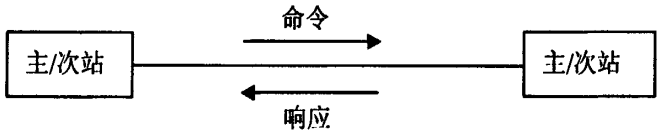

图2-2 平衡模式

如上图，平衡模式中，各站地位是对等的。每个站都既是主站，又是次站，每个站都可以发送命令消息和接收响应消息，也可以接收命令消息和发送响应消

息。主从模式并不能适用于所有场合，对于各站地位对等的各项应用，就需要采用平衡工作模式。

比如，我们来分析一下CPI-C。

CPI-C就是Common Programming Interface-Communications（通用程序通信接口），它与APPC（Advanced Program to Program Communication，先进程序间通信）相联系。APPC没有提供通用的应用编程接口，不能跨平台，必须为不同系统编写不同的代码。CPI-C帮助APPC解决了这个问题，它是一个子程序包，有一套通用的代码，实现了跨平台。CPI-C是一个同步的通信模式，采用双向交互。进行通信的某一方程序首先发起一次对话请求，并且它可以控制信息的流动，如下图。然后，对话发起者可以向对方程序发送数据，而对方程序则可接收数据。接收方在收到数据之后，可向发送方发送确认信息，发起方接收此信息。消息收发完成后，此次对话的发起方断开此次连接，本次对话结束。

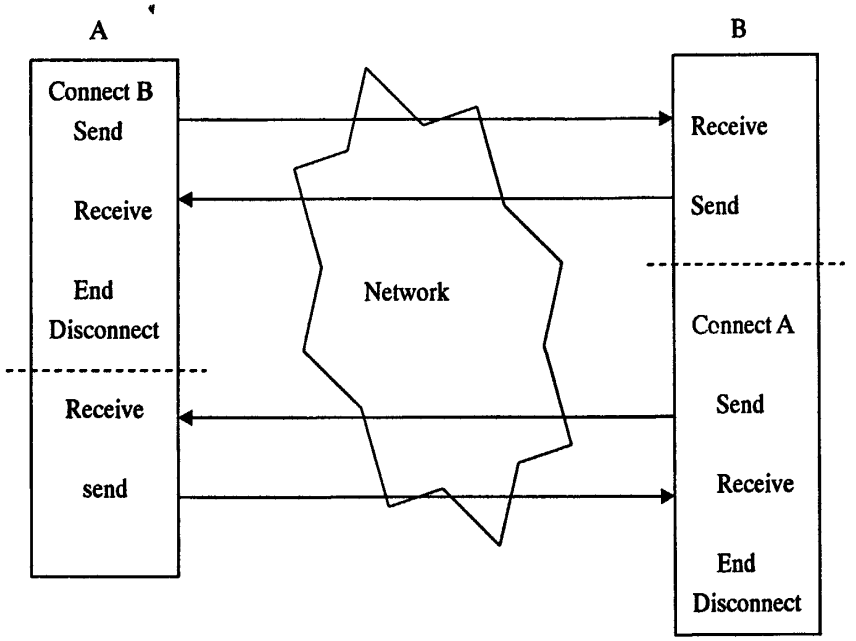

图 2-3 使用 CPI-C 的基于连接的同步通信方式

进行对话通信的程序双方必须跟踪对话的状态，以备发生故障时，对原对话进行恢复。在整个对话过程中，对话双方都必须同时参加对话，不能进行其他操作。如果由于某种原因，连接断开，那么就由对话发起者重建并恢复此次对话。由上图可见，通信双方其实也可处于对等地位，A发起并控制一次对话，那么在此次对话结束后，B同样也可以发起并控制另一次对话。这就表示CPI-C既支持主从模式，也支持平衡模式[1]。

根据以上分析，在设计系统时，需要根据实际的需求来确定使用主从模式还是平衡模式。

2.1.2 面向连接与面向非连接的传输方式

1、TCP

TCP（传输控制协议）提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。TCP提供超时重发，丢弃重复数据，数据检验，流量控制等功能，保证数据正确的从一端传到另一端。TCP的目的是提供可靠的数据传输。TCP在数据包接收无序、丢失或在交付期间被破坏时，负责数据恢复。它通过为其发送的每个数据包提供一个序号来完成此恢复。较低的网络层会将每个数据包视为一个独立的单元，因此，数据包可以沿着完全不同的路径发送，即使它们都是同一消息的组成部分。这种路由与网络层处理分段和重组数据包的方式非常相似，只是级别更高而已。

为确保正确地接收数据，TCP要求在目标计算机成功收到数据时发回一个确认（即ACK）。如果在某个时限内未收到相应的ACK，将重新传送数据包。如果网络拥塞，这种重新传送将导致发送的数据包重复。但是，接收计算机可使用数据包的序号来确定它是否为重复数据包，并在必要时丢弃它。

2、UDP

UDP（用户数据报协议）是一个简单的面向数据报的传输层协议。UDP不提供可靠性，它只是把应用程序传给网络层的数据报发送出去，但是并不保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快。

UDP与TCP的主要区别在于UDP 不一定提供可靠的数据传输。事实上，该协议不能保证数据准确无误地到达目的地。但是，UDP 在许多方面非常有效。当某个程序的目标是尽快地传输尽可能多的信息时，可使用UDP。

3、TCP和UDP的选择

当数据传输的性能必须让位于数据传输的完整性、可控制性和可靠性时，TCP协议是当然的选择。当强调传输性能而不是传输的完整性时，如：音频和多媒体应用，UDP是最好的选择。在数据传输时间很短，以至于此前的连接过程成为整个流量主体的情况下，UDP也是一个好的选择，如：DNS交换。把SNMP建立在UDP上的部分原因是设计者认为当发生网络阻塞时，UDP较低的开销使其有更好的机会去传送管理数据。

2.1.3 服务确认方式

1、无确认方式

无确认方式是指对于一方发出的请求消息，对方都不予回复确认。如下图所示。

图2-4 无确认方式

如上图，A端发出的所有消息，B端接收之后都不进行回复。这种方式节省传输所需的总时间，但是安全性不高。

2、部分确认方式

部分确认方式是指对于一方发出的请求消息，接收方只对其中的一部分进行确认。如下图所示。

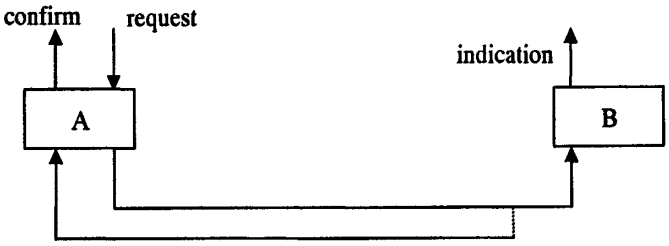

图2-5 部分确认方式

如上图，对于A端发出的所有请求消息，接收端B对于其中较重要的一部分数据进行回复，其他的部分不予回复。这种方式由于要发送回复消息，所以多花费了时间，但是在一定程度上提高了数据传输的安全性。

3、完全确认方式

完全确认方式是指对于一方发出的请求消息，接收方全部予以确认。如下图所示。

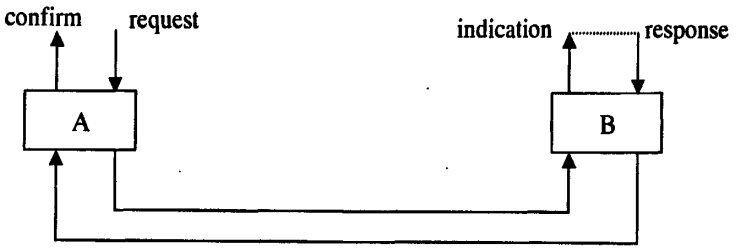

图2-6 完全确认方式

如上图，对于A端发出的所有请求消息，接收端B都予以回复确认。这种方式花费时间最多，但是安全性最好。

比如RPC。RPC（远程过程调用）是一种C/S的编程模式，调用者和被调用者关系是固定的，很难实现对等通信。它跨越了传输层和应用层，本质是实现网络七层协议中的会话层的功能。RPC是一种同步的、对话方式的模型，如下图所示。

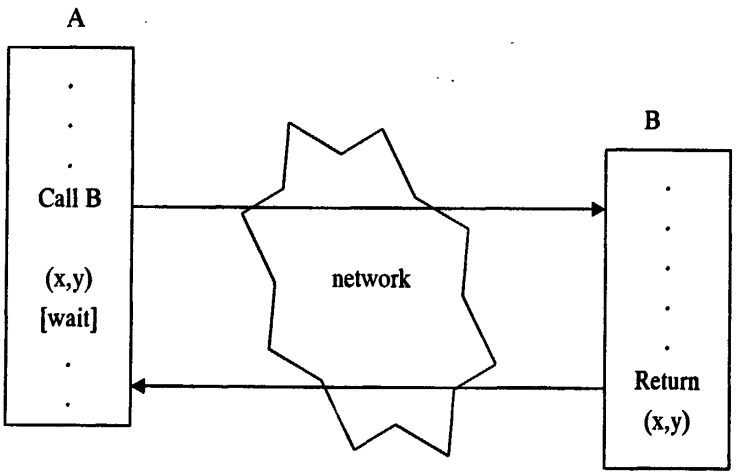

图 2-7 使用 RPC 的基于连接的同步通信方式

对于RPC，请求服务的程序就是客户端，服务提供者就是服务器。首先，客户端的一个调用程序发送一个服务请求给服务器端，然后进入等待状态，等待服务端的回复消息。服务器端接收到服务请求消息，获得请求参数，进行处理，计算结果，然后向客户端发送答复消息，然后进入睡眠状态，直到下一个服务请求消息到达为止。客户端接收到服务器端的回复消息，获取计算结果，然后继续执

行下面的程序[3]。这就是一个完全确认的例子。

2.2 进程间通信方式的比较

管道（pipe）是一种半双工的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用。进程间的亲缘关系通常是指父子进程关系。

有名管道（named pipe）也是半双工的通信方式，但是它允许无亲缘关系的进程间的通信。与进程持续，内容随进程消失而消失。

信号量（semophore）是一个计数器，可以用来控制多个进程对共享资源的访问。它常被作为一种锁机制，防止某进程正在访问共享资源时，其他进程也来访问该资源。因此，信号量主要作为进程间以及同一进程内不同线程之间的同步手段。

信号（signal）是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。

共享内存（shared memory）就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。共享内存是最快的IPC方式，它是针对其他进程间通信方式运行效率低的不足而专门设计的。它往往与其他通信机制（如信号量）配合使用，来实现进程间的同步通信。写入共享内存的东西是一直存在的，主要用于共享大数据文件。

鉴于本项目的需求分析，共享内存不适合用来当做整体通信模型。

消息队列（message queue）是由消息组成的链表，存放在内核中，并由消息队列标识符来标识。消息队列提供有格式字节流，具有类型及优先级别，克服了信号传递信息少、管道及有名管道只能承载无格式字节流等缺点，更简单、安全、灵活，应用空间更大。一旦有进程把消息读出消息队列，此消息就不会继续存在于消息队列中，它其实起到“信箱”的作用。消息队列和共享内存一样，都是随内核持续的，进程消失时，如果不是明确删除该IPC对象，它们并不会消失，只有重启机器才会使它们消失。消息队列常驻内存，可靠且通信效率高。

消息队列为程序提供了一种异步通信方式。一个程序以一个消息队列作为中转与另一个程序进行相互通信，其中，这个消息队列相对于此程序来说，可以是本地的，也可以是远程的。如下图。

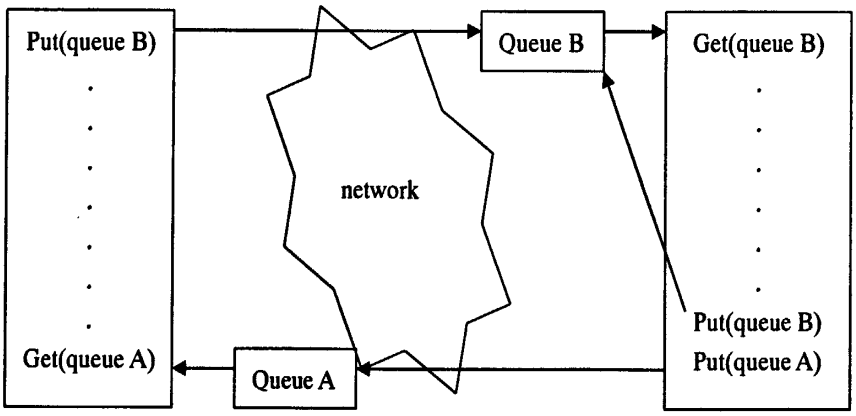

图 2-8 基于消息队列的异步通信方式

比如，程序A和程序B通过消息队列进行通信，它们都有自己的一个消息队列。程序A只需要PUT一条消息到与B相关联的消息队列上，然后程序A就可以干别的事情。在程序A向消息队列PUT消息的时候，程序B可以进行其他的处理，或者也可以未在运行，有时间或者需要的时候再到与之相关联的消息队列上GET一条消息进行处理即可。它们都几乎感觉不到通信的发生。而且，一个程序可以通过向不同的消息队列PUT消息来实现与多个应用程序的通信。

套接字（socket）也是一种进程间通信机制，但是与其他通信机制不同的是，它可以用于不同机器间的进程通信。socket通信采用的是客户/服务器模式，即客户向服务器发出连接请求，服务器接收到请求之后接受，连接建立。

通过比较上面的几种UNIX进程间通信方式，针对本项目的需求，我们选择消息队列通信机制。

2.3 消息队列技术的设计模式

1、异步查询：客户端可以向服务器端查询信息，但是不需要等待确认消息就可以继续往下执行。当确认消息达到时，由专门的处理程序接收并做相应处理。所以，客户端处于完全异步的方式工作。这是标准的查询过程，如果失败，客户端可重复发送请求。这种设计模式简单可靠，为查询应用程序广泛采用。

2、伪同步查询：客户端可以向服务器发送查询请求，并等待查询结果返回，但是客户端并不会永久的去等待。此方式有两个特点，一是客户端是同步工作的，二是如果失败，客户端可以重复发送请求。此模式难点在于如何确定确认消息的超时。

3、异步确认更新：客户端可以向服务器进行更新的操作，在确认消息到达之前可以继续执行下面的操作。此模式应用广泛，但设计难度较大。

4、无确认更新：客户端可以对服务器更新数据，却不需要等待确认消息。

5、伪同步更新：客户端可以向服务器进行更新的操作，并等待更新操作的执行结果。更新操作可以失败，但是客户端必须有处理失败的机制。

通过比较这几种消息队列的设计模式，结合本项目的需求，我们采用无确认更新的模式。

2.4 SOCKET 网络编程

socket通信是网间进程间的通信，即不同主机进程间的相互通信问题（可把同机进程通信看做是其中的实例）。socket通信的不同主机间的进程采用端口进行标识。端口是操作系统可分配的一种资源。类似于文件描述符，每个端口都拥有一个叫端口号的整数型标识符，用于区别不同端口。

socket通信采用的是客户/服务器模式，即客户向服务器发出连接请求，服务器接收到请求之后接受，连接建立。下面具体说明一下通信过程。

首先，创建套接字——socket（）。服务器在使用套接字之前，首先必须拥有一个套接字，系统调用socket（）函数向应用程序提供创建套接字的手段。

指定本地地址——bind（）。当一个套接字用socket（）创建之后，存在一个名字空间，但它没有被命名。bind（）将套接字地址（包括本地主机地址和本地端口地址）与所创建的套接字号联系起来，即将名字赋予套接字，以指定本地半相关。

监听连接——listen（）。它由服务器调用，表示它愿意接收连接，listen（）需要在accept（）之前调用。

建立套接字连接——connect（）和accept（）。这两个系统调用用于完成一个完整相关的建立，其中connect（）用于建立连接。accept（）用于使服务器等待来自某客户进程的实际连接。

数据传输——send（）和recv（）。当一个连接建立之后，就可以传输数据了。send（）用于发送数据，recv（）用于接收数据。

关闭套接字——closesocket（）。此调用用于关闭一个套接字，并释放分配给它的资源，如果此套接字涉及到一个打开的tcp连接，则该连接被释放。

服务器调用过程：socket（）——>bind（）——>listen（）——>accept（）——>recv（）或send（）——>closesocket（）。

客户端调用过程：connect（）——>send（）或recv（）——>closesocket（）。

2.5 UNIX 下多线程编程

和进程相比，使用多线程是一种非常"节俭"的多任务操作方式。在UNIX系统下，启动一个新的进程，必须分配给它独立的地址空间，建立众多的数据表来维护它的代码段、堆栈段和数据段，这是一种"昂贵"的多任务工作方式。而运行于一个进程中的多个线程，它们彼此之间使用相同的地址空间，共享大部分数据，启动一个线程所花费的空间远远小于启动一个进程所花费的空间，而且，线程间彼此切换所需的时间也远远小于进程间切换所需要的时间。据统计，总的说来，一个进程的开销大约是一个线程开销的30倍左右，当然，在具体的系统上，这个数据可能会有较大的区别。

使用多线程的第二个好处是线程间拥有方便的通信机制。对不同进程来说，它们具有独立的数据空间，要进行数据的传递只能通过通信的方式进行，这种方式不仅费时，而且很不方便。线程则不然。由于同一进程下的线程之间共享数据空间，所以一个线程的数据可以直接为其它线程所用，这不仅快捷，而且方便。当然，数据的共享也带来其他一些问题，有的变量不能同时被两个线程所修改，有的子程序中声明为static的数据更有可能给多线程程序带来灾难性的打击，这些正是编写多线程程序时最需要注意的地方。

上面是和进程比较的优势，另外，多线程程序作为一种多任务、并发的工作方式，有以下一些优点：

1）提高应用程序响应。当一个操作耗时很长时，整个系统都会等待这个操作，此时程序不会响应其他的操作，而使用多线程技术，将耗时长的操作置于一个新的线程，可以避免这种尴尬的情况。

2）使多CPU系统更加有效。操作系统会保证当线程数不大于CPU数目时，不同的线程运行于不同的CPU上。

3）改善程序结构。一个既长又复杂的进程可以考虑分为多个线程，成为几个独立或半独立的运行部分，这样的程序会利于理解和修改。

UNIX系统下的多线程遵循POSIX线程接口，称为pthread。编写UNIX下的多线程程序，需要使用头文件pthread.h，连接时需要使用库libpthread.a。

多线程编程是一个很有意思也很有用的技术，使用多线程技术的网络蚂蚁是目前最常用的下载工具之一，使用多线程技术的grep比单线程的grep要快上几倍，类似的例子还有很多。利用多线程技术可以写出高效实用的好程序来。

