文章编号:1671-4598(2023)07-0214-08 DOI:10.16526/j.cnki.11—4762/tp.2023.07.031 中图分类号:TP311.1 文献标识码:A

国产平台下可靠文件传输软件的设计及实现

虞炳文，龚建泽，丁思炜，王 益，袁化宇
（西昌卫星发射中心，四川 西昌 615000）

摘要：在国产自主可控平台加速推广的当下，为实现文件在测控数据传输网不同软硬件平台之间的可靠、快速传输，文章根据相关技术规范，基于文件交换协议，设计并实现功能完备、运行稳定的可靠文件传输软件；软件具备模块化、可移植、基于TCP协议满足用户可靠性需求、基于UDP协议满足用户传输效率需求、兼容不同软硬件平台等的功能特点，并具备满足安全编程需求的软件特性，立足核心代码设计、验证，对软件功能和稳定运行两个方面做了进一步的改良和提升，以满足跨不同软硬件平台稳定运行、恶劣网络环境中可靠传输、畅通网络环境中尽力快速传输的新要求。

关键词：文件交换协议；跨平台；安全编程；稳定运行；可靠传输；快速传输

Design and Implementation of Reliable File Transmission Software on Domestic Platform

YU Bingwen, GONG Jianze, DING Siwei, WANG Yi, YUAN Huayu
(Xichang Satellite Launch Center, Xichang 615000, China)

Abstract: In the current background of accelerated promotion of domestic autonomous controllable platforms, in order to achieve the reliable and fast file transfer between different software and hardware platforms in the measurement and control data transmission network, according to relevant technical specifications, a reliable file transfer software with complete functions and stable operation based on the file exchange protocol (FXP protocol) is designed and implemented. The software has functional features such as modularity, portability, meeting user reliability requirements based on TCP protocol, meeting user transmission efficiency requirements based on UDP protocol, and compatibility with different software and hardware platforms. It also has software features that meet security programming requirements. Based on the core code design and verification, both software functionality and stable operation are further improved to meet the stable operation across different software and hardware platforms, and to realize the new requirements of reliable transmission in harsh network environments and fast transmission in smooth network environments.

Keywords: FXP Protocol; cross platform; security programming; stable operation; reliable transmission; fast transfer

0 引言

在测控数据传输网中，通常使用基于文件交换协议（file exchange protocol，FXP协议）开发的文件传输软件来实现重要文件的传输。随着国产自主可控平台，以及通信网络的快速发展，对文件传输软件提出了更多新的要求，为保证测控数据在不同软硬件平台下的安全运行和在不同软硬件平台之间文件的快速收发，对所使用的文件传输软件，提出了更高的功能完备及安全可靠运行方面的要求。通过对现实情况的分析，梳理出当前传输网中使用的文件传输软件存在的如下几个问题。

1）通常仅支持在windows平台下使用，代码不支持跨平台编译使用。此处平台指不同型号CPU与不同操作系统的搭配。

2）通常以VC6++甚至更早的工具进行开发，不利于软件代码的维护。

3）不支持在不同平台之间传输文件。

4）软件界面的易用性和美观性较差。

5）软件运行的稳定性和可靠性较差，在软件运行及操作过程中，可能会出现报错而崩溃。

6）当网络环境较差（如丢包率高，延时大）时，传输的文件可能出现不可用的情况。

7）当网络环境较好时，传输速率受到明显的限制的情况。

为解决以上问题，实现国产平台下文件传输的可靠及高效传输，在以下几个方面设计该软件。

1）对FXP基于UDP及TCP协议进一步二次封装，分别实现基于UDP协议的快速传输和基于TCP协议的可靠传输。

2）设计软件具备自适应网络状态，自行决定基于UDP协议进行快速传输，还是基于TCP协议进行可靠传输的功能。

3）基于国产硬件和操作系统进行开发。例如龙芯CPU及银河麒麟操作系统等。

收稿日期:2023-03-25； **修回日期**:2023-03-31。

作者简介:虞炳文(1993-),男,浙江东阳人,大学本科,工程师,主要从事航天测控领域方向的研究。

引用格式:虞炳文,龚建泽,丁思炜,等.国产平台下可靠文件传输软件的设计及实现[J].计算机测量与控制,2023,31(7):214-221.

4）利用 Qt 平台开发一套可实现随处编译的文件传输工具。

5）将软件设计区分为聚焦软件界面操作的软件功能架构设计和聚焦软件后台运行的软件运行设计。

6）关注软件安全编程，聚焦代码的安全性、健壮性，确保软件在运行和功能操作过程中的稳定性和可靠性。即重点阐述从软件功能可靠实现和软件稳定运行的两个方面。

1　文件交换协议介绍

文件交换协议，即 FXP 协议，该协议支持对等的端到端文件交换，基于传输层 TCP 或 UDP 协议（基于 UDP 协议时，需与 RECP 协议配合使用，RECP 即传输质量保障协议），完成应用层数据的组包、解包以及应用层的协议控制[1]。

基于 TCP 协议时，适合在网络环境较差时传输文件，确保文件的完整性，缺点是传输效率较低。基于 UDP 协议时，传输效率较高，适合在网络环境较好时实现尽可能快速的传输文件，缺点是可靠性较低。