第三章　需求分析

接收端实时传输子系统与发送端实时传输子系统之间主要采用光纤链路通信，光纤链路故障状态下采用备用的卫星链路通信。链路正常状态下，各发送端按照时间表实时向接收端传输接收到的卫星数据；链路异常状态下，各发送端采用文件传输的形式向接收端传输卫星数据。数据传输过程中，需要传输的还有通信中的请求、应答、命令等信息。

每个实时传输子系统需要与本地的通信设备监控子系统通信。实时传输子系统将传输状态、告警信息等传递给通信设备监控子系统并加以显示。

1、功能性需求

1）传输任务调度功能

由传输任务调度模块负责整个通信行为的监控与调度，用来守护、管理、调度、监控整个子系统。

2）运控重传管理功能

接收端地面站需要接收来自运控分系统的重传命令；根据命令将传输指令分发到其他地面站；实时传输子系统需要根据收到的命令完成数据传输工作。

3）通信传输链路手动选择功能

软件对用户提供光纤通信链路传输方式和VSAT卫星专用信道传输方式选择功能。

用户可通过人机交互界面中的菜单选择或者脚本配置的方式，根据实际需要切换传输信道。

系统能够根据用户的实际选择，对传输文件进行相应的处理。

4）通信传输链路自动配置功能

实时传输子系统在用户未选择手动配置或者脚本配置传输方式的前提下，自动选择一个默认的传输方式。此默认传输方式可以由用户手动配置。如果用户未配置，则默认为光纤传输方式。

5）通信传输链路自动切换功能

实时传输子系统能够根据当前链路的实际状况自动选择切换传输方式。

如果当前光纤链路出现故障或者收发超时，该子系统应当及时切换到卫星链路传输，并且向监视子系统和运控分系统报警。

如果当前光纤链路故障恢复，该子系统应当在传输完成当前数据之后，及时切换回光纤链路进行下一数据的传输，并且向监视子系统和运控分系统报告链路恢复状态。

6）实时遥感数据传输功能

实时传输子系统需要将每个地面站接收到的实时遥感数据在第一时间安全、准确的传输到最终接收端地面站。

子系统需根据实际链路情况设置缓冲池，保证数据的传输效率和准确性。

7）实时数据强占式传输功能

因为每个地面站都有多颗卫星数据的接收，当出现接收数据冲突的时候，应该根据用户的预先配置，高优先级的接收数据可以抢占低优先级的接收数据，并优先传输。

需要向运控分系统报告本系统的抢占情况。

8）实时数据静态备份功能

该子系统在实时传输卫星遥感数据的同时，应将数据备份到磁盘，将数据静态化，以防数据丢失；在发生抢占后，高优先级的数据优先传送，低优先级的数据需要保留到磁盘，待信道空闲后，再次发送到地面站。

9）实时传输和静态数据保存自动切换功能

能够根据链路的实际状态，实时接收并同步转发遥感数据。

根据链路类型和状态的不同，系统将做出如下工作：

（1）如果当前采用光纤链路，且链路状态畅通，则系统将实时接收并同步转发遥感数据；

（2）如果当前采用光纤链路，但链路状态出现问题，无法连通，则将数据存储为文件，并且换到卫星链路传输，转发文件；

（3）如果当前采用卫星链路，且链路状态畅通，则系统实时接收数据，并保存为文件，转发文件；

（4）如果当前采用卫星链路，但链路状态出现问题，无法连通，则将数据存储为文件。

10）数据传输状态报告功能

实时传输子系统能够向运控分系统报告当前数据的传输状态，并在出现异常时提交报警信息，便于监控人员进行处理。

11）数据传输状态监视功能

通过人机交互界面可以查看实时传输子系统当前数据的传输状态和任务调度等。

12）断点续传功能

实时传输子系统能够在上次数据传输失败的地方继续传输剩余的原始数据，以减少重传的工作量，提高传输效率。

13）工作日志功能

能够及时准确的记录数据传输任务及软件工作状态。

14）数据压缩和解压缩功能

发送方根据系统配置要求采用相应的压缩算法对原始数据进行实时无损压缩，以降低对传输带宽的要求；

接收方能对接收到的压缩码流进行实时解压缩，恢复原有码流。

15）多码流调度与流控功能

根据优先级分配数据传输带宽。

16）软件配置

能够方便快捷地配置软件运行环境和参数，运用交互式、图形化的方式帮助管理员配置各种信息。

能够对不同数据来源的数据码流进行调度与流量控制。

2、质量特性

1）系统能够高稳定、高可靠地运行，系统可靠性达到99.9%。系统能自动检测并重置僵死进程，保证系统的稳定和健壮。

2）当采用卫星信道传输方式时，为了充分利用卫星信道带宽，要求两个发送端地面站至接收端的文件传输软件能够适当提高等效传输带宽，信道利用率达到85% 以上。

3）当采用光纤信道传输方式时，要求达到85%以上的信道利用率。

4）实时传输时，提供QOS服务，保证数据不因链路不稳定而丢失数据或出错。

5）人机交互界面响应时间不超过3秒。

3、可靠性要求

通信分系统采用可靠、成熟的技术开发，以保证软件系统具有一定容错性、易恢复性。

软件故障：在系统出现故障后立即退出重启。

硬件故障：保存当前状态信息。

4、易用性要求

通信分系统有监控和人机交互画面，软件设计中尽量采用面向对象技术，以直观、美观和方便易学的界面提供给操作员和管理员。每个画面要求有联机提示和联机帮助。

5、维护性要求

通信分系统按军标软件工程设计，要求最终软件系统始终与文档保持一致。文档包括中文文档和硬拷贝两类，要求同时更新。

第四章　实时传输子系统的设计与实现

4.1 实时传输子系统模型设计

4.1.1 采用主从模式

按照XXXX卫星实时传输子系统的需求，只允许接收端向发送端下达命令消息，反向的命令发送是不可以的，发送端只可以向接收端发送卫星数据消息。在这种情况下，只能采用主从模式，才符合要求。如下图所示。

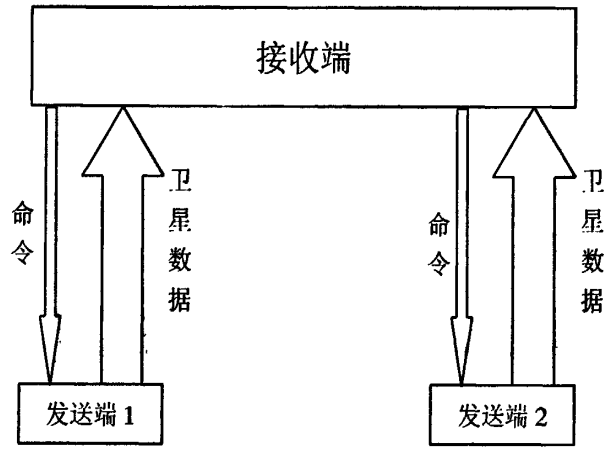

图4-1 实时传输子系统主从模式模型

4.1.2 采用面向连接的传输方式

通过对实时传输子系统的需求进行分析，我们可以知道系统对可靠性的要求很高：1）系统能够高稳定、高可靠地运行，系统可靠性达到99.9%；2）通信分系统采用可靠、成熟的技术开发，以保证软件系统具有一定容错性、易恢复性。

通过前面对面向连接的传输方式和无连接的传输方式的分析比较，可知，TCP提供超时重发、丢弃重复数据、数据检验、流量控制等功能，能够保证数据正确的从一端传到另一端。TCP的目的是提供可靠的数据传输。所以，本系统采用面向连接的传输方式，即在建立连接时，采用基于TCP协议的socket通信。

4.1.3 系统采用的服务确认方式

本系统采用无确认方式、部分确认方式和完全确认方式相结合的方式。本系统向运控报告传输状态时，就不需要运控给予回复，采用无确认方式；发送端向接收端发送一些控制消息时，需要接收端进行回复，但是发送卫星数据时，就不需要接收端进行回复，采用部分确认方式；接收端向发送端发送传输命令时，发送端在接收到任何一个命令并进行相应处理后，都要进行回复确认，采用完全确认方式。

根据以上的分析，下面给出实时传输子系统的模型。

4.1.4 实时传输子系统模型

根据系统需求及以上分析，我们提出了实时传输子系统模型。下图为发送端实时传输子系统向接收端实时传输子系统进行数据传输的整体模型工作流程图。

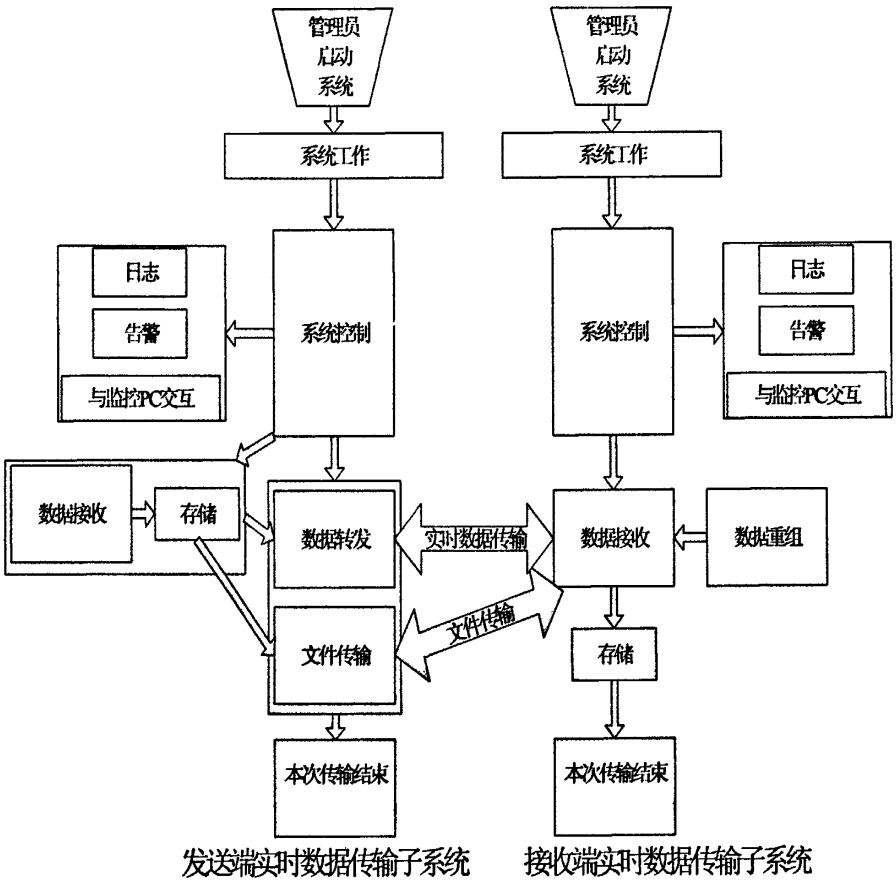

图4-2 实时传输子系统模型工作流程图

1）首先，由通信服务器管理员启动实时传输子系统，实时传输子系统在操作系统中启动。

2）实时传输子系统监视是否有传输请求。当需要传输时，启动系统控制，接收端的系统控制在接收数据时启动数据接收与数据重组、存储，发送端的系统控制在转发数据时启动数据接收用以接收来自本地数据记录PC的数据，启动存储进行接收数据的本地存储，启动数据转发实时转发数据。

3）数据传输过程中，由系统控制监视全局运行状况，接收处理各个功能模块的信息并反馈，同时修改日志，并在故障状态下向运控、监视系统告警，及时将传输状态通过与监控PC的交互发送给本地的监控系统用以显示。

4）在链路故障状态下，在备用卫星链路下，启动文件传输进行文件传输。

5）传输结束或突然出现故障时结束传输，返回待命工作状态等待新的指令。

下图为接收端实时传输子系统和发送端实时传输子系统进行数据传输的具体工作流程图。

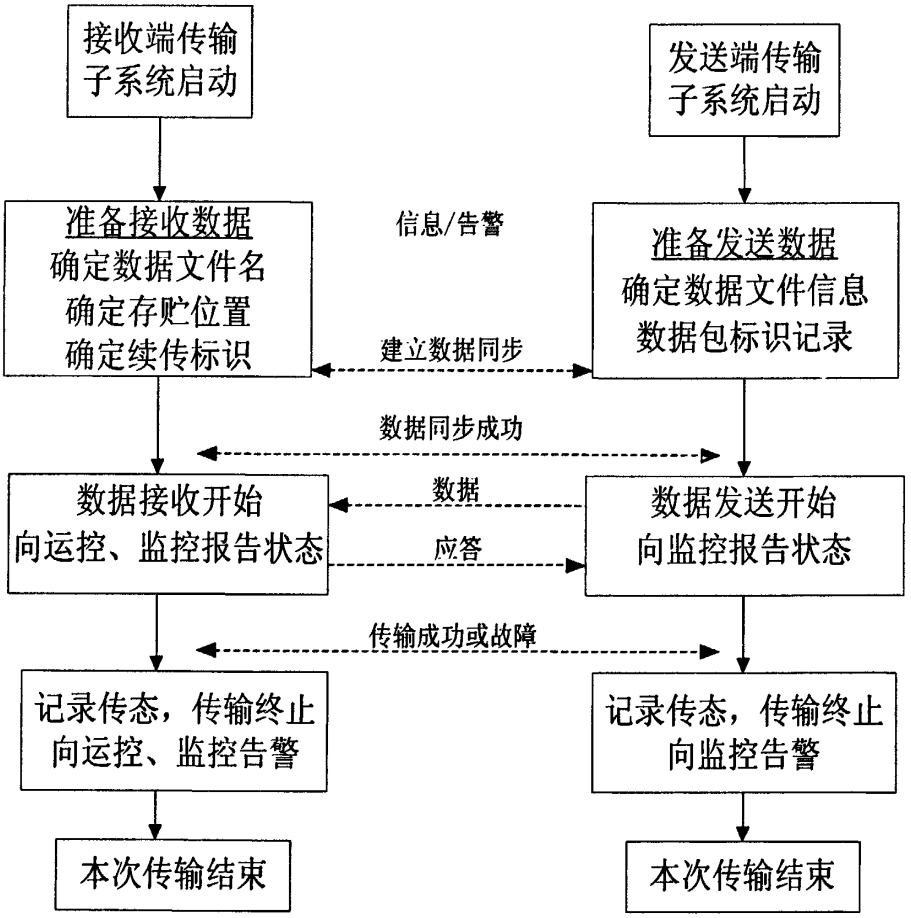

图 4-3 实时传输子系统工作时序图

1）发送端和接收端启动后，收发双方分别确定通信所必须的参数。发送端：

实时数据文件路径、目的主机地址、文件是否压缩、通信连接的有关参数、发送过的数据包号记录（在断点续传时使用）等；接收端：目标数据文件路径、源主机地址等。

2）系统控制向本地的监控系统报告传输状态，如传输数据字节、是否正常等。

3）发送方对原始数据文件进行传输预处理，在原始数据文件压缩后添加头文件（内容包括压缩后的数据文件大小、名称、目录、数据包号等信息），以便于监视数据传输的具体情况，必要时向运控分系统和监控子系统报告异常：原始数据文件不存在、原始数据文件压缩失败等。

4）收发双方自行建立数据同步。

5）发送方开始发送数据，接收方向发送方传送应答信息。

6）接收方定时向运控分系统与监控子系统报告传输过程中的状态参数（如数据传输完成的百分比），并在出现异常时及时向运控、监控发送告警信息（如网络中断、原始数据文件在传输过程中有丢失、目标文件存取失败等）。

7）传输成功或终止时，接收方对接收文件进行传输后处理，并根据原始数据文件的结束标志所携带的信息来分析数据的传输是否结束，并形成相应的状态报告，向运控分系统报告传输状态：原始数据是否已成功发送完毕等。

8）收、发程序自行终止运行。

4.1.5 发送端和接收端的交互协议

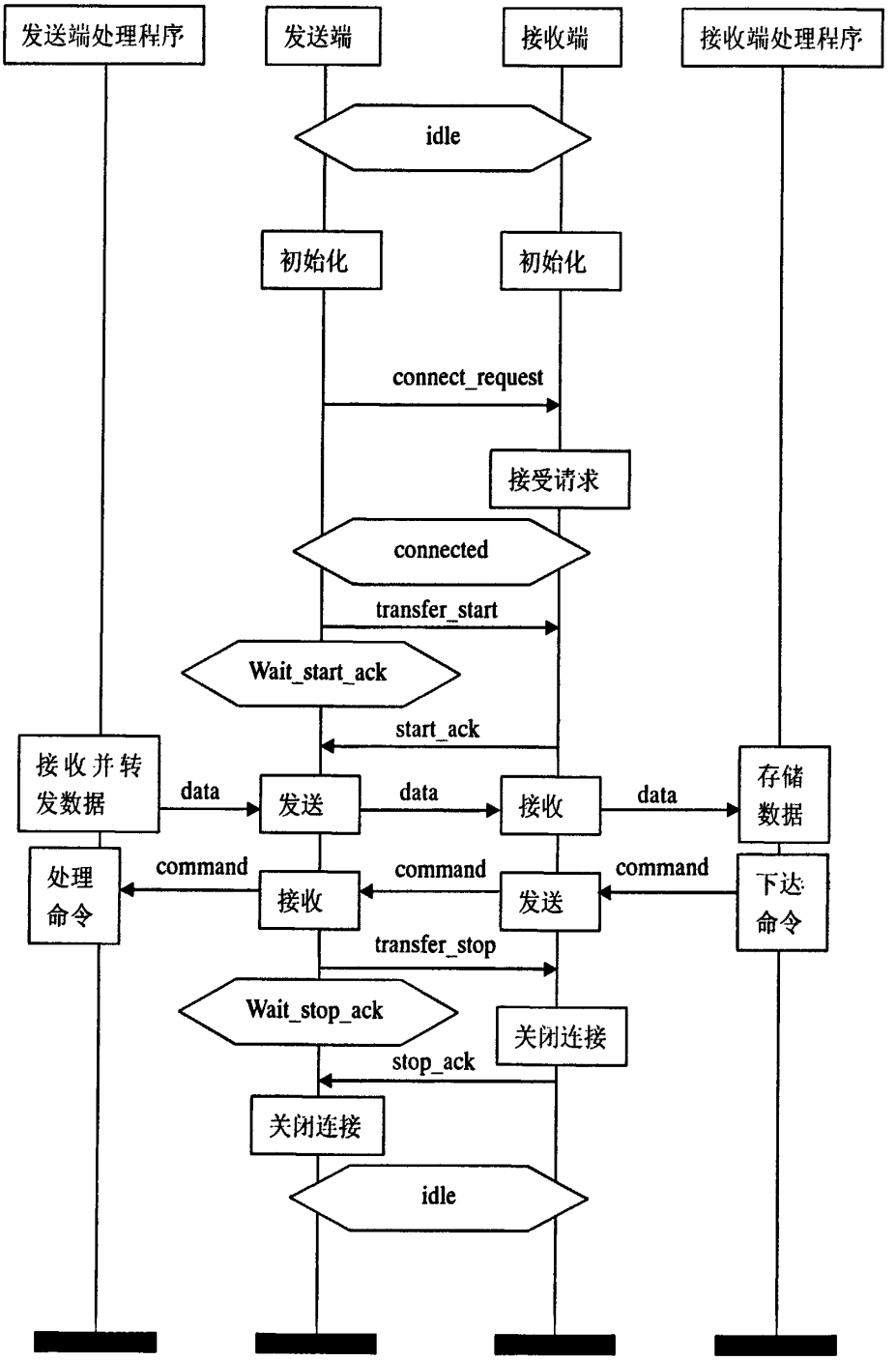

图4-4 发送端和接收端的交互协议

接收端初始化，等待发送端来连接。发送端初始化之后，发送连接请求，接收端接受连接，此时连接建立。发送端发送transfer_start消息，接收端回复start_ack消息，此时握手成功，可以开始传输数据。发送端和接收端的消息发送是全双工的。发送端向接收端发送大量的卫星数据，反方向只是少量的命令信息。传输完

最后一包后，发送端发送transfer_stop消息，接收端关闭本次连接后，回复stop_ack消息，发送端也关闭对应的连接，此时，传输连接完全关闭。如果还需要建立连接，那么接收端会继续监听，否则，接收端也关闭初始化的套接字，结束全部服务。

transfer_start、start_ack、transfer_stop和stop_ack四个都是连接控制消息，都包含以下五个字段：控制类型，线程数，包号，文件名，当地路径，通过控制类型字段来区别。

发送端处理程序和接收端处理程序将会在后面的系统设计部分分模块详细介绍。

4.1.6 消息传输系统

- 系统内的消息流动

本系统是一个基于消息队列和socket连接的安全、可靠的通信系统。系统内的消息流动控制和管理着整个系统，使得这个系统的多台机器之间通过传递消息完成整个工作流程。本系统基本上由一个消息传输系统和一组应用程序接口组成，消息和队列是其重要组件。原理如下图。

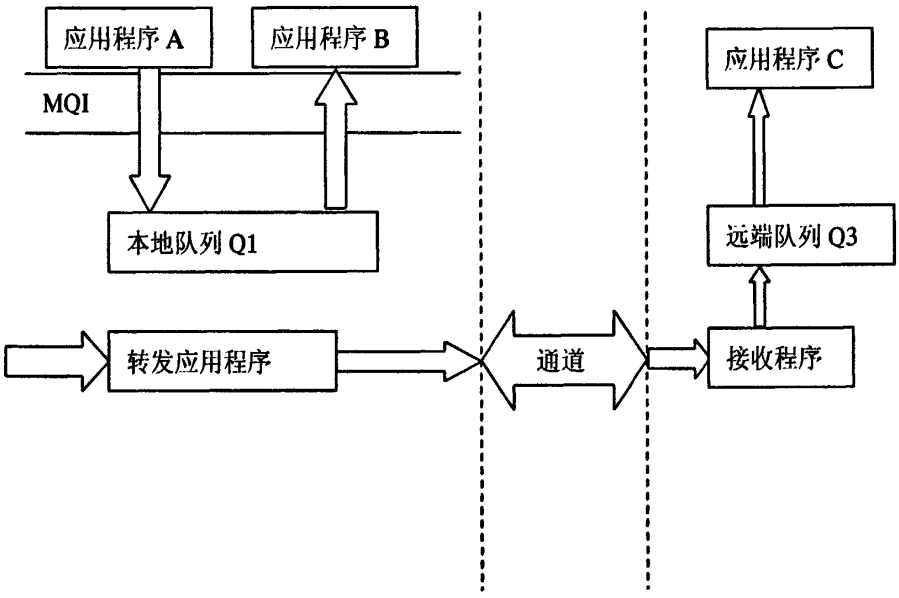

图4-5 系统内的消息流动

如上图所示，应用程序A和应用程序B运行于同一系统之上，它们不需要直接的通讯，而是通过消息队列。应用程序A调用发送接口向本地队列Q1发送一条

消息，而当应用程序B需要时，就可以调用读取接口获取该消息。

如果消息传输的最终目标是在远端系统上的应用程序C，此时，转发应用程序就会将消息通过socket通道发送到对端。远端系统的接收程序调用读取通道的接口成功接收消息之后，将消息发送到对应的远端队列Q3，当应用程序C在需要的时候，即可以从队列Q3里读取消息[5][20]。

- 多层队列机制

因为系统控制模块中最高层的消息队列接收各种消息，所以当消息数量太大时，可能会造成阻塞现象。为解决这个问题，我们采用多层队列机制进行消息的分发。如下图。

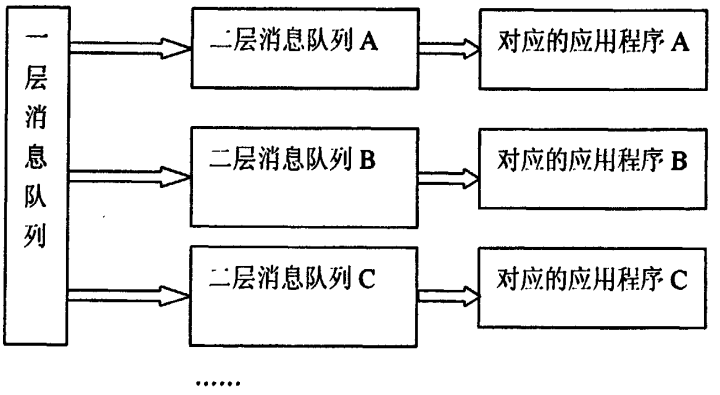

图4-6 多层队列机制

从上图可见，一层消息队列根据消息类型等进行分类，如果可以直接处理，则直接处理，若需要进一步处理且此类消息需要排队处理的，则根据消息标志发送到下一层对应的消息队列，再由下一层的对应处理程序读取消息并进行处理。这样的设计减轻了上一层消息队列的压力，有效避免了队列堵塞，分类处理比较简洁，分工明确，提高了处理速度[4][18]。

- 队列创建

在本模型中，队列是由队列控制器进程来创建的，使用以下接口：int initqueue(key_t key)。

- 队列删除

在本模型中，队列是由队列控制器进程来删除的，使用以下接口：int remmsgqueue(int queueid)。

- 向队列放消息

在本模型中，任何有访问权限的应用程序都可以向某个队列里放消息，使用

以下接口：int msgwriteStatus(int queueid, char *buf)。

- 从队列取消息

在本模型中，一般情况下，由队列控制器进程来读取系统管理消息队列里的消息，使用以下接口：int msgread_log(int msg_log_id, void *buf)[21]。

某些有特殊用途的队列，单独设置了一个接口：int msgread(int queueid, void *buf)。

至于从队列里取消息的顺序，我们暂时只采用了先进先出（FIFO）。

- 与队列控制器有关的进程/线程

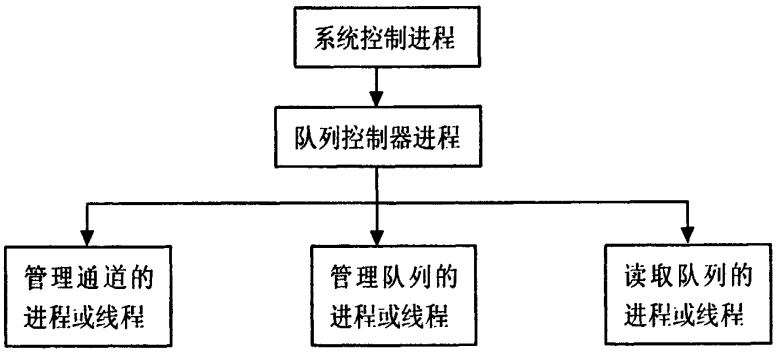

图 4-7 与队列控制器有关的进程/线程

队列控制器进程是由系统控制进程创建的，它启动后，将会启动一系列的进程或线程。如上图所示，下面简单介绍一下它们的功能。

1）管理通道的进程或线程，主要来创建通道，删除通道。

2）管理队列的进程或线程，主要来创建和删除队列。

3）读队列的进程或线程，主要来读取某个队列里的消息，因为系统可能需要多种用途的队列，所以读队列的进程或线程就需要多个[7][13]。

- 触发的概念

我们把某种条件叫做触发事件，如果触发事件发生了并且触发消息被发送到启动队列，那么触发启动程序（即阅读此队列的消息的应用程序）就会读取此队列的消息并处理，然后启动相应的应用程序。

- 触发所涉及的对象

1）触发事件：它是一种引起应用程序产生并发送触发消息的事件。

2）触发消息：当触发事件发生时，应用程序产生触发消息，并发送到启动队列。触发信息包含了将要被启动的应用程序的信息。

3）启动队列：这是一个本地的触发队列，它用来存储触发消息。一个平台可以拥有多个启动队列。一个启动队列也可以为多个应用程序服务。也就是说，

一个启动队列里可以存放启动多种应用程序的消息。

4）触发启动程序：这是一个持续运行的程序，它一直在堵塞监视启动队列，一旦有触发消息到达启动队列，它就立马读出此触发消息，并分析触发消息中的信息，启动相应的应用程序。然后等待并读出下一条消息。

触发的工作原理图如下。

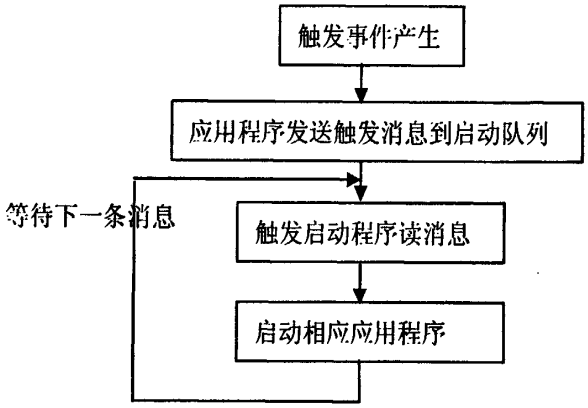

图 4-8 触发的工作原理图[8]

- 本机队列间消息通信

工作原理如下。

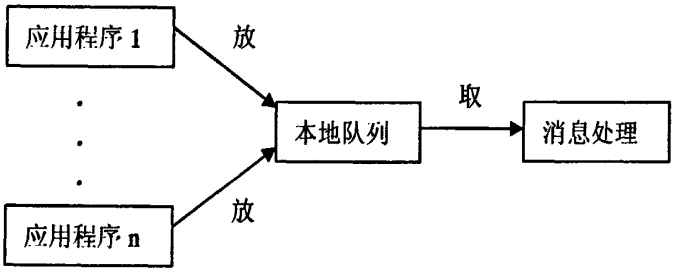

图 4-9 本机队列间消息通信

在一台机器上，可以有多个应用程序都将某些消息放入同一个消息队列，然后读消息队列的应用程序读取一条消息，进行相应处理。

- 异机队列间消息通信

不同平台之间的消息通信是通过消息通道（本模型采用socket通道）来实现的。工作原理如下。

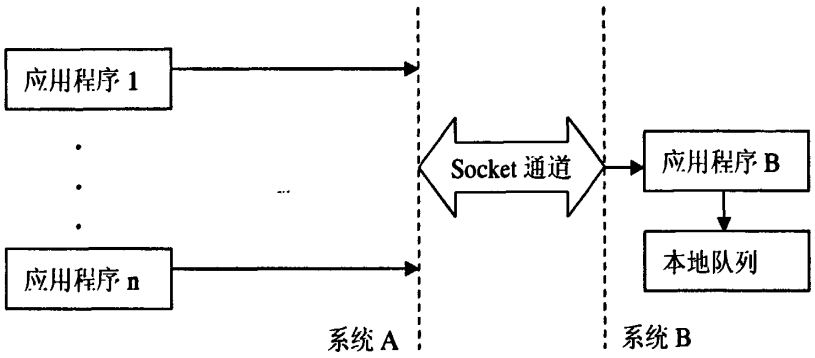

图 4-10 异机队列间消息通信

上图中，系统A中的多个应用程序将要发送给远端的消息通过通道发送到远端系统B，由系统B的应用程序B接收，并放入相应的本地队列。

在系统 B 上的本地队列其实就是目标队列，在远端的接收程序接收到消息之后便会将消息发送给它本地的目标队列。它用于最终接收源机器发送过来的消息。

- 通道操作接口

通道创建：

int u_open(u_port_t port, const char *cp);

int u_accept(int fd);

int u_connect(u_port_t port, const char *cp);

通道删除：

int r_close(int fildes);

写通道：

ssize_t r_write(int fd, void *buf, size_t size);

读通道：

ssize_t r_read(int fd, void *buf, size_t size) [2];

另外，我们还封装好了通道两端的应用程序在发送数据之前进行交互握手确认的接口。

一端：

发送控制消息：void sendCTRLMessage (int sockfd, CTRLTYPE type, int count, int num);

接收确认消息：void waitAck (int sockfd, CTRLTYPE type);

另一端：

接收控制消息：int waitCTRLMessage(int sockfd, ControlMessage *ctlMsgp,

CTRLTYPE type);

发送确认消息：void sendAckMessage (int sockfd, CTRLTYPE type, int count, int packageId,char *filename)。

- 共享通道

如果有不同类型（消息格式、消息长度等不一样）的消息要传送，那么可以使用相同的通道，也可以使用不同的通道。共享通道的情况如下图所示，描述了多种类型的消息通过同一个通道传送，最终到达远端系统不同的目标队列。共享通道可以减少系统资源开销。但是处理消息耗费时间，如果处理不当，还会提升不安全性。

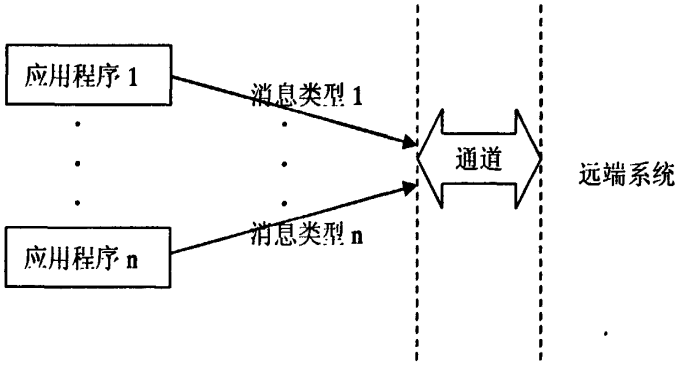

图 4-11 共享通道

- 使用不同的通道

如下图所示，不同类型的消息采用单独的通道，最终也可以到达自己的目标队列。使用不同的通道，处理消息的时间少，可以提高安全性，但是却增大了系统的资源开销。

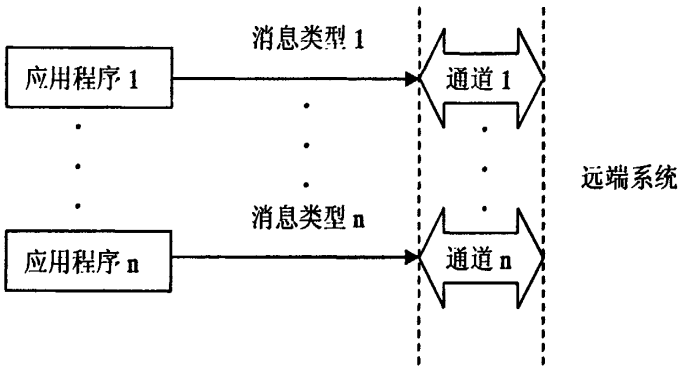

图 4-12 使用不同的通道

4.2 实时传输子系统中的基本概念

● 消息

消息就是一个信息单元，这个信息单元可以是一个请求，可以是一个应答，也可以是一个状态报告或一份报文，也可以是命令信息。一般情况下，一个消息包含两个要素——描述信息（用于存储控制信息、包号、消息目的地、优先级等）和消息内容（存储真正要执行的命令内容、状态内容等）。各应用程序之间的通信不是直接调用，而是通过传递消息来进行。消息是对使用它的应用程序来说有意义的以字节为单位的字符串。我们可以使用消息来实现相同或不同平台上的应用程序之间的通信。

在本系统中，从内容上来说，消息大体由以下两个部分组成：

消息控制信息：包含各种控制信息，如消息来源模块号、消息类型、消息号、消息的优先级等。

消息内容：这是执行操作真正需要的信息。如传输状态信息、命令内容、要传输的文件名字、文件类型、卫星代号等。

消息大体结构如下图所示。

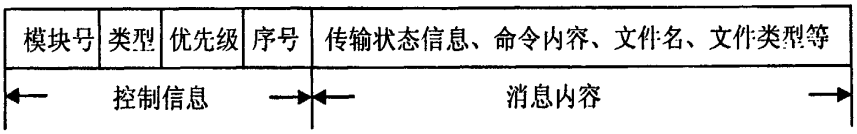

图 4-13 消息结构

以上结构中，具体的内容和结构由调用它的应用程序来定义和组织。

● 消息的类型

本系统主要定义以下几种类型的消息：

1）请求消息

请求消息由发送端发往接收端，并且需要接收端的应答。请求消息中应该包含消息来源，以保证回复消息能够发回正确的目的地。某些情况下，发送端和接收端的两台机器可以互换。如本系统中的接收端和发送端在建立某些连接时，地位可以颠倒。

2）命令消息

命令消息一般由运控服务器发往接收端子系统，要求实时传输子系统完成某项任务，比如重传或者传输某一个或某一些文件。命令消息同样需要接收端子系

统在任务完成后对其进行回复。

3）回复消息

回复消息是对请求消息和命令消息的回应。请求消息中的信息决定了回复消息的目的地。消息的处理和回复仅由直接通信的两端应用程序控制，和队列控制器没有关系。消息自身能够携带足够的信息，以实现应用程序之间的这种关联。

4）状态消息

状态消息是不需要对方回复的消息，它只是一次单向的信息传送，状态上报完成即可。

5）错误报告消息

错误报告消息用于对一些系统异常作出响应。任何进程都可以上报错误报告消息。错误报告消息包括传输失败，发送消息到队列失败，从队列读消息失败等。

另外，有一些消息有触发的作用，当它被读出的时候，会马上触发一些动作的发生。我们还可以称之为触发消息。触发消息是触发机制的重要组成部分。

- 消息长度

消息长度因消息类型的不同而不同。相同类型的消息我们用一个结构体或组合来定义，所以某种类型消息的长度取决于定义其的数据结构的长度。具体长度可以用sizeof（数据结构的名字）来获取。

- 队列

队列是用于存储消息的数据结构。在队列里，消息的存储一般是顺序的。队列使得消息能够被分阶段地发送和接收。消息存放在队列中，各应用程序可以相互独立地运行，时间、地点和速度都可以不统一，也就是说，各程序可以异步运行，若需要向队列发送消息或者需要从队列读取消息，可以随时去发送和读取，实现了异步通信。