基于 TCP 协议的文件交换协议包含四种报文类型，分别为发送请求包、请求应答包、数据包和结束确认包。工作流程为：文件发送方发送发送请求包，接收方响应请求应答包，发送方发送数据包，接收方根据数据包的长度判断发送是否结束，返回确认结束包，结束流程。见图 1。

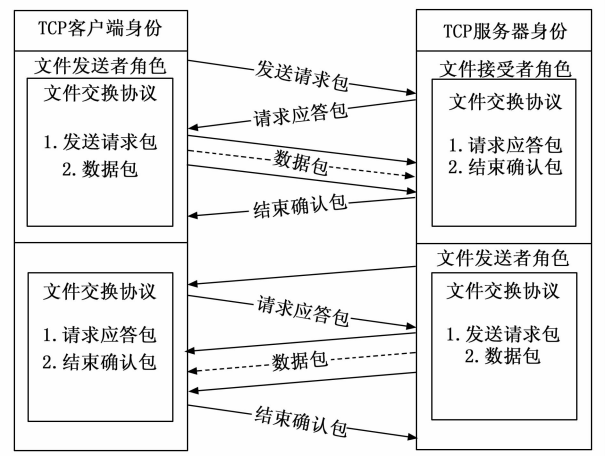

图 1　基于 TCP 文件收发流程图

基于 UDP 协议的文件交换协议包含四种报文类型，连接包、应答包、数据包和结束连接包。工作流程为：文件发送方发送连接包，接收方响应应答包，发送方发送数据包，接收方对每一帧进行响应，发送应答包，发送方发送完毕，发送结束连接包，接收方响应应答包，结束流程。见图 2。

2　软件功能完备性设计

文件传输软件主要需要实现以下五个部分功能。

1）是重新设计软件界面，力求方便美观；

2）是实现对同 FXP 协议软件的无缝衔接；

3）实现基于 UDP 协议在畅通网络环境中的快速尽力传输；

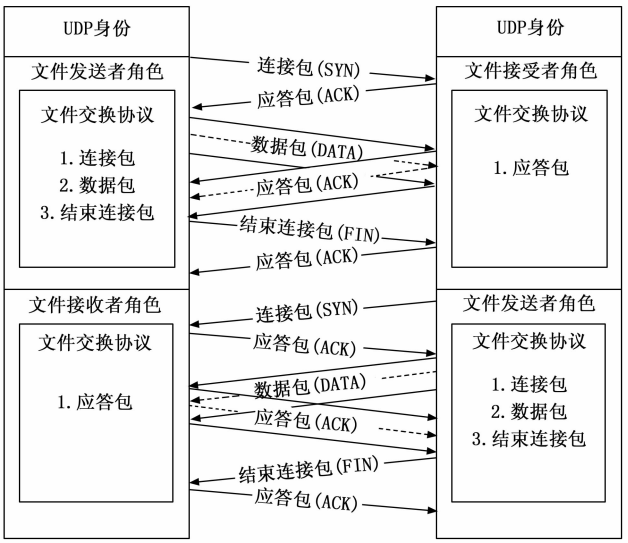

图 2　基于 UDP 文件收发流程图

4）实现基于 TCP 协议在拥塞网络环境中保证文件传输的完整；

5）基于模块化实现上述功能，方便二次调用和扩展。

2.1　软件界面及使用流程设计

设计软件界面时，应当充分考虑用户的操作便捷，尽量减少操作步骤及按钮。

软件使用流程设计[4]见图 3。

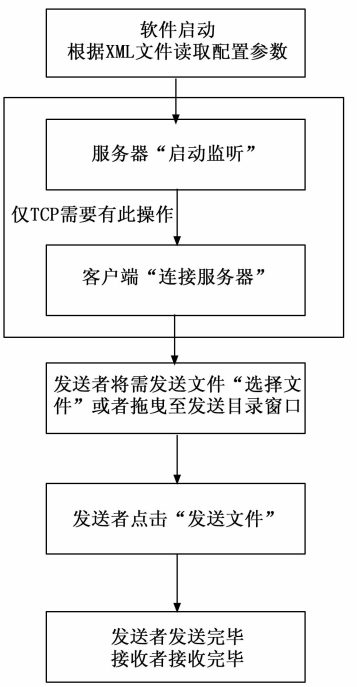

图 3　软件使用流程

2.2　网络编程分层结构

TCP/IP 分层模型是最常见的网络分层模型之一，该模型将网络划分为五个层次，由上到下依次为应用层、传输层、网络层、数据链路层、物理层[2]。“由上到下”的描述，

其含义可理解为数据的发送方向，从应用软件至操作系统至网卡至物理链路（如网线、光纤等线缆）的一个数据传输方向。

TCP 及 UDP 协议属于五层结构中的传输层，而自定义的文件传输协议，便是基于 TCP 及 UDP 传输协议的应用层的网络传输协议。换言之，从封装网络报文的角度分析，文件传输协议的报文，便是 TCP 或者 UDP 协议中的数据域内容。

2.3 软件功能分析及设计

文件传输软件应当具备以下特点。一是可靠传输，即便在较差网络状态下进行文件传输，也能保证文件接收完整；二是高效传输，尽可能多地设计并行传输机制，减少数据包排队等待的时间；三是模块化，封装成函数库，以便在其它软件开发调用时尽量减少代码的重写，提高开发效率；四是部署简单，尽量减少对运行环境的依赖，做到随处部署；五是使用简单，人机交互合理，实现同一功能，尽可能减少鼠标点击次数，显示尽可能多的提示信息，并且通过后台托盘等设计，实现无感化运行；六是部分参数可预配置，避免每次启动软件需要重新设置参数值。