- 本地队列

本地队列是物理上位于本地队列控制器中的队列。本地队列实际上存在于本地系统的内存或磁盘存储中。本地的队列控制器可以控制队列，比如创建和删除它们。本地的应用程序可以调用发送接口发送消息到本地队列，也可以调用接收接口从本地队列取出消息。

- 远端队列

远端队列所属的队列控制器不与该应用程序在同一机器上。若要在本地队列和远端队列间通信，则需要用到通道。

本地的应用程序不能直接对远端队列进行操作。本地的应用程序只能向本地队列发送消息或从本地队列读取消息。

- 消息传输系统

消息传输系统用于确保各队列之间的消息传输，包括本地系统和网络中不同系统上的远端队列之间的消息提供，并保证系统故障或网络故障后的错误恢复。

- 队列控制器

在本系统中，队列控制器是基本的软件组件，它对队列进行管理和维护。本系统中的每一个队列都被一个队列控制器所管理，由队列控制器创建和删除。

一个队列控制器是整个系统中的一个基本的独立的执行单元，可以是一个进程，也可以是一个线程。一台机器上可以运行一个或多个队列控制器。

本地应用程序可以操作、管理本地队列控制器所拥有的各种资源，比如发消息和取消息，调用通道向远程队列发送消息等。

应用程序通过MQ接口调用服务。这些服务包括“放”一条消息到队列或从队列“取”一条消息等一些基本操作。另外，队列控制器自身也可以将消息放入队列或从队列取消息。

- 通道

本系统中的通道其实就是一条socket连接。它是一种通信路径，它用在分布式系统中，用于连接交互的两台机器。为了实现此通信，一端的应用程序必须调用创建通道的接口，来创建一个通道，对方接受创建通道的请求，此时通道（即socket连接）建立。消息通道是一个全双工链接。本地的应用程序可以通过消息通道把消息发送到一个对方，也可以从同一个通道来接收对方的应用程序发送来的消息。

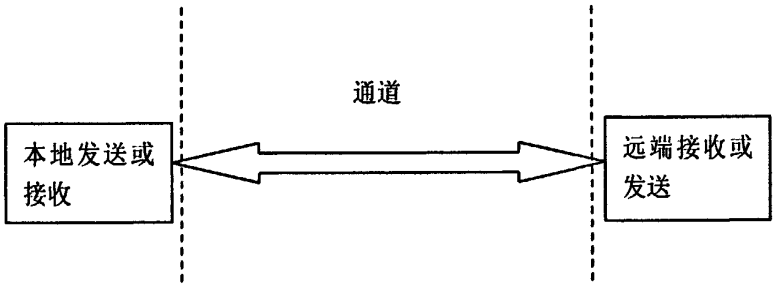

图 4-14 通道的使用

- MQI

MQI（消息队列接口）由以下部分组成：

1. 函数接口：应用程序通过函数接口才可以访问队列和管道等，如从本地队列取消息需要调用“取”函数接口，向本地队列放消息需要调用“放”函数接口，创建管道需要调用“创建管道”的函数接口等。

2. 数据结构：仅仅函数接口并不能实现全部操作，还要有函数接口所需要的数据结构，比如，涉及到日志消息的操作就要用到已经定义好的日志消息数据

结构。

3. 基本数据类型：上面的数据结构是本系统的具体实现定义的，而基本数据类型则是系统本身定义的，我们在编程的时候肯定会用的到。

4.3 实时传输子系统的概要及详细设计

4.3.1 系统总体架构

实时传输子系统分为接收端实时传输子系统和发送端实时传输子系统。接收端实时传输子系统主要通过光纤链路和发送端实时传输子系统进行通信，在光纤链路故障状态下采用备用的卫星链路进行通信。

实时传输子系统的整体结构如下图：

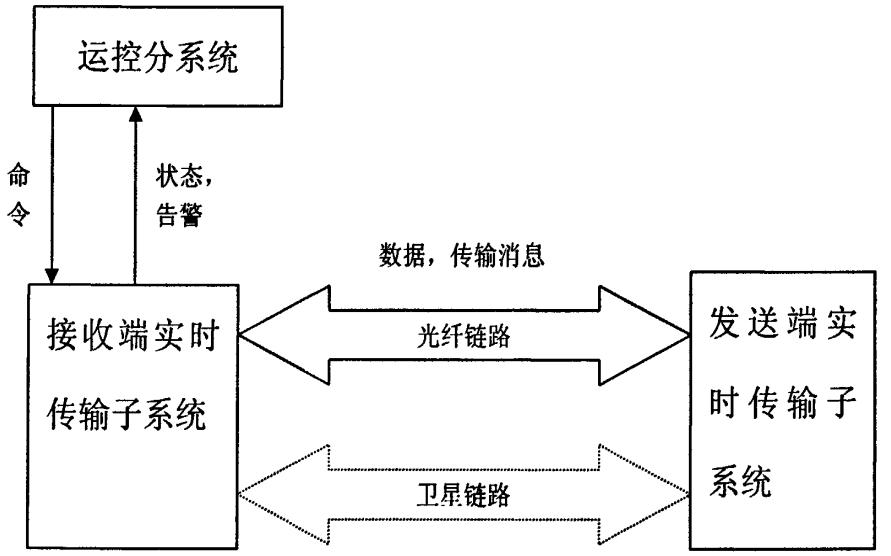

图 4-15 实时传输子系统的整体结构

- 接收端实时传输子系统

接收端实时传输子系统一方面负责接收并存储接收端地面站的卫星数据，另一方面负责接收发送端实时传输子系统的实时数据（如果光纤链路断开，则是用卫星链路传输的数据文件），并进行重组和本地存盘。

接收端实时传输子系统按照任务时间表提前一段时间启动，同时，在接收数据的过程中，向运控分系统报告传输开始，传输状态，传输结束，传输异常等信息，在设备监控子系统中输出当时的传输状态等信息。

从该子系统的设计需求可概括出本传输的设计目标如下：

1、准确地接收并存储接收端地面站的卫星数据；准确地接收发送端通信服务器传输的实时数据，实现数据重组并正确存盘；

2、能够与运控分系统交互，即时上报传输状态和异常状态等；能准确接收运控的传输重传命令，并准确执行相应的传输操作；

3、能够与设备监控子系统交互，即时上报传输状态，并能反馈设备监控子系统对参数的更改。

为实现上述目标，在本传输设计中，按各个模块对本传输进行功能的划分。

接收端实时传输子系统的框架如下图：

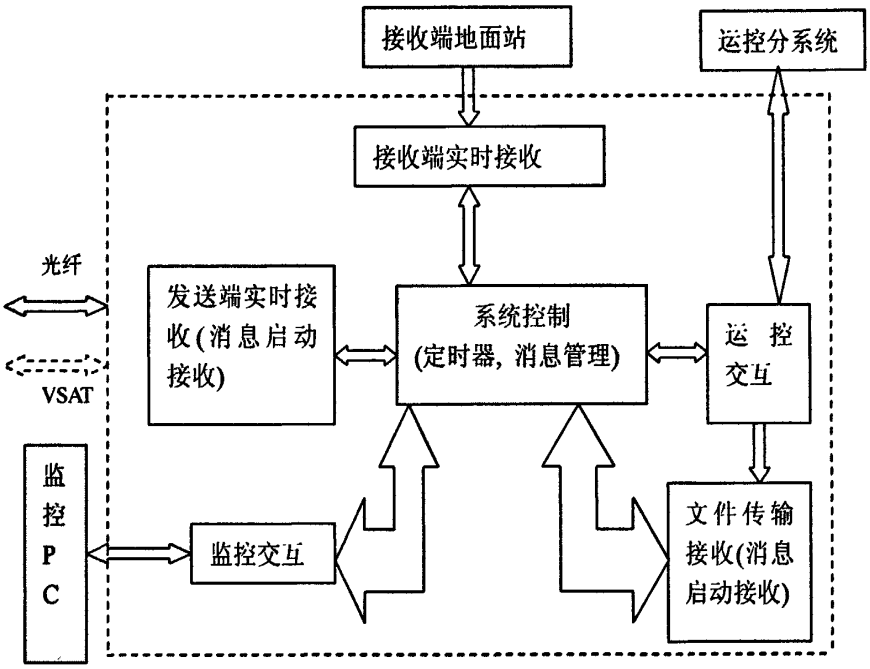

图 4-16 接收端的框架

各个模块说明如下：

1、系统控制

系统控制模块监控整个系统的运行状态，负责系统的启动，系统的重新启动以及安全退出。开始，初始化消息队列，任务信息，载入配置文件，连接运控，创建队列控制器进程，创建其他几个进程或线程；队列控制器进程创建相应的实时传输消息队列和文件传输消息队列；根据时间表提前一定时间将启动消息发送到相应的消息队列；定时器定时将心跳消息发送给运控和监控，并定时检查并管理任务列表；从消息队列里读取消息，并对其分类处理，其中包括写日志，任务处理，向运控发送心跳及状态等。

2、接收端实时接收

读消息队列，若有消息，则启动接收进程，从接收端地面站接收实时数据，先存到本地磁盘，然后传到盘阵。接收完成，退出，继续读消息队列。

3、发送端实时接收

读实时传输消息队列，若有消息，则启动接收进程，从发送端接收实时数据，先存到本地磁盘，然后传到盘阵。接收完成，退出，继续读消息队列。

4、文件传输接收

一方面，监听发送端文件传输连接，连接成功之后等待开始消息，然后将文件启动的消息发送到文件传输消息队列。

另一方面，读文件传输消息队列，若有消息，则启动接收文件进程，从发送端接收文件，先存到本地磁盘，然后传到盘阵。接收完成，退出，继续读消息队列。

5、运控交互

跟运控交互，将消息队列里面的传输状态消息以及告警传给运控，并且接收运控传来的传输重传命令，传给系统控制模块。

6、监控交互

跟监控 PC 交互，包括两个部分：定期将消息队列里面的传输状态消息以及告警传给监控 PC，由监控 PC 将其显示在人机交互界面上，方便管理员观察传输状态；接收来自监控 PC 更改过的参数（管理员更改过的参数），发送给传输控制模块，由系统控制模块及时进行修改。

- 发送端实时传输子系统

发送端实时传输子系统一方面负责接收并存储本地地面站传输的卫星数据，另一方面负责将接收到的卫星数据转发给接收端通信服务器，在正常情况下采用光纤链路传输，在光纤链路断开情况下采用卫星链路传输。

发送端实时传输子系统接收接收端实时传输子系统的命令启动，同时，在传输数据的过程中，向设备监控子系统上报消息，在设备监控子系统中输出显示，并接收设备监控子系统的参数更改命令，进行相应的更改操作。

从该子系统的设计需求可概括出本传输的设计目标如下：

1、准确地实时接收并存储发送端地面站的卫星数据；

2、高效、可靠、实时地传输接收到的数据给接收端通信服务器；

3、按照卫星数据的优先级，动态分配并调节传输的带宽；

4、在光纤链路断开的情况下，采用卫星链路进行高效的文件传输；

5、准确判断网络拥塞状况，并能进行断点续传。

6、能够与设备监控子系统交互，即时上报传输状态，并能反馈设备监控子

系统对参数的更改。

为实现上述目标，在本传输设计中，按各个模块对该传输进行功能的划分。

发送端实时传输子系统的框架如下图：

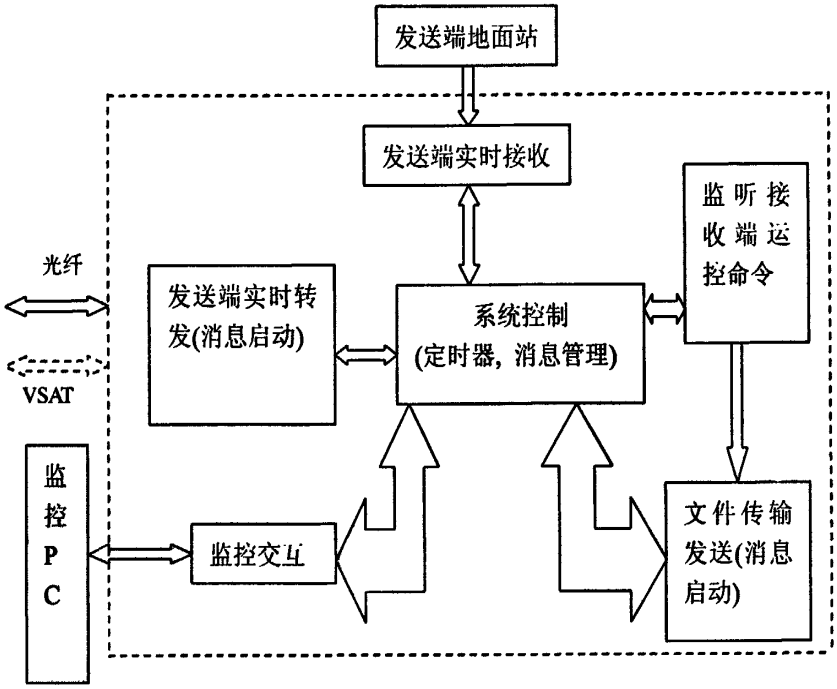

图 4-17 发送端的框架

各个模块说明如下：

1、系统控制模块

系统控制模块监控整个传输系统的运行状态，负责系统的启动，系统的重新启动以及安全退出。开始，初始化消息队列，任务信息，载入配置文件，创建队列控制器进程；队列控制器创建相应的消息队列；创建其他几个进程或线程；接收接收端的启动任务消息，将启动消息发送到相应的消息队列；定时器定时将心跳消息发送给监控，并定时检查并管理任务列表；从消息队列里读取消息，并对其分类处理，其中包括写日志，任务处理，向监控发送心跳及状态等。

2、发送端实时接收

监听发送端地面站，若有连接且有数据，则判断数据是否是即将要接收的数据，若是，则接收，否则，拒绝接收。接收发送端地面站的数据，存到本地磁盘。接收完成，继续监听。

3、发送端实时转发

读实时转发启动消息队列，若读到消息，则连接接收端的实时数据监听端口，

建立连接，然后建立多条 TCP 连接，转发数据，并向控制模块上报转发状态，异常等。转发完成，以上连接全部断开，继续读实时转发启动消息队列。

4、监听接收端运控命令

监听接收端运控模块连接，接收端实时传输子系统的运控模块一旦来连接，则等待接收端实时传输子系统的运控模块发来的开始消息，接收到开始消息之后，对参数进行解析，然后将文件开始消息发送到文件传输启动消息队列，然后向接收端实时传输子系统的运控模块发送开始回复消息。关闭连接，继续监听来自接收端运控模块的连接。

5、文件传输发送

一方面，接收控制模块的命令，连接接收端的文件传输监听端口，连接成功，则将文件发送启动的消息发送到文件传输消息队列。然后给接收端发送开始消息。关闭连接。继续等待控制模块的命令。

另一方面，读文件传输消息队列，若有消息，则启动发送文件进程，向接收端发送文件，发送完成，退出，继续读消息队列。

6、监控交互

跟监控 PC 交互，包括两个部分：定期将消息队列里面的传输状态消息以及告警传给监控 PC，由监控 PC 将其显示在人机交互界面上，方便管理员观察传输状态；接收来自监控 PC 更改过的参数（管理员更改过的参数），发送给传输控制模块，由系统控制模块及时进行修改。

4.3.2 接收端处理程序概要及详细设计

下面分模块对接收端的处理程序进行详细的说明。

4.3.2.1 系统控制模块设计说明

系统控制模块结构设计图如下：

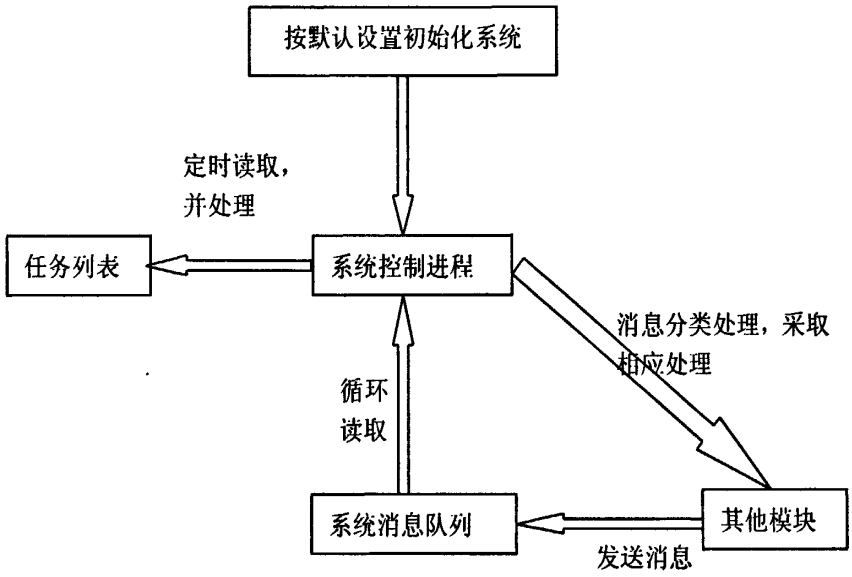

图 4-18 接收端系统控制模块结构

控制流程：系统控制进程读取默认参数表，加载参数启动相应任务，系统控制进程循环读取系统消息队列。消息分类处理后，将控制信息发送给对应的模块，采取相应的处理。系统控制进程创建定时器，定时检查任务列表，并对各任务做出相应的处理。其他功能模块的消息进入消息队列时采取优先级控制原则，高优先级的消息先处理，低优先级的消息后处理。系统任务结束时，系统控制进程定时读取任务列表，将各任务停止。

- 默认配置

在系统运行过程中，与运行相关的参数都在默认配置中。参数的配置信息是系统控制进程的运行策略。系统中采用一个数据结构sctCIB来记录默认参数。参数包括：系统控制进程的运行目录，各服务器的IP，端口等等。