2.3.1 传输可靠性设计

为保证软件可靠性，主要采用了四个方面的设计，一是基于 TCP 网络协议传输机制，二是应答机制，三是队列及超时重传机制，四是自适应网络状态机制。

2.3.1.1 基于 TCP 网络协议传输机制

软件应当具备 TCP 网络传输机制。TCP 协议是面向连接的协议，在 TCP 网络传输中，包含以下设计。

1）区分客户端和服务器端设计。由服务器端启动监听，等待连接，由客户端发起连接。

2）两端等权设计。虽然在网络传输层中区分服务器与客户端，但在应用层文件传输中，无论是客户端还是服务器，都可以发起文件传输和接收。

3）采用功能、角色、身份三层设计。身份包含客户端、服务器两种身份，角色包含文件的发送者和文件的接收者两种，功能包括发送、处理、接收各类型报文等操作。一个身份可以包含有不同的角色，一个角色可以包含有不同的功能，最终一个身份根据其所含角色及角色所含功能，形成一个身份集合，以此实现客户端和服务器的区分及等权。见图 4 所示。

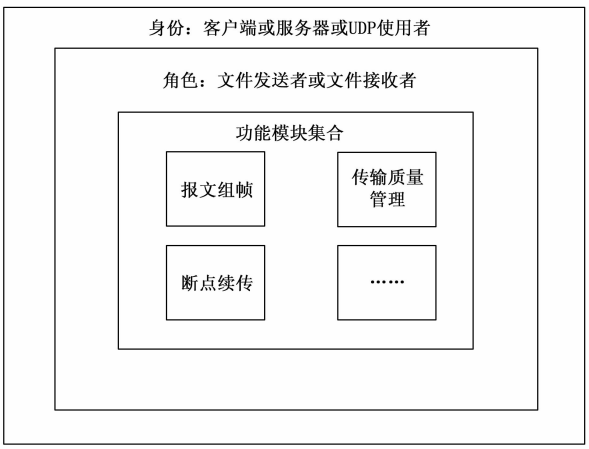

图 4 三层关系示意图

2.3.1.2 应答机制

为保证在文件传输的过程中，不丢包、不重包，并且软件发送或者接收过程中数据处理有序，设计收发流程。收发流程是指在发送者或者接受者角色中，在发送文件或者接收文件的过程中，对每一个步骤进行编号，逐步推进，如在某个步骤内收到非本步骤的报文，则不予处理。采用了以下设计。

1）基于 TCP 协议的发送流程设计值，0 代表新建进程，1 表示发送请求包阶段，2 表示收到请求应答包阶段，3 表示在数据发送阶段，4 表示收到结束确认包。见表 1。

表 1 基于 TCP 协议的发送流程设计

流程序号	所处阶段
0	新建进程
1	发送请求包阶段
2	收到请求应答包阶段
3	数据发送阶段
4	收到结束确认包

2）基于 TCP 协议的接收流程设计值，0 为客户端与服务器建立连接阶段，1 为接收到发送请求包阶段，2 为发送请求应答包阶段，3 为接收数据包阶段，4 为发送结束确认包阶段。见表 2。

表 2 基于 TCP 协议的接收流程设计

流程序号	所处阶段
0	客户端与服务器建立连接阶段
1	接收到发送请求包阶段
2	发送请求应答包阶段
3	接收数据包阶段
4	发送结束确认包阶段

3）基于 UDP 协议的发送流程设计值，0 代表新建进程，1 表示已发送请求包阶段，2 表示收到请求应答包阶段，3 表示发送数据包 ASK 数据阶段，4 表示收到数据包响应包阶段，5 为发送数据包 DATA 数据阶段，6 为已发送结束确认包阶段，7 为收到确认响应包阶段。见表 3。

表 3 基于 UDP 协议的发送流程设计

流程序号	所处阶段
0	新建进程
1	已发送请求包阶段
2	请求应答包阶段
3	发送数据包 ASK 数据阶段
4	收到数据包响应包阶段
5	发送数据包 DATA 数据阶段
6	已发送结束确认包阶段
7	确认响应包阶段

4）UDP 协议的接收流程设计值，0 为无文件传输状态，1 为接收到 SYN 发送请求包阶段，2 为发送 SYN 应答包阶段，3 为接收 DATA 数据包中的发送请求包阶段，4 为响应阶段，5 为接收 DATA 数据包阶段，6 为接收到确认结束包阶段，7 为发送确认结束包阶段。见表 4。

表 4　UDP 协议的接收流程设计

流程序号	所处阶段
0	无文件传输状态
1	接收到 SYN 发送请求包阶段
2	发送 SYN 应答包阶段
3	接收 DATA 数据包中的发送请求包阶段
4	响应阶段
5	接收 DATA 数据包阶段
6	接收到确认结束包阶段
7	发送确认结束包阶段