- 系统控制进程（SCP）

系统控制进程主要实现以下功能：

1、负责整个系统的正常运行；

2、接收其他各个模块的消息；

3、处理各个模块的消息并作出反馈；

对应的实现如下：

1、为了保证整个系统的正常运行，系统控制进程定时检查任务列表，对任务进行及时的管理。

2、接收其他各个模块的消息，是通过消息队列实现的。系统控制进程读取消息，并进行分类处理。比如任务重启，状态上报，心跳等。消息队列是进程间

的一种通信机制，它允许进程向其他进程发送消息以及从其他进程接收消息。

本模块中的消息分发机制如下：

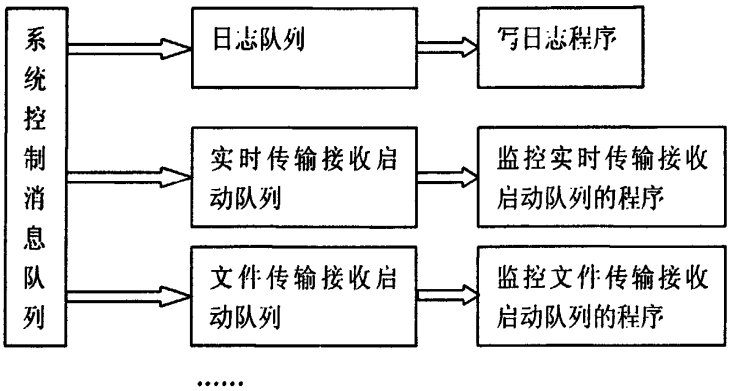

图4-19 接收端消息分发机制

首先，系统控制消息队列是全局队列，它综合接收各种消息，为减轻其压力，如果可以直接处理的消息，由其直接处理；需要写日志的消息存入日志队列，由相应的应用程序读取并写入日志；若是实时传输接收任务命令消息，则写入下层的实时传输接收启动队列，由相应的监控程序读取并启动相应的实时传输接收进程或线程；若是文件传输接收任务命令消息，则写入下层的文件传输接收启动队列，由相应的监控程序读取并启动相应的文件传输接收进程或线程。

写入日志文件的消息格式：

表4-1 日志消息格式

消息名称	字段	意义
msg_log	mType	消息类型
	mModule	模块号
	mPriority	优先级
	mContent	消息内容

关键接口的定义和封装：

创建队列：int initqueue(key_t key)，内部封装了系统调用msgget。

从队列取日志消息：int msgread_log(msg_log_id, &logstatus)，内部封装了系统调用msgrcv。

从队列取消息：int msgread(int queueid, void *buf)，内部封装了系统调用msgrcv。

向队列放状态消息：int msgwriteStatus(int queueid, char *buf)，内部封装了系统调用msgsnd。

删除队列：int remmsgqueue(int queueid)，内部封装了系统调用msgctl。

3、系统控制进程向其他模块传送反馈消息，比如通过调用函数更改参数，通过socket连接上报消息等。

系统控制进程的流程如下：

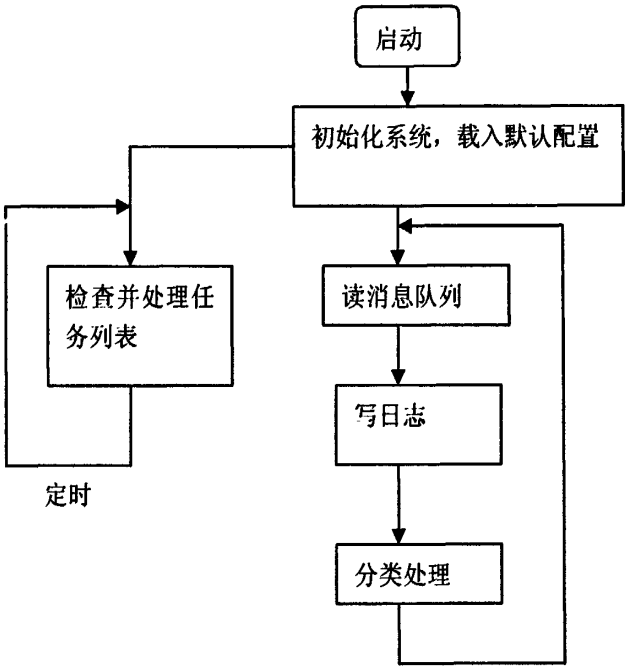

图 4-20 接收端系统控制进程流程图

4.3.2.2 接收端实时接收模块设计说明

接收端实时接收模块结构设计图如下：

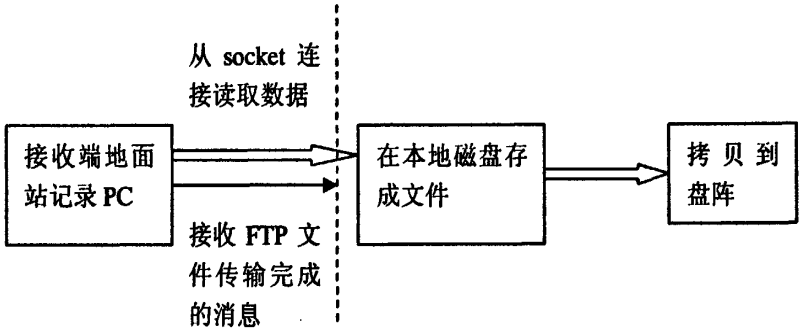

图 4-21 接收端实时接收模块

接收端实时接收模块跟接收端地面站有两条连接，一条接收FTP文件传输完成的消息，另一条从socket连接读取实时数据，然后在本地磁盘存成文件，然后将文件拷贝到接收端盘阵上。

接收端实时接收模块主要负责接收来自接收端地面站的XXXX星数据，这个接收是实时的，并且XXXX星数据并未进行分离，所以直接存入磁盘即可。和接收端地面站的连接采取TCP/IP的SOCKET编程。

XXXX星数据接收线程由系统控制进程创建，主要实现以下功能：

正确判断数据的类型信息；

正确接收并存储来自接收端地面站的XXXX星数据；

对应的实现如下：

每一个数据的传输都有开始消息，同时包含数据的信息，判断数据标志位即可。

XXXX星数据接收线程流程如下图：

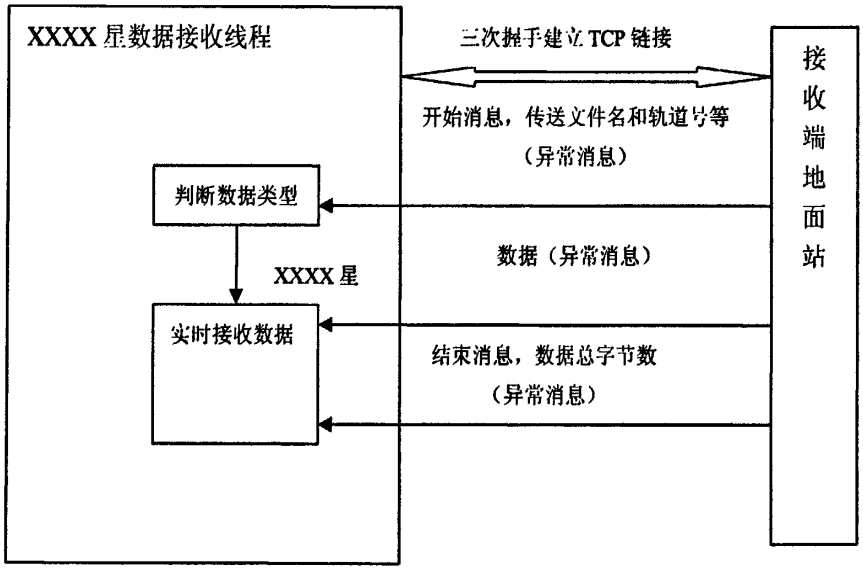

图 4-22 XXXX 星数据接收线程

在XXXX星数据接收线程被创建的同时，该线程就要请求和接收端地面站的建立链接，这个过程还是由TCP的SOCKET实现。在这个过程中，没有数据转发的需求，所以直接将数据存入磁盘中并移入盘阵，无需做任何处理。

程序逻辑：

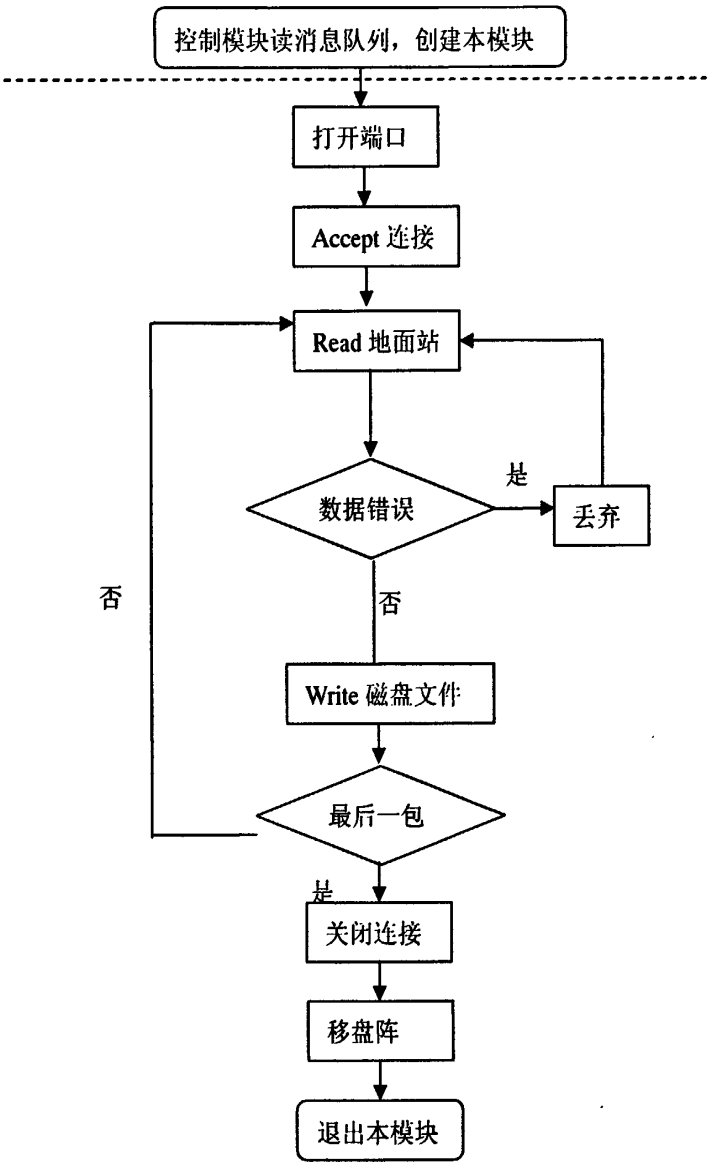

图 4-23 接收端实时接收模块程序逻辑

地面站接收文件数据消息格式：

表4-2 地面站接收文件数据消息格式

消息名称	字段	意义
GroundYKMsg	MsgLevel	消息级别
	DType	消息类型
	JobId	作业号
	SatId	卫星代号
	OrbNum	轨道号
	DFrom	信源标识
	DTo	信宿标识
	DTime	日期时间
	DLen	全包长
	FGSOverMsg	结束标志

主要接口：

int gsRecvProc(int tid);

static void * gsDataRecv(void *arg);

本模块开始处，控制模块读取消息队列，一旦发现有数据接收的需求，就启动本模块。

4.3.2.3 发送端实时接收模块设计说明

发送端实时接收模块的结构设计如下图：

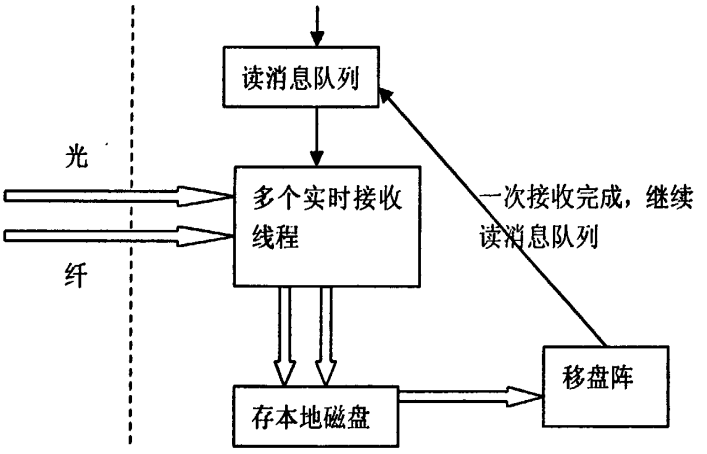

图 4-24 发送端实时接收模块结构图（接收端）

本模块包括一个主进程，一个实时数据接收控制线程，多个实时数据接收线程。

- 主进程

主进程由系统控制进程创建，主要实现以下功能：

1、负责本模块的正常运行；

2、向系统控制模块的消息队列传送数据接收的状态和告警等信息；

3、接收系统控制模块的反馈信息及控制命令；

4、初始化模块配置；

5、读消息队列，启动实时接收。

对应的实现如下：

1、主进程负责对整个模块的控制，所以对整个模块的异常都有相应的处理措施。

2、采用消息队列。

3、接收控制命令并作出相应的动作。

4、启动XXXX星数据接收线程和实时数据接收管理线程，加载一些默认的配置参数，采用数据结构sctCIB来记录配置信息。配置信息主要包括数据接收默认的存储目录路径，接收端口，默认的接收方式等等。

主进程的流程如下图：

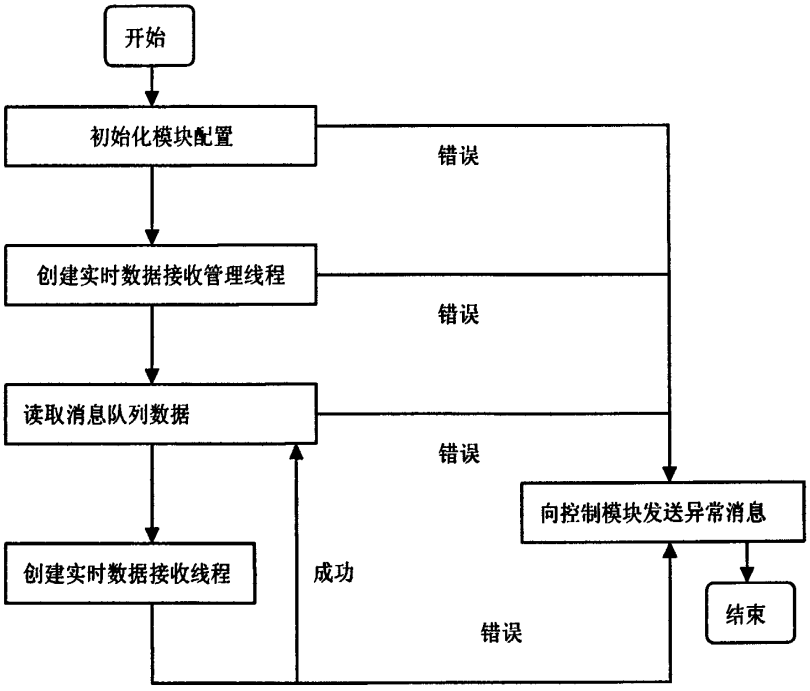

图 4-25 主进程的流程

- 实时数据接收控制线程

实时数据接收控制线程主要实现以下功能：

1、创建多个实时数据接收线程；

2、管理各个实时数据接收线程；

对应的实现如下：

1、采用线程函数实现。由实时数据接收控制线程和发送端实时传输子系统保持链接，监听端口，查看是否有实时数据到来。每当有实时数据到来，则创建多个线程来接受，但是，为了系统的可靠性，线程的个数是有限制的，不能无限增长。从效率的角度，多次试验能得出一个最佳的线程上限。

2、管理多个实时数据接收线程的同步等操作。

- 实时数据接收线程

实时数据接收线程由实时数据接收控制线程在监测到有实时数据到来的时候创建。主要完成实时数据的接收。

- 数据重组模块设计说明

在多个实时数据接收线程中，数据的到达是乱序的，所以在本地存盘之前，需要对数据进行重组处理，使之有序。

数据重组模块是一套函数库，供存储模块调用。函数的主要功能是：为每一份数据动态创建并打开一个文件。每接收一个数据包，根据该数据包的数据标识，在文件描述符中找到相应的文件指针（若文件描述符中不存在相应的数据标识，则新建一个文件描述符），读取数据包的数据开始点和数据结束点，将数据写入到对应的文件中，并将文件指针指向文件开始位。当数据全部到齐后，关闭文件指针，存盘结束。

控制模块一旦检查到接收发送端的实时数据接收消息，就创建该模块。打开端口，等待发送端等的连接，握手成功，连接建立，创建多个接收线程，待全部数据接收完毕后，将多个接收线程结束，关闭各个socket连接。

程序逻辑：

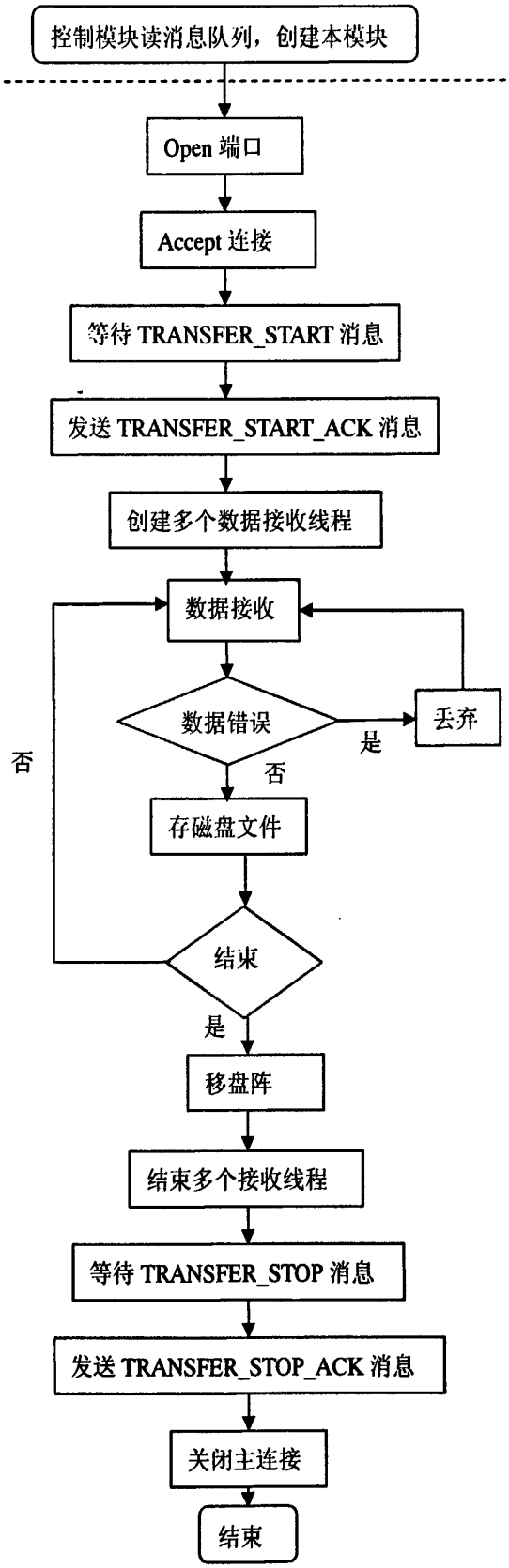

图 4-26 发送端实时接收模块程序逻辑

建立socket连接过程的控制消息格式：

表4-3 建立socket连接过程的控制消息格式

消息名称	字段	意义
ControlMessage	controlType	控制类型
	threadcount	线程数
	packageId	包号
	FileName	文件名
	LocalPath	当地路径