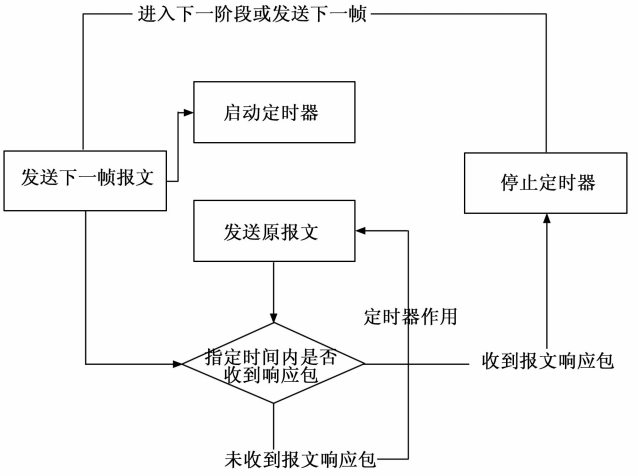

图 5　响应流程图

5）发送者，根据接受者的响应报文，决定是否进入下一发送流程。

6）接受者收到报文后，根据对报文进行报文类型、包序号等内容进行判断，决定是否进入下一流程。

2.3.1.3　队列及超时重传机制

文件发送者读取并传输文件做如下设计。

1）将文件读入缓存区，每次读取指定报文长度，如 4 096 个字节长度的数据，读取之后单独开辟缓存区，在文件数据发送阶段，写入发送报文并发送。

2）在 TCP 协议文件传输阶段中间过程中，无响应报文仅在发送请求和发送结束阶段进行报文响应。但是在文件传输的过程中，每一个数据报文，都携带该部分数据内容在整个文件中所处的偏移位置。

3）在 UDP 协议中，接收者会对发送者的每一个报文进行响应，发送者会根据响应报文信息，决定是否改变所处发送流程的阶段，在此做一些超时设计，如果在指定的时间内未收到响应报文，或者响应报文内容检查（主要为包序号的检查）不通过，则会将此报文挂起，启动定时器，进行该报文的定时重传，而流程也不会进入下一阶段。直到收到指定包序号的响应报文，才会继续发送下一报文。见图 5。

4）在 TCP 协议中，如网络传输中断，导致客户端与服务器重连，并重传文件，发送者发起传输请求报文后，接受者回传的响应会携带文件传输中断位置信息，发送者根据此信息，从文件该位置开始重传此文件。

2.3.1.4　自适应网络状态机制

自适应网络状态机制，主要针对当网络状态出现异常，报文传输在一定程度上受阻时软件的应对机制，该部分内容设计专用于基于 UDP 进行文件传输的过程，做如下设计。

1）记录网络延时、丢包率两个参数指标。基于 UDP 进行文件传输时，根据报文发送时间及接收到相应响应报文的时间差，作为网络延时，根据发包数累计值与收包数累计值之差为丢包数，除以发包累计值，作为丢包率。每次接收报文时，将计算相应的网络延时和丢包率并连同计算时间及接收方地址存入数据库。

2）外推下一时刻的网络延时和丢包率，作为网络状态的判别依据。在此提及的下一时刻，以设定的超时重传界限值作为时间长度。选取最近的 6 次记录数据，根据最小二乘法，进行曲线拟合，可选用二阶、三阶、四阶等进行拟合。

3）计算时延抖动。选取最近的 6 次网络延时值，计算方差，作为时延抖动。

4）根据网络延时、丢包率、时延抖动三个指标综合判断[3]网络状态。将网络状态分为正常、干扰、阻塞、稳定四个状态。正常状态时，将超时重传等待时间值减去 10 毫秒，但必须大于 0 值，干扰状态时，将超时重传等待时间值加上 10 毫秒，阻塞状态时，将超时重传等待时间增加一倍，稳定状态时，不作操作。

2.3.2　效率设计

为保证软件的文件传输效率，采用了三个方面的设计，一是自适应网络状态机制，二是数据处理多线程机制，三是数据驱动处理数据报文机制。

1）自适应网络状态机制。滑动窗口[5]设计，自适应网络状态机制在可靠性设计中已陈述主要内容，在效率设计方面的考虑，即当网络状态恢复时，会适当减少超时重传时间，以加快网络报文传输效率。

2）数据处理多线程机制。多线程机制主要针对两种情况做处理，一是在 TCP 服务器端设计可同时接收多个客户端的连接，接收处理不同数据发送者发来的数据，二是将数据处理线程独立出来，加快效率。

3）数据驱动处理数据报文机制。在多线程的基础上，定义一个全局的数据容器。当数据接受者从主线程接收到数据后，将数据放入数据容器，在数据处理线程利用 while 函数无限循环，不断判断数据容器是否为空，非空则处理

数据，处理完则将该部分数据扔出容器。

2.3.3 模块化设计

在软件中，根据功能，角色，身份三层管理的设计，将三个方面的内容逐层实现模块化，在别处可根据需求通过接口调用即可方便使用。

2.3.3.1 发送者和接受者角色模块化

根据发送者和接受者的角色区分，以及 TCP 和 UDP 的功能区分，可以设计为 TCP 发送者类、TCP 接受者类、UDP 发送者类、UDP 接受者类四个模块，实现数据报文的发送、处理、接收功能。

2.3.3.2 客户端和服务器的身份模块化

根据客户端和服务器的身份区分，并根据发送者和接受者角色区分，可以分为客户端类、服务器类、UDP 收发类三个模块，分别实现客户端、服务器及 UDP 收发的功能。

2.3.3.3 重写界面显示类

为实时显示文件发送和接收进度，实现文件拖曳上传等辅助性功能，重写一个继承自 QTableWidget 的类，作为显示模块，在调用时，只需要将控件提升为自定义类即可。

2.3.3.4 集成工具类

为方便调用，设计一系列自定义工具类，实现数据库相关操作的数据库类、实现自适应网络状态的网络质量保障控制类以及实现读取 XML 预配置参数的读取类。

2.3.4 兼容性设计

为实现软件部署方便，提高可移植性，需要尽量减少对环境因素的依赖，做以下三方面设计。

1）部署简单。最后发布的软件版本应该将其依赖库统一发布。

2）轻量级数据库。为避免对环境数据库的依赖，不采用 mysql 等需要安装部署的数据库，采用 Qsqlite 数据库。

3）区分系统加入不同的库。在调用系统自带的库实现一些功能时，可能会出现适配的问题，因此需要采用 # if defined（Q _ OS _ WIN32），# else，# endif 语句，对库的调用加上限制条件。

2.3.5 易用性设计

为实现交互界面的优化，有以下三方面设计。

1）精简界面按钮。尽量减少界面中的点击按钮，仅保留选择文件、发送、启动监听、连接服务器四种类型按钮。

2）设置预配置项。将部分不常修改的配置内容，作为预配置项，通过修改 xml 文件进行修改，如本地监听端口，报文长度等配置。

3）无感化运行。可将软件最小化至托盘，并在后台运行，接收文件。

3 软件运行安全稳定性设计

为保证软件稳定可靠运行，避免因代码编写或逻辑上错误导致软件崩溃，对软件安全编程内容做一些设计，以进一步提升软件运行可靠。

3.1 安全编程概念

应用软件安全编程，是从提升软件安全性的角度，依照《GB-T 38674-2020 信息安全技术 应用软件安全编程指南》等应用软件安全编程规范性引用文件，针对应用软件编程过程进行规范，实现软件开发全过程的指导，以达到有效降低软件安全风险的目的。

3.2 安全编程设计

3.2.1 报文组帧拆帧设计

在软件中，涉及到大量的报文的组帧和拆帧，在此类操作中，应当先将所有报文根据其报文类型及报文中不同字段的数据类型和含义，先设计成一个结构体，当处理到报文的组帧或者拆帧时，只需将指定长度的数据赋值给同样长度的结构体即可。结构体主要可分为两种类型，一是定长结构体，二是不定长结构体。在软件中，传输文件数据的报文设计为不定长结构体，因其最后一帧不一定为完整的一帧，除此之外的报文，都是定长结构体。

3.2.2 报文赋值操作设计

在软件中，涉及到大量的数据内容赋值，主要涉及到将接收到的数据赋值给相应结构体，将结构体中的数据赋值给报文以发送两个操作，在此过程中，如涉及到指针的赋值，避免使用 strcpy（）等不能指定赋值长度的函数，因为此类函数通常认为“\0”是停止符，会停止赋值。可使用 memcpy（）代替。

3.2.3 数据写入文件设计

接受者收到数据流写入新建文档时，通常使用 QTextStream 函数库的 out（）函数或使用 QFile 的 write（）函数，前者遇见数据中有“\0”时，认为是停止符，会停止写入，故采用 QFile 的 write（）函数写入文件。另外，利用 QFile 写入文件时，有多重模式可选，TCP 传输中采用在文件中末尾追加数据的 QIODevice：：Append 模式，UDP 传输中采用覆写文件中原数据的 QIODevice：：ReadWrite 模式，因为 TCP 传输中需要断点续传，而 UDP 传输中是中断重传。

3.2.4 多线程设计

在多线程中，为了保证线程的安全，需要注意两点。

1）退出线程采用 exit（）或者 quit（）函数，避免使用 wait（）函数。

2）在多线程中，有两种实现方式，一种是新建一个继承自 QThread 的类，将需要在新的线程中执行的程序，写入 run（）函数，在 run（）函数之外，使用 start（）和 stop（）启停。另一种方法是新建一个类，在调用的类中实例化后，利用 moveToThread（）函数将该类放入一个实例化的 QThread 中。因为采用 while 的死循环机制，使用第二种方法容易出错，在设计中通常采用第一种方式实现多线程。

3）在多线程中，注意当使用到 while 的死循环时，务必要添加标志位，用于及时退出 while 循环，对安全结束线程有重要意义。

3.2.5 数据收发设计

网路数据收发使用的 QUdpSocket 和 QTcpSocket 套接字，在实现数据收发时，数据接收可以跨线程，但是数据

发送不能跨线程使用，即在哪个线程中实例化创建的套接字，则只能在该线程中实现数据发送。因此，需要设计信号槽，进行数据中转，将发送数据发射至实例化的线程中。

特别注意，在将接收线程和处理线程分开设计时，为使用数据驱动方式加快处理效率，应当将接收数据存放的容器和处理线程待处理的数据存放的容器分开，例如区分为 2 个 Qvector，以避免在容器中进行取值、移除等操作时，出现位置偏移导致的软件崩溃。

另外，在数据接收时，可能会存在数据接收报文不完整的情况。即便发送方按照每一帧大小为指定长度的报文在发送，如一帧 4 096 个字节，在接收方也不一定会按照 4 096个字节的长度接收每一帧。