主要接口：

int processmain(int rcv_ctlSock,int datalissock,int tid);

void *data_receive(void *arg);

4.3.2.4 文件传输接收模块设计说明

文件接收模块的结构设计如下图：

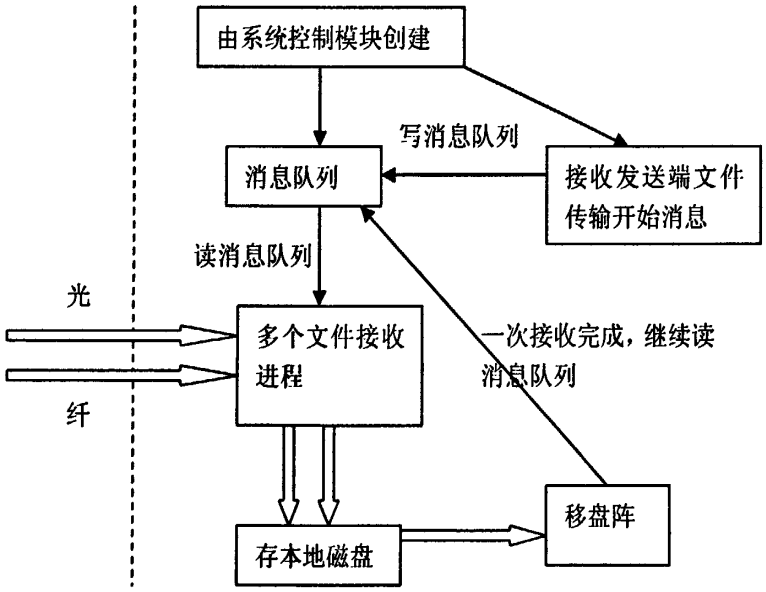

图 4-27 发送端实时接收模块结构图（接收端）

本模块有一个主控制进程，一直在读消息队列，一旦读到一条消息，则启动

frx文件传输及其配置，创建16个文件接收进程，接收来自发送端的文件数据，接收完成，frx进程退出，文件接收主控进程向系统控制模块的消息队列发送成功消息，并继续读消息队列。

程序逻辑：

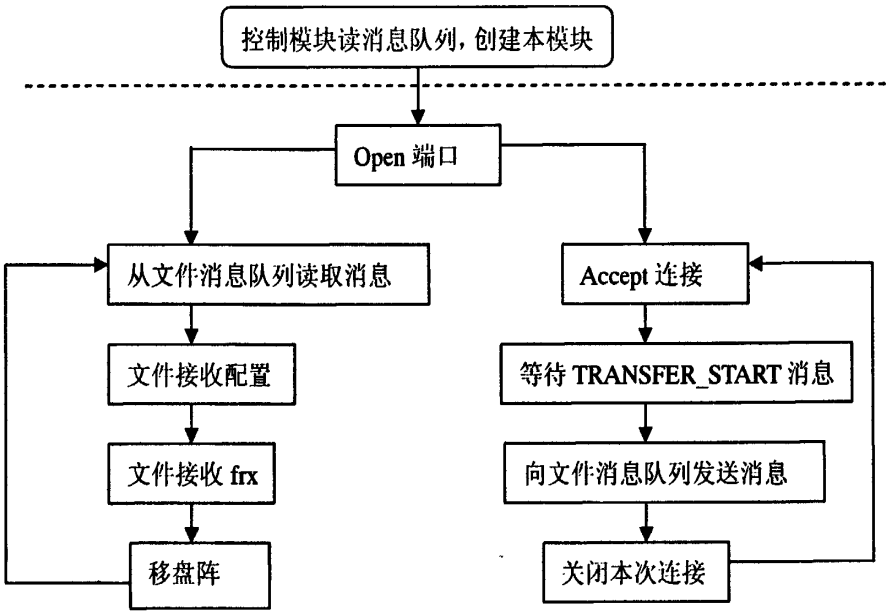

图 4-28 文件传输接收模块

文件传输的控制消息：

表4-4 文件传输的控制消息格式

消息名称	字段	意义
ControlMessageFileTran	controlType	控制类型
	type	消息类型
	loopnum	循环次数
	FileName	文件名
	pMsg	运控命令附带的信息

主要接口：

```
void *openPort(void *arg);
void *ReadMsg(void *arg);
int filetrans();
int filetransProc(int tid);
```

4.3.2.5 运控交互模块设计说明

运控交互模块的结构设计如下图：

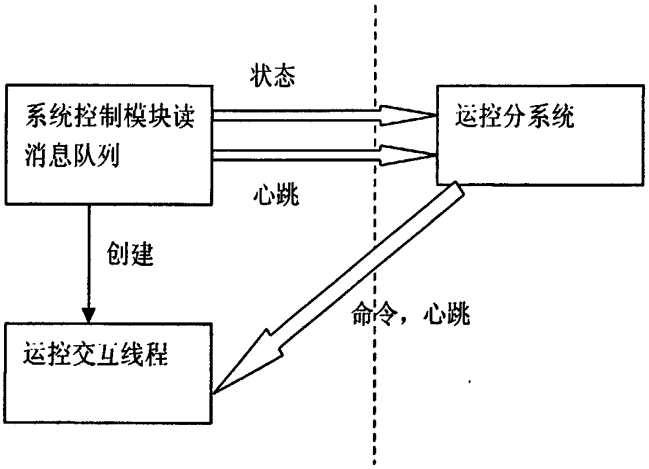

图 4-29 运控交互模块

程序逻辑：

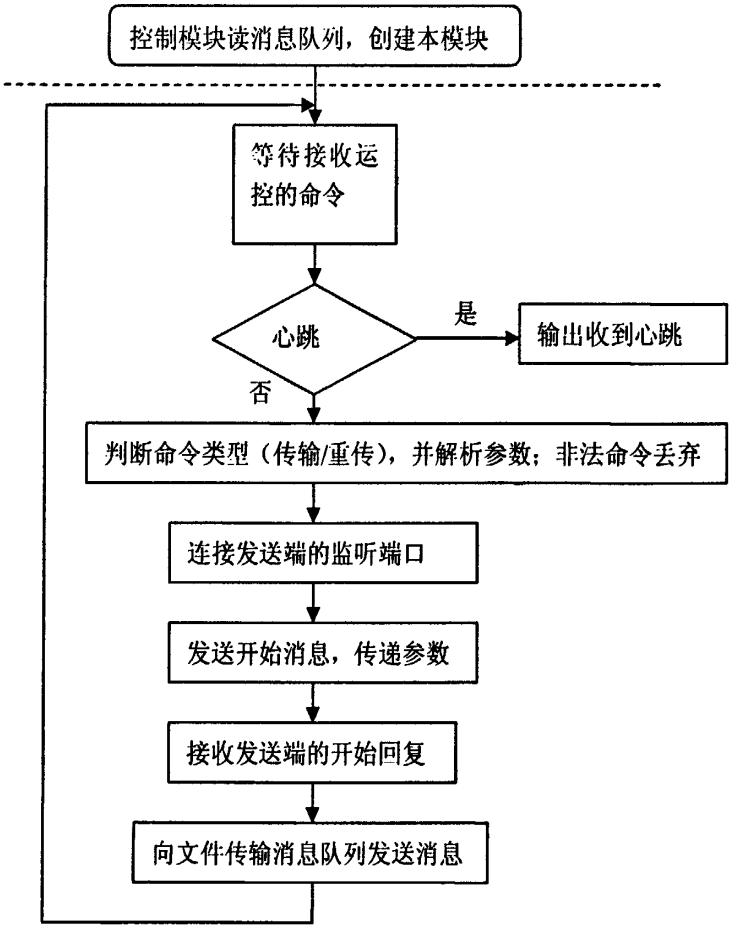

图 4-30 运控交互模块程序逻辑

系统控制模块循环读消息队列，一旦读到状态或者心跳消息，则向运控分系统上报；系统控制模块在启动时创建了运控交互线程，此线程一直在读来自运控的消息，一旦收到心跳，则说明通信分系统和运控分系统尚处于连接状态，否则，说明连接断开；若收到传输重传命令，则根据参数启动相应的文件传输接收模块，并且连接发送端的文件传输监听端口，令其启动文件传输的发送模块，最后完成文件传输，退出，继续接收运控的命令消息。

与运控交互的消息格式，也就是实时传输子系统跟运控的外部接口：

表4-5 与运控交互的消息格式

消息名称	字段	意义
运行状态：IMSRunStateReport	StateCode	状态码
	ParamNum	参数个数
	RunState	状态内容
传输命令/回令：ISKDcsSchCmdTran	StationID	站标
	FromFileName	远程机器文件名
	LocalFileName	本地文件名
重传命令/回令：ISKDcsSchCmdRetran	StationID	站标
	FileType	传输文件类型
与运控交互的完整消息格式：ISKOCSDataMessage	MsgLevel	信息级别
	DType	信息类型
	JobId	作业号
	SatId	卫星代号
	OrbNum	轨道号
	DFrom	信源标识
	DTo	信宿标识
	DTime	日期时间
	DLen	全包长
	text	具体消息内容

主要接口：

```
void rcv(void);
void connectSy(int type,YKMessage p,int loopnum);  //18-tran,17-retran
```

void sendtoyk(char *c,int DCSType);

4.3.2.6 监控交互模块设计说明

监控交互模块负责监控PC与传输系统之间的交互，功能包括两个部分：从控制模块定期取传输状态信息，告警信息，还有监控PC参数改变命令执行结果等，经过处理后，发送给监控PC，由监控PC将其显示在人机交互界面上，方便管理员观察传输状态；接收来自监控PC的传输系统参数改变命令，经过处理后，发送给系统控制模块，由系统控制模块及时进行修改。

监控交互模块主要负责将实时的传输状态，进程调度情况等发送给监控PC，消息格式如下：

传输开始：状态标识/站标/星号/时间/轨道号/文件名/0

传输状态：状态标识/站标/星号/时间/轨道号/文件名/当前传输的字节数

传输异常：状态标识/站标/星号/时间/轨道号/文件名/异常原因

传输结束：状态标识/站标/星号/时间/轨道号/文件名/传输总的字节数

进程状态：状态标识/进程标识/进程状态

主要接口：

int sendToJk(int sock, char *status);

参数sock表示与监控PC连接的套接字，status表示要传输的状态消息，在函数中作格式处理。

4.3.3 发送端处理程序概要及详细设计

下面分模块对发送端的处理程序进行详细的说明。

4.3.3.1 系统控制模块设计说明

发送端实时传输子系统的系统控制模块的程序逻辑设计与接收端实时传输子系统的系统控制模块设计相同。

主要接口：

static int startProcess(int tid);

static void tasktblInit(void);

static int taskAdd(int tid,int pid);

static void taskClear(int tid);

static void tasktblDel(int tid);

static int taskparInit(void);

```
static void uTaskClose(int uTid);
static void uTaskCreateFault(int uTid);
static int uTaskStart(int uTid,int mode);
static int uTaskparInit(void);
static int realTranErrhandle(mainPrcMsg *msg);
static void * msgDisposal(void *arg);
static int initialLog(void);
```

本模块中的消息分发机制如下：

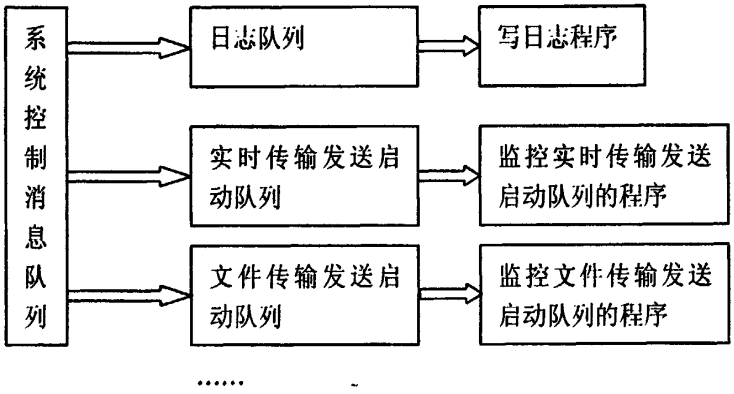

图 4-31 发送端消息分发机制

首先，系统控制消息队列是全局队列，它综合接收各种消息，为减轻其压力，如果可以直接处理的消息，由其直接处理；需要写日志的消息存入日志队列，由相应的应用程序读取并写入日志；若是实时传输发送任务命令消息，则写入下层的实时传输发送启动队列，由相应的监控程序读取并启动相应的实时传输发送进程或线程；若是文件传输发送任务命令消息，则写入下层的文件传输发送启动队列，由相应的监控程序读取并启动相应的文件传输发送进程或线程。

写入日志文件的消息：

表4-6 日志消息格式

消息名称	字段	意义
msg_log	mType	消息类型
	mModule	模块号
	mPriority	优先级
	mContent	消息内容

关键接口的定义和封装：

创建队列：int initqueue(key_t key)，内部封装了系统调用msgget。

向文件传输启动队列放消息：int msgwriteFiletran(int queueid,int type,unsigned short port,int orbit,char *ip,char *path,char *filename, char *Localfilename)，内部封装了系统调用msgsnd。

向日志队列写日志消息：int msgwriteLog(int queueid, void *buf, int mod,int pri, int type)，内部封装了系统调用msgsnd。

从日志队列取日志消息：int msgread_log(int queueid, void *buf)，内部封装了系统调用msgrcv。

删除队列：int remmsgqueue(int queueid)，内部封装了系统调用msgctl。

4.3.3.2 发送端实时接收模块设计说明

监听发送端地面站，若有连接且有数据，则判断数据是否是即将要接收的数据，若是，则接收，否则，拒绝接收。接收发送端地面站的数据，存到本地磁盘。接收完成，继续监听。

工作流程如下。

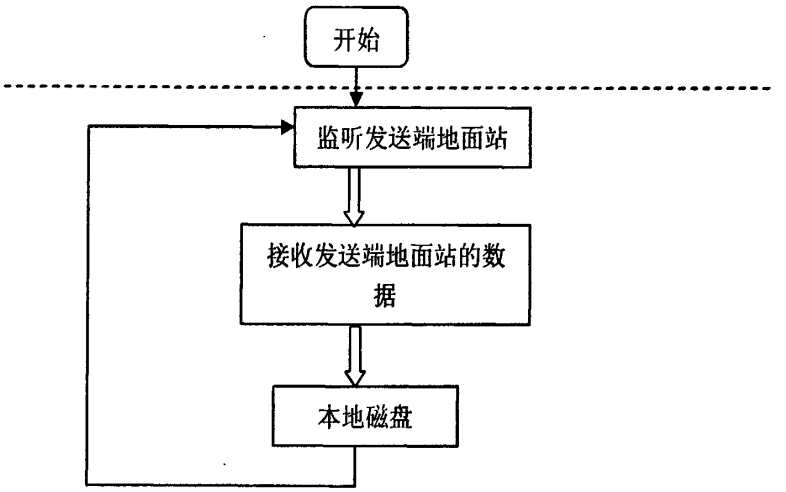

图 4-32 实时接收模块流程

接收本地地面站的卫星数据是通过TCP/IP的SOCKET，同样基于C/S模型，此链接的客户端是本地地面站，服务器端是本地接收模块，在接收进程启动之后，客户端就向服务器端发送链接请求，链接成功后，发送数据，结束时断开连接。

主要的控制流程如下：

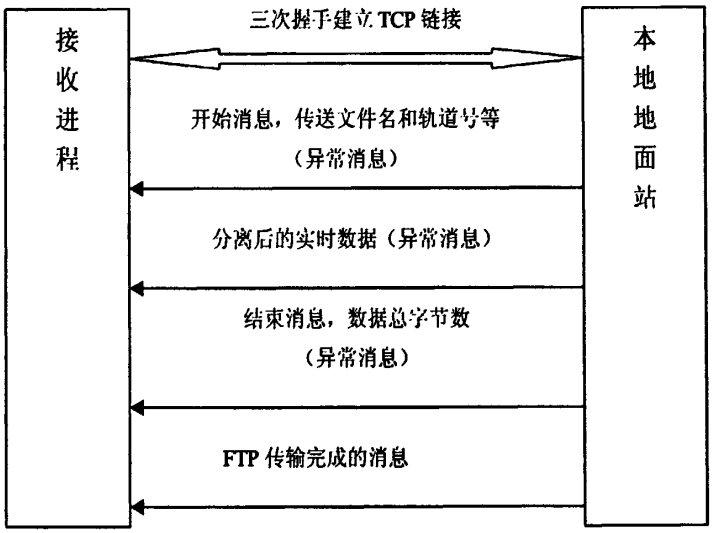

图 4-33 本地地面站的接收流程

通过TCP协议中自带的反馈信息来判断传输是否成功，在TCP/IP的API中，读取accept等函数的返回值，并进行传输错误的处理。

接收到的实时数据是要通过处理才能进行转发的，因为转发的数据有一定的格式规定，以方便接收端通信服务器的接收存储。

控制模块一旦检测到有来自地面站的传输消息，就启动该模块。打开端口，跟地面站建立socket连接，接收实时数据，将其发送到缓冲管道中，同时写入本地磁盘文件以备份，接收完毕后，关闭连接，退出本模块。

程序逻辑：

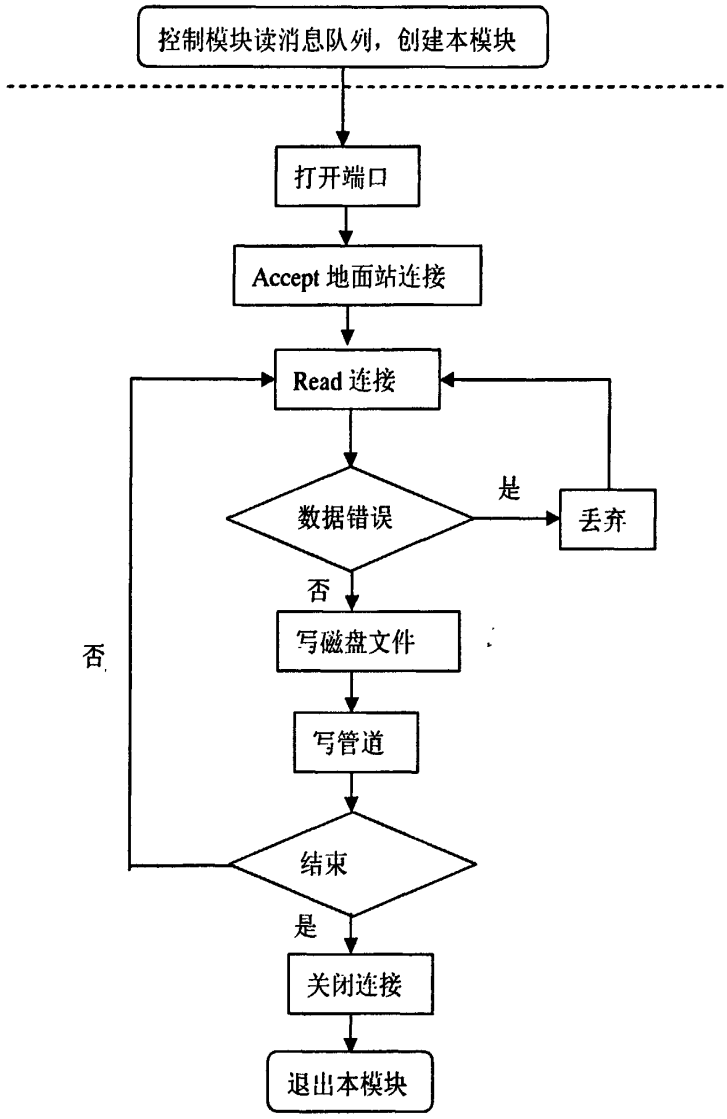

图 4-34 发送端实时接收模块程序逻辑

主要接口：

int gsRecvProc(int tid);

void * gsDataRecv(void *arg);

int procSendData(procCommu *p,void *buf,int len);