有时候当发送方的发送速率过快，网络传输时延非常低时，对于接收方，会在缓存区中同时存储多帧数据，然后在一次 readyRead（）触发下，会一起被例如 readAll（）读到。

但是在 TCP 协议的网络缓存区中，仅仅能缓存 65 535 字节的数据，当超过这个字节数，多余的报文就会进不到缓存区，也不会丢弃，而是开始排队。

例如在很短的时间内收到 16 帧的 4 096 字节的报文时，总字节就会超过 65 535，此时最后一帧就会不完整，注意，最后一帧不一定是将 65 535 填满后再剩下，最后一帧会在某个位置被斩断。因此，需要妥善处理可能出现的数据接收不完整的现象。

3.2.6　定时器设计

在实现数据超时重发功能时会用到定时器，定时器不能跨线程启停的限制，因此，当需要在定义全局的定时器时，在线程中启停定时器，需要设计信号槽实现。

3.2.7　信号槽设计

利用 connect 关联一个信号槽后，当涉及到重建操作时，须记得用 disconnect 取消关联，不然可能会出现多次创建的情况，即一个信号被响应多次，槽函数被执行多次。也需要警惕过分考虑 disconnect 的情况，因为在很多实例化的过程中创建的信号槽，在对象被 delete 后，会自动释放，要避免重复释放。

3.2.8　结构体设计

结构体所占内存大小，可能会大于实际定义的大小，这是因为操作系统中有字节对齐机制。在软件设计中，涉及到大量的结构体与报文之间的相互赋值，必须确保结构体所占内存大小等于实际大小，因此需在定义结构体的头文件中，使用＃pragma pack（1）函数设置对齐长度为 1 字节，可避免大小不一致的问题。

4　软件测试

软件测试内容主要聚焦于该软件的可靠性及性能测试。以传输质量，软件稳定运行情况，异常处置情况，软件适配性等作为记录指标，分析软件的可靠性。

4.1　兼容性测试

测试新设计的文件传输软件是否兼容之前的 FXP 文件传输协议软件。测试通过。

表 5　兼容性测试

测试内容	兼容性测试	
测试方案	与现有基于 FXP 的文件传输软件进行文件传输	
记录内容	文件传输是否正常	
期待结果	现有软件及新研软件文件收发均正常	
测试结果	是否正常接收	正常
	文件是否完整	完整

4.2　传输质量测试

测试对不同网络环境的适应情况。传输质量包含传输速率，传输过程中的丢包率，传输文件的完整性等作为指标。TCP 及 UDP 均采用此测试内容。测试相关内容见表 6。

表 6　传输质量测试记录表

测试内容	传输质量测试	
测试方案	设计传输 3 个大型文件压缩包，每个压缩包体积大概在 3～5 个 G，总体积控制在 10 个 G 左右。第一个压缩包内存储数据文档为主，第二个压缩包存储视频为主，第三个压缩包存储应用软件安装程序为主	
记录内容	传输最大速率，传输使用时间，丢包率	
期待结果	1. 文件传输完毕 2. 文件解压成功，文档能正常打开，内容完整，视频能正常播放，并完整播放，应用程序能正常安装	
测试结果	是否传输完毕	传输完毕
	文件是否正常	解压成功，内容完整
	传输最大速率	TCP：11.2 Mbps，UDP：44.1 Mbps
	传输平均速率	TCP：10.04 Mbps，UDP：42.8 Mbps
	丢包率	无丢包

4.3　软件运行测试

软件运行测试主要测试软件的稳定运行情况，测试内容及结果见表 7。

表 7　软件运行测试

测试内容	软件运行测试	
测试方案	软件连续挂机 12 小时，每隔 1 小时，进行一次传输质量测试	
记录内容	无	
期待结果	软件运行正常	
测试结果	第 1 小时	传输结果正常，软件运行正常
	第 2 小时	传输结果正常，软件运行正常
	第 3 小时	传输结果正常，软件运行正常
	第 4 小时	传输结果正常，软件运行正常
	第 5 小时	传输结果正常，软件运行正常
	第 6 小时	传输结果正常，软件运行正常
	第 7 小时	传输结果正常，软件运行正常
	第 8 小时	传输结果正常，软件运行正常
	第 9 小时	传输结果正常，软件运行正常
	第 10 小时	传输结果正常，软件运行正常
	第 11 小时	传输结果正常，软件运行正常
	第 12 小时	传输结果正常，软件运行正常

4.4 软件适配性测试

适配性测试，即测试该软件代码在不同的系统平台下的能否正常编译、运行，功能是否正常，测试内容及结果见表 8。

表 8 软件适配性测试

测试内容	软件适配性测试	
测试方案	在 3A4000 与银河麒麟平台上编译运行。 在 3A5000 与 loongnix 平台上编译运行。 在 FT1500 与银河麒麟平台上编译运行。 在 3A4000 与 UOS 平台上编译运行。 在 FT1500 与中标麒麟平台上编译运行。 在 x86 与 UOS 平台上编译运行。 在 x86 与 Windows10 平台上编译运行。 在 x86 与 Windows7 平台上编译运行。	
记录内容	文件传输是否正常	
期待结果	编译且使用正常	
测试结果	3A4000 与银河麒麟	编译成功并运行正常
	3A5000 与 loongnix	编译成功并运行正常
	FT1500 与银河麒麟	编译成功并运行正常
	3A4000 与 UOS	编译成功并运行正常
	FT1500 与中标麒麟	编译成功并运行正常
	x86 与 UOS	编译成功并运行正常
	x86 与 Windows10	编译成功并运行正常
	x86 与 Windows7	编译成功并运行正常

4.5 性能测试

与 FTP 协议比较：

FTP 协议基于 C/S 模式，服务器和客户端不对等，传输只能由客户端发起，客户端可以选择使用下载服务（客户端将从文件从服务器下载到客户端）或者上传服务（客户端将文件上传到服务器中）。如果两个传输节点间单向部署 FTP 服务器，则客户端无法及时了解服务器端文件状态，需要明确服务器端如何及时有效与客户端进行各种控制信息的交互。为了解决该问题，一个解决办法就是每个节点都部署 FTP 服务器，这样每个节点既是服务器，同时又是客户端，势必增加部署难度及复杂度，加大了端口管控的安全风险。

换言之，FTP 协议需要严格区分客户端与服务器在使用时。如果要实现两点之间的文件传输，则需要在某一侧搭建 FTP 的服务器，另一侧以客户端的形式登陆，而后传输文件，或者将某个文件约定在某个 FTP 服务器中进行周转，文件发送方将文件上传至该服务器，文件接收方再去服务器获取。

但是利用本文的文件交换协议，在基于 UDT 实现时，无需考虑客户端及服务器的区别，即便是在基于 TCP 协议实现时候，其服务器端的操作也相对较为简单。是基于端到端的对等传输及主动推送需求开发的开放式协议，每一段都既可以作为传输的发起者，又可以作为传输的接收者，而且支持发送和接收并行操作。

表 9 与 FTP 协议比较

测试内容	与 FTP 协议比较	
测试方案	利用 FTP 协议和 FXP 协议传输相同大小的文件，大小为 1 G	
记录内容	文件是否完整，以及传输所用时间	
期待结果	文件均完整，且 FXP 协议快于 FTP 协议	
测试结果	FTP 传输时间	15 分钟
	基于 UDP 传输时间	3.2 分钟
	基于 TCP 传输时间	13.6 分钟
	文件是否完整	完整

4.6 异常处置测试

异常处置测试，主要测试软件使用时，针对一些特殊操作的处理能力，测试内容及结果见表 10。

表 10 异常处置情况

测试内容	异常处置情况	
测试方案	多个客户端连接服务器，同时发送文件。 断开服务器与客户端的连接，重新连接。 在传输过程中，中断传输，并重传。	
记录内容	文件传输是否正常	
期待结果	多个客户端能同时发送文件，服务器同时接收，文件完整。 客户端与服务器之间的连接恢复后能再进行文件传输。 文件传输中断后，TCP 协议能断点续写文件，UDP 能覆写文件。	
测试结果	是否正常接收	正常
	文件是否完整	完整

5 应用场景

软件开发完成后，可根据部署的设备系统环境不同，分别编译生成银河麒麟、中标麒麟等平台下的可执行程序，运用于测控数据传输网进行文件可靠传输，并且可兼容 Windows 平台。

在测控设备执行完任务后，其事后数据可利用本软件进行上传，网络质量较好时，可使用文件传输协议的 UDP 传输功能进行快速传输，网络较为拥堵时，可利用文件传输协议的 TCP 传输功能进行稳妥可靠的传输。

6 结束语

根据相关技术规范，基于文件交换协议（FXP 协议）设计该数据传输软件，该软件可根据其 TCP 特性进行可靠传输，根据其 UDP 特性进行快速传输。在软件设计及编程过程中，对软件的数据传输功能的功能性设计以及软件运行的安全稳定性进行了充分考虑。此软件基于 Qt 设计开发，可实现一处编写，随便编译，跨操作系统平台可用，对于国产平台具备良好的适应性，并保证了代码的可读性和可维护性。此软件的设计，满足当前测控数据网中对文件传输软件在跨不同软硬件平台下稳定运行、在恶劣网络环境中可靠传输、在畅通网络环境中尽力快速传输的新要求。

参考文献：

[1] 尹然然，等. 基于 UDP 协议的可靠性改进协议 [J]. 电脑知识与技术，2010 (16)：4379－4380.

[2] 朱先飞，等. UDT 协议及其拥塞控制机制 [J]. 广东通信技术，2011 (10)：49－52.

[3] 王艳芳，等. 基于 UDP 的数据可靠传输技术研究与应用 [J]. 计算机工程与应用，2010 (3)：105－108.

[4] 赵　飞，等. UDP 协议与 TCP 协议的对比分析与可靠性改进 [J]. 计算机技术与发展，2006 (9)：219－221.

[5] 刑　璐，等. 高速网络环境中适合大数据传输的改进 UDT 协议 [J]. 计算机应用与软件，2018，35 (6)：138－145.

[6] 黄玉昌，等. 基于 FTP 协议的企业级文件传输系统设计 [J]. 中国新通信，2018 (6)：134－135.

[7] 夏伟东，等. 基于 UDP 的可靠传输协议丢包测量方法研究 [D]. 南京：东南大学，2021.

[8] 袁乐平，等. 基于自有传输协议的可靠文件传输模块设计 [J]. 科学技术创新，2019 (2)：91－92.