4.3.3.3 发送端实时转发模块设计说明

发送端实时转发模块的结构设计如下图：

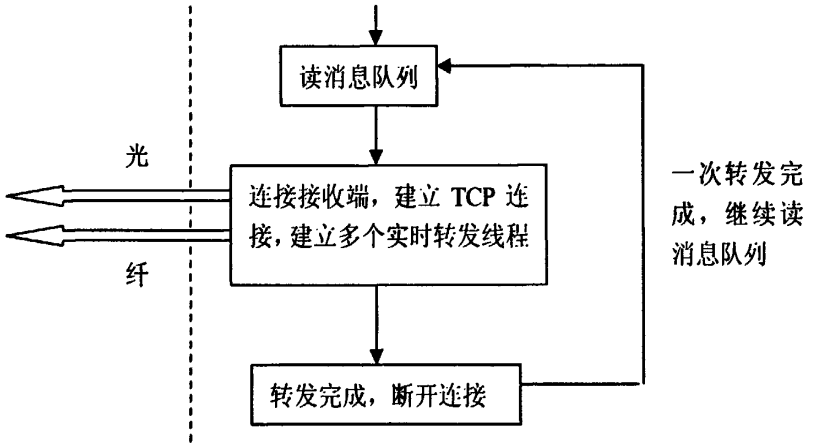

图 4-35 发送端实时转发模块结构图

本模块包括一个主进程，一个实时数据转发控制线程，多个实时数据转发线程。

- 主进程

主进程由系统控制进程创建，主要实现以下功能：

1、负责本模块的正常运行；

2、向系统控制模块的消息队列传送数据转发的状态和告警等信息；

3、接收系统控制模块的反馈信息及控制命令；

4、初始化模块配置；

5、读消息队列，启动实时发送。

对应的实现如下：

1、主进程负责对整个模块的控制，所以对整个模块的异常都有相应的处理措施。

2、采用消息队列。

3、接收控制命令并作出相应的动作。

4、启动实时数据发送管理线程和实时数据发送线程，加载一些默认的配置参数，采用数据结构sctCIB来记录配置信息。配置信息主要包括数据接收默认的存储目录路径，发送端口，默认的发送方式等等。

主进程的流程如下图：

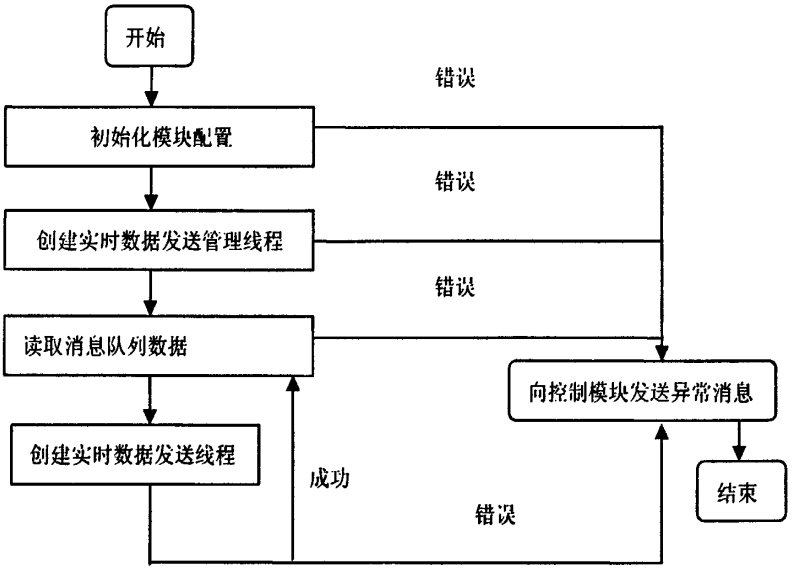

图4-36 主进程的流程

- 实时数据发送控制线程

主要功能如下：

1、初始化配置信息；

2、创建多个实时数据发送线程；

3、负责转发模块的正常运行，管理各个实时数据发送线程；

4、准确转发实时数据到接收端通信服务器。

对应的实现如下：

1、初始化一些基本信息，包括sever端的IP地址，端口等；

2、采用线程函数实现。收到控制模块的发送实时数据的命令后，连接接收端的实时数据接收控制线程，成功后，创建多个线程来发送，但是，为了系统的可靠性，线程的个数是有限制的，不能无限增长。从效率的角度，多次试验能得出一个最佳的线程上限；

3、管理多个实时数据发送线程的同步等操作，向系统控制模块传送数据转发的状态以及告警等信息；接收系统控制模块的反馈信息及控制命令；

4、与对端服务器建立主socket连接。光纤链路正常时，正常转发，并定时向控制模块发送状态信息，最后发送转发完成消息，自动退出转发线程；光纤链路异常时，向控制模块实时发送告警信息，并自动退出转发线程；

- 实时数据转发线程

实时数据转发线程由实时数据转发控制线程在监测到有实时数据到来的时候创建。主要完成实时数据的转发。在同步与互斥控制下，各线程从磁盘中读取

数据；通过光纤链路将实时数据发送到服务器端；记录已转发成功的最后数据包的编号。

读实时转发启动消息队列，若读到消息，则连接接收端的实时数据监听端口，建立连接，然后建立多条TCP连接，转发数据，并向控制模块上报转发状态，异常等。转发完成，以上连接全部断开，继续读实时转发启动消息队列。

程序逻辑：

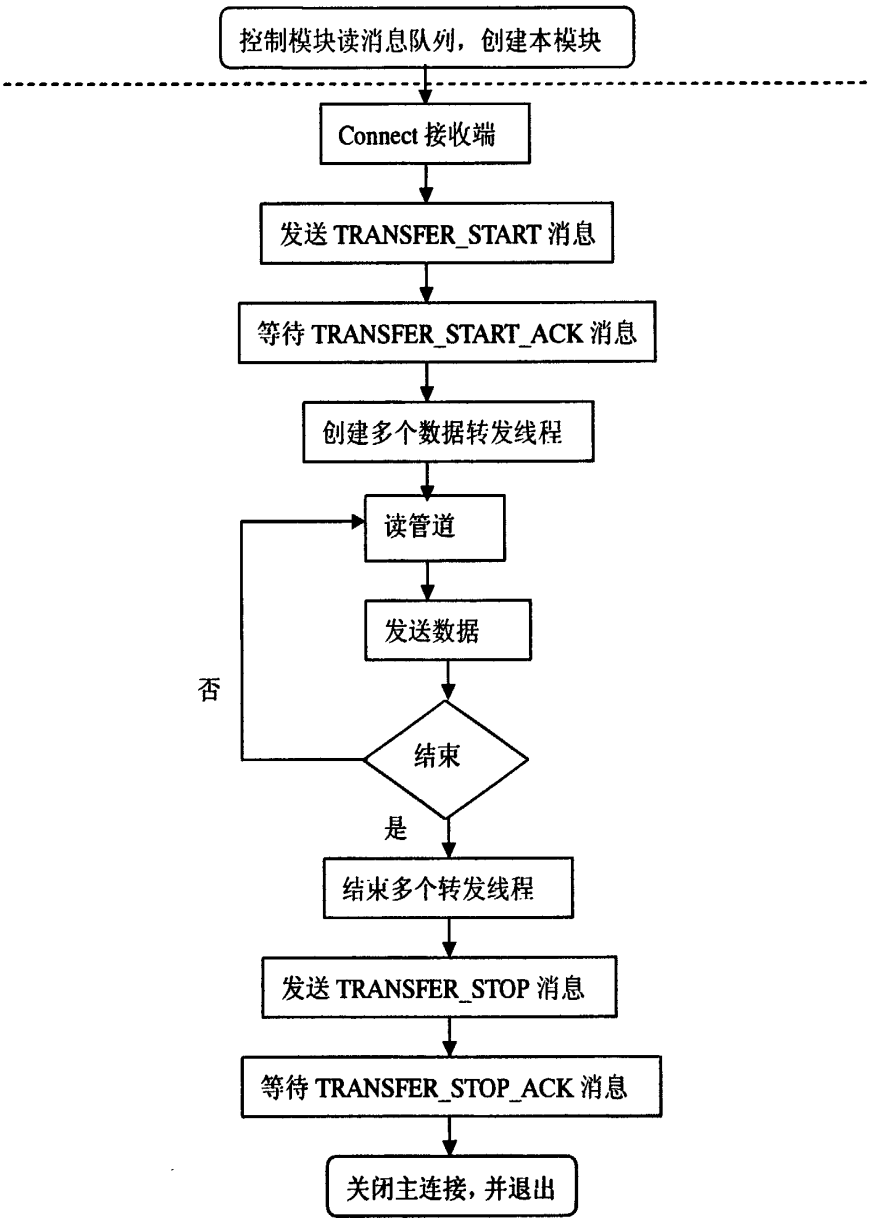

图 4-37 发送端实时转发模块程序逻辑

建立socket连接过程的控制消息：

表4-7 建立socket连接过程的控制消息格式

消息名称	字段	意义
ControlMessage	controlType	控制类型
	threadcount	线程数
	packageId	包号
	FileName	文件名
	LocalPath	当地路径

主要接口：

int msTransProc(int tid);

void *msDataTrans(void *arg);

int sendCTRLMessage1 (int sockfd, CTRLTYPE type, int count, int num,char *filename);

int waitAck1 (int sockfd, CTRLTYPE type,char *filename);

int bhyInitial(int num);

void *data_trans(void *arg);

int sendFileData(int fd,int pktstart,int pktend);

static int sendpack(int len);

4.3.3.4 监听接收端运控命令模块设计说明

监听接收端运控模块连接，接收端实时传输子系统的运控模块一旦来连接，则等待接收端实时传输子系统的运控模块发来的开始消息，接收到开始消息之后，对参数进行解析，然后将文件开始消息发送到文件传输启动消息队列，然后向接收端实时传输子系统的运控模块发送开始回复消息。关闭连接，继续监听来自接收端运控模块的连接。

工作流程如下。

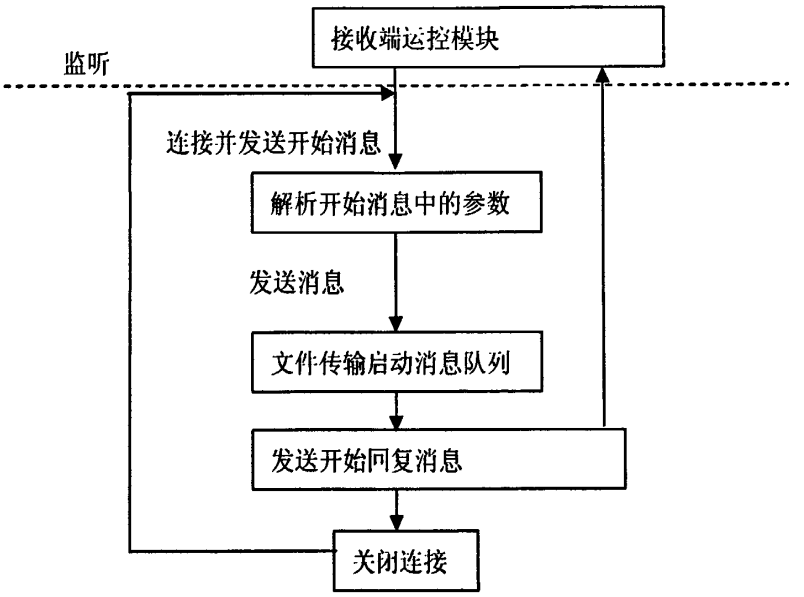

图 4-38 监控运控命令流程图

程序逻辑：

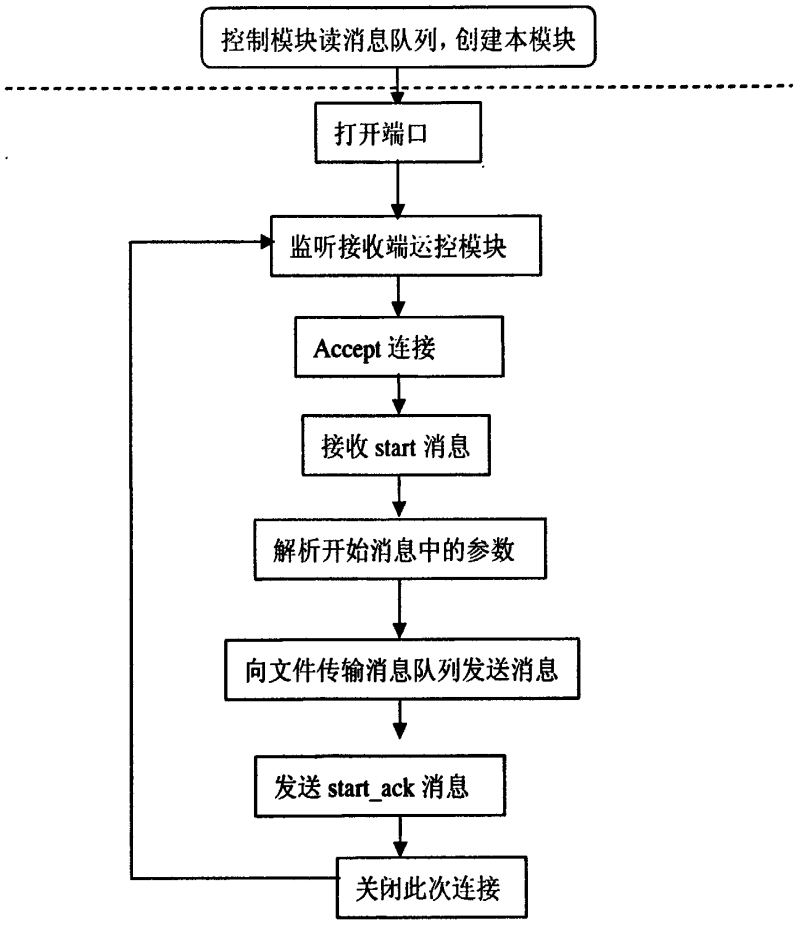

图 4-39 监听运控命令模块程序逻辑

主要接口：

int filetranyk(void);

void *tranRetran(void *arg);

4.3.3.5 文件传输发送模块设计说明

一方面，接收控制模块的命令，连接接收端的文件传输监听端口，连接成功，则将文件发送启动的消息发送到文件传输消息队列。然后给接收端发送开始消息。关闭连接。继续等待控制模块的命令。

另一方面，读文件传输消息队列，若有消息，则启动发送文件进程，向接收端发送文件，发送完成，退出，继续读消息队列。

具体流程如下。

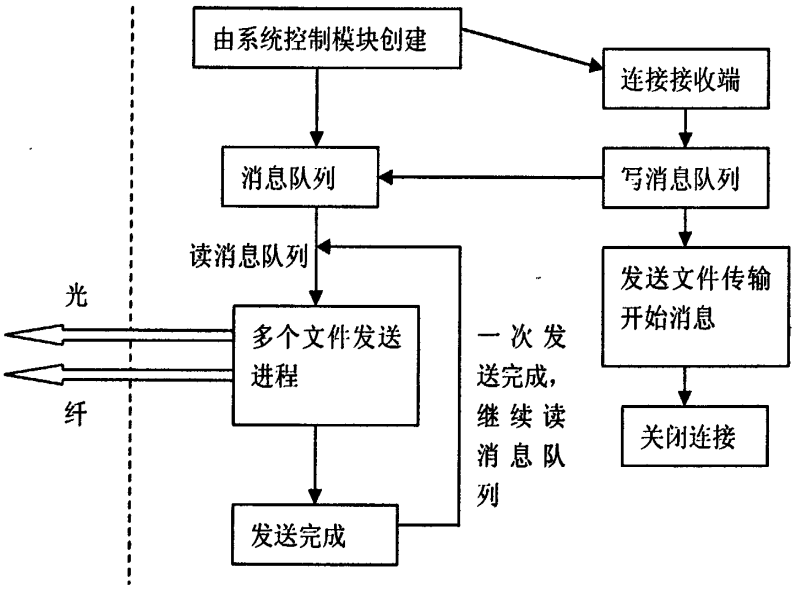

图 4-40 文件传输发送模块

程序逻辑：

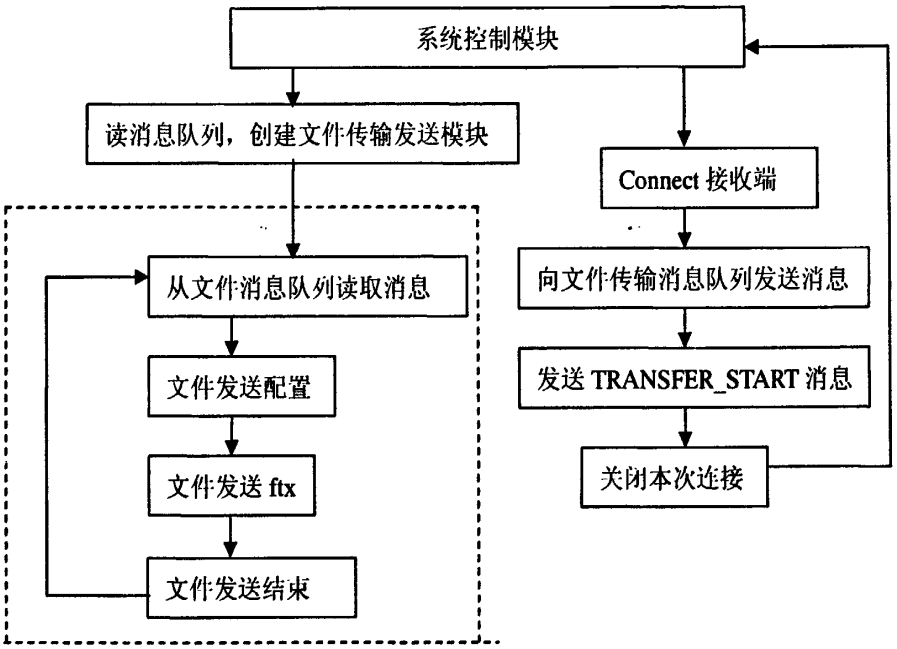

图 4-41 文件传输发送模块程序逻辑

文件传输的控制消息：

表4-8 文件传输的控制消息格式

消息名称	字段	意义
ControlMessageFileTran	controlType	控制类型
	type	消息类型
	loopnum	循环次数
	FileName	文件名
	pMsg	运控命令附带的信息

主要接口：

int sendMsgFT(int mqid,char *path,char *filename,char *ip,unsigned short port,int orbit);

int filetransProc(int tid);

int filetrans(int tid);

void *ReadMsg(void *arg);

int msgwriteFiletran(int queueid,int type,char *ip,char *path,char *filename);

int sendCTRLMessage1 (int sockfd, CTRLTYPE type, int count, int num,char

*filename);

4.3.3.6 监控交互模块设计说明

监控PC交互模块设计与接收端实时传输子系统的监控PC交互模块设计相同。

4.3.3.7 文件传输模块设计说明

文件传输模块独立于实时传输模块，在光纤链路断开的情况下，采用卫星链路连接，进行文件传输。在实现中，文件传输模块被单独设计为一个模块，当光纤链路发生故障时，停止实时传输，启动已经编译且链接好的执行文件，进行文件传输。本模块的设计采用C/S结构设计。在各种情况下，文件的发送端都作为服务的提供端。本系统对被传输的数据文件的操作主要包括两方面：发送方对数据文件压缩预处理及发送缓存；接收方对分段的数据文件的组合及解压缩[22]。

第五章　系统测试

本章介绍整个系统的测试情况，首先介绍了整个系统的运行环境，然后介绍了测试计划及方案，最后对测试结果进行了分析。

5.1 环境介绍

通信分系统分为实时传输子系统和设备监控子系统两部分。实时传输子系统运行在 IBM 的 AIX 平台下，设备监控子系统运行在 Windows 2003 平台下。

AIX（Advanced Interactive eXecutive）是 IBM 开发的一套 UNIX 操作系统。它符合 Open group 的 UNIX 98 行业标准(The Open Group UNIX 98 Base Brand)，通过全面集成对 32-位和 64-位应用的并行运行支持，为这些应用提供了全面的可扩展性。

AIX 5L 是 AIX 的当前使用版本，它支持 IBM POWER 和 Intel 64 位(IA-64)平台。1）通过在 AIX 5L 中引入动态逻辑分区（DLPAR），IBM 为基于 POWER4 的 p 系列系统提供了高级的灵活性和可扩展性功能。2）运行效率和容量规划。为提供更高的灵活性、可扩展性和可用性，AIX 5L 在 p650，p670 和 p690 系统上推出了动态随需应变容量升级（CUoD）功能。CUoD 允许客户在进行系统安装时安装比初始需要数量更多的处理器，保持这些处理器（处于休眠状态）直至业务的增长要求将其激活。CUoD 选项将为系统管理员提供一个用于激活更多处理器的加密密钥，可以在不中断系统运行的情况下将新激活的处理器动态分配给各个分区。CUoD 提高了系统可用性。3）集群管理。为实现快速同步和协调响应，集群环境要求节点之间能够进行全面的协作。AIX 5L 使用基于 AIX 5L 的 UNIX 软件和 IBM 集群系统管理器（CSM）支持和优化集群服务器的管理。4）高可靠性。利用自身的软件，HACMP 可以实现双机切换，确保高稳定性的实现。5）UNIX 亲和性。AIX 5L 与 UNIX 之间的亲和性可以帮助以速度更快、成本更低的方式实现跨 AIX 和 UNIX 平台的多平台集成解决方案。6）安全性。通过 C2 级认证的 AIX 5L 提供并全面使用了强大的行业标准安全技术和目录技术。

5.2 测试方案及结果介绍

对本系统的测试，主要根据系统的需求分析中的功能和性能需求来进行。在系统的实际运行环境中，对系统的每个功能和性能，选取合适的一组或几组实际

数据进行测试。

测试结果如下。

表 5-1 实时遥感数据传输功能测试

用例	用例描述	预 期 结 果	测试结果
接收端实时传输	开始时间	02:02:06	
	结束时间	02:13:59	
	文件个数	2	
	文件大小	H1BIB090326020210207.ORG.DAT=292262400 H1BQB090326020210207.ORG.DAT=292278450	
	文件正确性	二进制匹配正确	与预期结果一致
发送端实时传输	开始时间	02:22:50	
	结束时间	02:35:10	
	文件个数	2	
	文件大小	H1BQS090329022210250.ORG.DAT=304053585 H1BIS090329022210250.ORG.DAT=303820800	
	文件正确性	二进制匹配正确	与预期结果一致

表 5-2 实时数据静态备份功能测试

用例	用例描述	预 期 结 果	测试结果
接收端实时传输	备份文件大小	文件名：H1BQB090328024410236.ORG.DAT 来源一：接收记录微机，346030080 字节 来源二：通信服务器备份文件存放目录，346030080 字节	

	备份文件的正确性	二进制匹配正确	与预期结果一致
发送端实时传输	备份文件大小	文件名：H1BQS090329022210250.ORG.DAT 来源一：接收记录微机，304053585 字节 来源二：通信服务器备份文件存放目录，304053585 字节	
	备份文件的正确性	二进制匹配正确	与预期结果一致

表 5-3 工作日志功能测试

用例	用例描述	预 期 结 果	测试结果
接收端实时传输	日志内容	记录了传输过程中，任务启动，传输状态，进程调度，任务结束等信息	与预期结果一致
	日志时间	当天的传输过程时间	与预期结果一致
发送端实时传输	日志内容	记录了传输过程中，任务启动，传输状态，进程调度，任务结束等信息	与预期结果一致
	日志时间	当天的传输过程时间	与预期结果一致

表 5-4 软件配置功能测试

用例	用例描述	预 期 结 果	测试结果
系统发送端配置	配置帮助	屏幕显示配置参数和参数代表的意义	与预期结果一致
	配置出错提示	屏幕显示参数出错的提示和出错原因	与预期结果一致
	正确配置保存	屏幕正确显示配置完成后的详细信息	与预期结果一致
系统接收端配置	配置帮助	屏幕显示配置参数和参数代表的意义	与预期结果一致
	配置出错提示	屏幕显示参数出错的提示和出错原因	与预期结果一致

	正确配置保存	屏幕正确显示配置完成后的详细信息	与预期结果一致

表 5-5 断点续传功能测试

用例	用例描述	预 期 结 果	测试结果
发送端到接收端的文件传输	以文件方式传输发送端[A01S.09088.1705.PDS/GBD]到接收端通信服务器，传输过程中人为断开光纤和 VSAT 链路，3 分钟后恢复链路后，重新启动程序传输该文件	文件应该从链路断开时的状态续传，且传输完毕后目标文件与源文件一致。	与预期结果一致

表 5-6 运控心跳功能测试

用例	用例描述	预 期 结 果	测试结果
每次系统的运行	只要系统运行，就应该交互心跳	每 5 分钟，向运控发送心跳成功，并立即收到运控回复的心跳消息	与预期结果一致

表 5-7 数据传输状态报告功能测试

用例	用例描述	预 期 结 果	测试结果
H1BIB090329035510251.ORG.DAT H1BQB090329035510251.ORG.DAT	接收端地面站的实时传输	成功上报传输开始，中间传输量，传输结束和告警的消息	与预期结果一致
H1BIS090329040110251.ORG.DAT H1BQS090329040110251.ORG.DAT	发送端到接收端的实时传输	同上	与预期结果一致
A01B.09088.0635.GBD A01B.09088.0635.GBD.OK A01B.09088.0635.PDS A01B.09088.0635.PDS.OK	接收端本地的文件传输	同上	与预期结果一致

A01S.09088.0453.PDS A01S.09088.0453.PDS.OK A01S.09088.0453.GBD A01S.09088.0453.GBD.OK	发送端到接收端的文件传输	同上	与预期结果一致

表 5-8 运控重传管理功能测试

用例	用例描述	预 期 结 果	测试结果
轨道号：10122 站标：B 文件类型：A	重传接收端I,Q路数据	将指定的2个文件正确的从接收端磁盘传输到接收端盘阵上	与预期结果一致
轨道号：10122 站标：B 文件类型：C	重传接收端I路数据	将指定的1个文件正确的从接收端磁盘传输到接收端盘阵上	与预期结果一致
轨道号：10122 站标：B 文件类型：Z	重传接收端Q路数据	将指定的1个文件正确的从接收端磁盘传输到接收端盘阵上	与预期结果一致
轨道号：10208 站标：S 文件类型：A	重传发送端I,Q路数据	将指定的2个文件正确的从发送端磁盘传输到接收端盘阵上	与预期结果一致
轨道号：10208 站标：S 文件类型：C	重传发送端I路数据	将指定的1个文件正确的从发送端磁盘传输到接收端盘阵上	与预期结果一致
轨道号：10208 站标：S 文件类型：Z	重传发送端Q路数据	将指定的1个文件正确的从发送端磁盘传输到接收端盘阵上	与预期结果一致

表 5-9 运控传输管理功能测试

用例	用例描述	预 期 结 果	测试结果
站标：S 发送端全路径： /data/HY1B/H1BIS0903260347 10208.ORG.DAT 接收端全路径： /data/HY1B/H1BIS0903260347 10208.ORG.DAT	传输发送端的某个文件到接收端	将指定的1个文件正确的从发送端磁盘传输到接收端盘阵上	与预期结果一致

表 5-10 系统性能测试

组数	文件大小/文件名	传输时间(s)	传输速率(M/s)	统计结果(M/s)
第一组	174541392 A01B.09083.1955.PDS	103	1.7	1.69
	959485050 T01S.09084.0342.PDS	571	1.68	
	751027008 A01S.09084.0516.PDS	439	1.71	
	818040832 A01S.09084.0653.PDS	484	1.69	
	290151024 T01S.09084.1436.PDS	173	1.67	
第二组	265904208 A01B.09084.1858.PDS	151	1.68	1.688
	932116590 T01B.09085.0241.PDS	523	1.7	

	953796930 T01S.09085.0247.PDS	538	1.69	
	725085714 T01S.09085.0426.PDS	411	1.68	
	982490478 A01S.09085.0557.PDS	554	1.69	
第三组	426002310 A01B.09086.0335.PDS	241	1.68	1.682
	292261236 T01S.09087.1506.PDS	164	1.7	
	277556388 A01B.09087.1751.PDS	159	1.66	
	277617582 A01S.09087.1757.PDS	157	1.68	
	226688562 A01B.09087.1929.PDS	127	1.69	
<备注>传输速率=单个文件大小/传输时间 统计结果=当日所有测试文件传输速率的平均值				

表 5-11 系统强度测试

步骤	操　作	测试输入	预　期结　果	实际测试结果
1	在接收端实时传输的同时，发送端也进行实时传输	接收端实时传输文件名： H1BIB090328042410237.ORG.DAT H1BQB090328042410237.ORG.DAT 发送端实时传输文件名： H1BIS090328042410237.ORG.DAT H1BQS090328042410237.ORG.DAT	正常运行	与预期结果一致

2	在接收端实时传输的同时，发送端也进行文件传输	接收端实时传输文件名： H1BQB090329035510251.ORG.DAT H1BIB090329035510251.ORG.DAT 发送端文件传输文件名： H1BQS090329040110251.ORG.DAT H1BIS090329040110251.ORG.DAT	正常运行	与预期结果一致
3	在发送端实时传输的同时，发送端也进行文件传输	发送端实时传输文件名： H1BQS090328043110237.ORG.DAT H1BIS090328043110237.ORG.DAT 发送端文件传输文件名： T01S.09087.0235.PDS	正常运行	与预期结果一致

从以上测试结果来看，本系统在功能和性能上已基本满足了需求规格说明书上的要求，证明系统的设计和实现是合理的。

第六章　总结和展望

本文以 UNIX 进程间通信机制、socket 网络通信和多线程编程分析为基础，对现有的通信模型及技术进行分析，针对项目需求，对 XXXX 卫星实时传输子系统进行了分析和设计，并在 UNIX 平台下实现，完成了系统的各项功能和性能需求。

本次研究工作围绕以下几个主要内容展开：

1、分析现有通信模型，研究 UNIX 进程间通信方式、socket 网络通信和多线程编程等技术，对消息队列进行了仔细的研究，从消息队列机制的各个组成部分作了比较细致和深入的理解，形成系统的概念，为进一步研究打下基础。

2、对本系统进行需求分析，研究其具体的功能和性能需求，分析大体架构，为系统的设计打好基础。

3、设计本系统的通信模型和大体框架，在此基础上，对接收端和发送端分别进行概要和详细设计，包括模块划分、各个模块的功能和实现、内部接口和外部接口等。

4、在系统实际运行环境中，对整个系统进行测试。首先介绍了测试环境，然后介绍了测试方案及结果，并对结果进行了分析总结，本系统在功能和性能等各方面已基本达到了预期效果。

在现有工作的基础上，本实时传输子系统的现有实现还需要优化，功能还需进一步完善，以求获得更好的性能。另外，测试工作也有待进一步完善，目前的测试工作还不够完备，比如系统稳定性方面的测试，可以引进更好的测试方法和工具予以完成。

参考文献

[1] 张赞波，陈剑鸿 CPI_C在PC与AS_400客户_服务器连接中的应用 计算机应用 第20卷（第7期） 2000年7月 P57-P59

[2] 白静慧 Linux下基于Socket的网络通信 计算机应用 第1期（总第217期） 2008 P31-P33

[3] 李小白，刘水文，李斐 远程过程调用的体系结构以及相关参数 科技广场 2009年1月 P50-P52

[4] Matthew J. Fischer. System and method for message queue management in a power-save network. US Patents, Aug. 4, 2005.

[5] Jussl Pekka Ruutu. System and method for identifying applications targeted for message receipt in devices utilizing message queues. US Patents, Oct. 14, 2004.

[6] Jeffrey E. Stall. Method and apparatus for providing and integrating high-performance message queues in a interface environment. US Patents, Sep. 29, 2005.

[7] Nick Scott Russell. System and method for managing messages on a queue. US Patents, Apr. 14, 2005.

[8] David O. Driver, Geoffrey M. Kizer, Krishnan Srinivasan, Uday S. Hedge. Monitoring message queues and starting processing applications. US Patents, May. 24, 2007.

[9] Paul Judge, Guru Rajan. Systems and methods for adaptive message interrogation through multiple queues. US Patents, Nov. 2, 2006.

[10] Spiro Michaylov, Sanjeev Banerji, Craig W. Stanfill. Managing message queues. US Patents, Dec. 28, 2006.

[11] Anthony Alan Garrard, David John Locke. Method, Apparatus and Software for Managing a Transactional Message Queue. US Patents, Aug. 14, 2008.

[12] Stephen James Todd. Method, apparatus, and computer program product for processing a queue of messages. US Patents, Apr. 20, 2006.

[13] Douglas W. Clark, Richard Hathaway. Method and apparatus for network-level monitoring of queue-based messaging systems. US

Patents, Oct. 24, 2006.

[14] Adam Sussman, Robert Bennett, Dennis Denker. Apparatus and methods for providing queue messaging over a network. US Patents, Mar. 8, 2007.

[15] Christian Herrmann. Processing messages in a message queueing system. US Patents. Jul. 6, 2006.

[16] Hironobu Kitagawa, Yoshitaka Honishi, Hideki Kawamoto, Takayoshi Shitan. Message queue control program and message queuing system. US Patents, Sep. 6, 2007.

[17] Craig A. Critchley, Richard D. Hill, Krishnan Srinivasan, Uday S. Hedge, Alexander Dadiomov. Failed message error recovery using application specific error queues. Sep. 28, 2006.

[18] Nancy Reiko Ikeda, Ashwinder Singh Ahluwalia, Chao Liang, Krishnan Meiyyappan, Sreenivas Gollapudi, Lakshminarayanan Chldambaran. Methods and systems for efficient queue propagation using a single protocol-based remote procedure call to stream a batch of messages. US Patents, Mar. 16, 2006.

[19] Paul Hopewell, Paul Kettley, Jeffrey M. Nick, Peter Siddall, James H. Warnes. Implementing MQI indexed queue support using coupling facility list structures. US Patents, Jul. 4, 2002.

[20] 丁静 基于Socket和消息队列的中后台接口通讯软件的设计 大连民族学院学报 第3期（总第32期） 2006年5月 P65-P68

[21] W. Richard Stevens, Stephen A. Rago著，尤晋元等译 UNIX环境高级编程 第2版 机械工业出版社 2009年 P418-P422

[22] 《HY-1A项目设计文档》 2004年

致 谢

在我的求学生涯中，有幸能进入北京邮电大学计算机学院攻读计算机科学与技术专业的硕士学位，经过两年半的努力学习，终于完成了论文，回顾这两年半的学习过程让我难以抑制自己内心激动的心情。在这段时间里我收获了很多，周围的人给予了我太多的帮助，对此，我一直心存感激。

首先，我要感谢我的导师卞佳丽教授，卞老师对我的学习和生活都非常关心，在卞老师的帮助和教诲下，我才能圆满地完成学业。卞老师治学严谨，学识渊博，品德高尚，平易近人，在我学习期间不仅传授了做学问的方法，还传授了做人的准则，这些都将使我终生受益。无论是在理论学习阶段，还是在论文的选题、资料查询、开题、研究和撰写的每一个环节，都得到了导师的悉心指导和帮助。在此我要向卞老师表示衷心的感谢！同时，我还要感谢实验室的戴志涛老师和邝坚老师，二位老师无论在论文的开题的时候，还是课题研究的过程中都给我提出了很多中肯的意见和无私的帮助，指导了我整个科研工作的完成。他们认真的科研态度和无私的奉献精神给我留下了非常深刻的印象。在此，感谢他们给予的帮助和支持。

其次，感谢我的同窗李旭光、李贞等，回顾我们在一起度过的研究生美好时光，在项目和学习上，你们给予了我许多的帮助和鼓励，我们建立了深厚的友谊，感谢你们！

感谢S07405硕士班的所有同学，与你们共度的研究生生活让我终生难忘。

感谢所有帮助过我，教育过我以及正在辛勤工作的全体老师，是你们教会了我如何学习，如何做人。

感谢北京邮电大学，我在这里求学、生活，度过了人生中最难忘的两年半研究生生活。

最后，我要感谢我的父母，是你们给予了我生命和奋斗的勇气，我的一切与你们默默地支持是分不开的。你们为我的成长付出了极大的心血，真心地祝福你们、感谢你们。

攻读学位期间发表的学术论文目录

[1] 孙秀云 Linux下聊天工具Ekiga的研究与优化 2009信息技术与应用学术会议 2009年12月