[9] 李　勇，等. 基于麒麟系统的即时通讯系统设计与实现 [J]. 自动化技术与应用，2020，39 (3)：55－59.

[10] 张杨阳，等. 航电通信网络中简单文件传输协议的实现研究 [J]. 信息通信，2019 (1)：82－83.

[11] 王　洋，等. 空间通信自适应文件传输协议设计 [J]. 中国空间科学技术，2017，37 (3)：53－61.

[12] 吴义三，等. 文件传输协议下网络数据的传输 [J]. 天津中德职业技术学院学报，2014 (1)：100－101.

[13] 张　恺，等. 基于 UDP 的可靠文件传输协议的设计与实现 [D]. 西安：西安电子科技大学，2014.

[14] 甘清云，等. 浅谈应用软件安全编程 [J]. 网络安全技术与应用，2021 (1)：37－38.

[15] 张晓丽，等. Java 多线程编程中数据安全的应用研究 [J]. 计算机光盘软件与应用，2014，17 (16)：159－160.

[16] 张俊峰，等. 计算机网络安全的软件编程与系统运维分析 [J]. 软件，2022，43 (7)：180－182.

[17] 刘　渊，等. 基于对象状态的软件测试方法分析 [J]. 互联网周刊，2022 (21)：44－46.

[18] 史传倩，等. 计算机软件测试技术与深度开发应用研究 [J]. 信息与电脑，2022，34 (14)：44－46.

[19] 丁　娴，等. 国内软件质量与测试标准化现状研究 [J]. 中国标准化，2022 (2)：26－31.

[20] 张雅东，等. 软件工程中软件有关测试技术的思考 [J]. 石河子科技，2022 (1)：17－19.

(上接第 162 页)

[6] YUE YANG, HUANG HAO, AHMED N, et al. Reconfigurable switching of orbital-angular-momentum-based free-space data channels [J]. Optics Letters, 2013, 38 (23): 5118－5121.

[7] YAN Y, XIE G, LAVERY M P J, et al. High-capacity millimetre-wave communications with orbital angular momentum multiplexing [J]. Nature communications, 2014, 5 (1): 1－9.

[8] REN Y X, LI L, XIE G D, et al. Line-of-sight millimeter-wave communications using orbital angular momentum multiplexing combined with conventional spatial multiplexing [J]. IEEE Transactions on Wireless Communications, 2017, 16 (5): 3151－3161. doi: 10.1109/TWC.2017.2675885.

[9] WILLNER A E, LIU CONG. Perspective on using multiple orbital-angular-momentum beams for enhanced capacity in free-space optical communication links [J]. Nanophotonics, 2020, 10 (1): 225－233.

[10] THIDé B, TAMBURINI F, THEN H, et al. Angular momentum radio [C] // Proceedings of SPIE 8999, Complex Light and Optical Forces VIII, San Francisco, USA, 2014. doi: 10.1117/12.2041797.

[11] ALLEN L. Orbital angular momentum: A personal memoir [J]. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2017, 375 (2087): 20160280. doi: 10.1098/rsta.2016.0280

[12] PADGETT M J. Orbital angular momentum 25 years on [Invited] [J]. Optics Express, 2017, 25 (10): 11265－11274.

[13] FRANKE-ARNOLD S, ALLEN L, and PADGETT M. Advances in optical angular momentum [J]. Laser & Photonics Reviews, 2008, 2 (4): 299－313.

[14] THIDé B, THEN H, SJÖHOLM J, et al. Utilization of photon orbital angular momentum in the low-frequency radio domain [J]. Physical Review Letters, 2007, 99 (8): 087701.

[15] MOHAMMADI S M, DALDORFF L K S, BERGMAN J E S, et al. Orbital angular momentum in radio—A system study [J]. IEEE Transactions on Antennas and Propagation, 2010, 58 (2): 565－572.

[16] TRICHILI A, PARK K H, ZGHAL M, et al. Communicating using spatial mode multiplexing: Potentials, challenges, and perspectives [J]. IEEE Communications Surveys & Tutorials, 2019, 21 (4): 3175－3203.

[17] 郭忠义，汪彦哲，郑　群，等. 涡旋电磁波天线技术研究进展 [J]. 雷达学报，2019，8 (5)：631－655.

[18] GUO Z Y, WANG Y Z, ZHENG Q, et al. Advances of research on antenna technology of vortex electromagnetic waves [J]. Journal of Radars, 2019, 8 (5): 631－655.

[19] JACKSON J D. Classical Electrodynamics [M]. New York: John Wiley & Sons, 1999.

[20] SHU J Y, DENG L, LI S F, et al. Use OFDM in OAM communication to redcuce multi-path effects [C] // The 3rd International Conference on Electronic Information and Communication Technology, Shenzhen, China, 2020: 54－56.

[21] LIANG L P, CHENG W C, ZHANG W, et al. Joint OAM multiplexing and OFDM in sparse multipath environments [J]. IEEE Transactions on Vehicular Technology, 2020, 69 (4): 3864－3878.

[22] LIANG J, JING Z L, FENG Q, et al. Synthesis and measurement of a circular-polarized deflection OAM vortex beam with sidelobe suppression array [J]. IEEE Access, 2020, 8: 89143－89151. doi: 10.1109/ACCESS.2020.2993877.