**武汉大学计算机学院**

**本科生课程实验报告**

**Fishield基于C++的网络云盘程序**

专 业 名 称 ：软件工程

课 程 名 称 ：网络及分布式计算

指 导 教 师 一：胡继承 教授

学 生 学 号 ：201302580055

学 生 姓 名 ：了然

二○一九年五月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名： 了然 日期： 2019.05.31

摘 要

本次试验的主要目的是利用本学期课堂上所学的关于Web网络编程相关知识，在实践中运用。实现一个完整的网络云盘程序，包含完整的服务器端编程以及相应的客户端编程。

程序将包含一个常见网络云盘所应具备的所有基本功能，包含上传文件、下载文件等。同时对用户设置优先级，实现更细粒度的权限管理。并且，为了给予用户更好的体验，我实现了更加细节的功能，包括删除文件、重命名文件、暂停上传/下载任务、继续上传/下载任务等、取消上传/下载任务等。允许用户将暂停任务并在之后继续任务极大的扩展了该软件的适用范围，使之可以应用于较大的文件，允许通过断点续传的方式将一个较大的文件分块上传。

实现方面，我利用由Google开发的开源工具Protobuf。Protobuf是一款跨平台、跨语言、可扩展的，数据结构序列化工具。我定义了本软件的服务器端与客户端信息交互的协议，本质上这是一个我自己定义的应用层网络协议。随后，我通过Protobuf将之简洁快速的序列化，并用TCP协议在服务器端与客户端间传递信息。在服务器端，有一个线程作为守护线程不断地运行我自己设计的调度器算法，等待客户端发来请求信息并作出相应反馈。

**关键词：**C++；网络云盘；服务器端编程；客户端编程；Socket编程；Protobuf；网络应用层协议

**目 录**

**1** **实验目的和意义**

1.1 实验目的…………………………………………………………………………………4

1.2 实验意义…………………………………………………………………………………4

**2** **算法原理**

2.1概述…………………………………………………………………………………………5

2.2暴力枚举算法……………………………………………………………………………5

2.3 SIFT算法…………………………………………………………………………………7

**3** 图像拼合

3.1概述…………………………………………………………………………………………8

3.2变换矩阵…………………………………………………………………………………8

3.3坐标变换…………………………………………………………………………………8

3.4优化方案…………………………………………………………………………………9

**4** 图像修饰

4.1概述…………………………………………………………………………………………9

4.2图像剪裁…………………………………………………………………………………10

4.3图像缩放…………………………………………………………………………………10

4.4图像剪裁…………………………………………………………………………………11

**5** 成果展示

5.1概述………………………………………………………………………………………12

5.2支持多种重叠……………………………………………………………………………12

5.3支持广义重叠图像……………………………………………………………………13

**参考文献 ……………………………………………………………………………………** 14

**1 实验目的和意义**

**1.1** **实验目的**

本次试验的主要目的是利用本学期课堂上所学的关于Web网络编程相关知识，在实践中运用Socket编程。实现一个完整的网络云盘程序，包含完整的服务器端程序以及相应的客户端程序。

**1.2** **实验意义**

学生通过亲自设计专属的应用层网络协议，深入理解了网络分层结构的服务模型。同时通过切实的Socket编程联系，深入的掌握了TCP/UDP的常见特性以及常见使用方式。与此同时，通过服务器端的调度器算法以及数据库的相关代码，学生较好的讲所学网络方向知识与操作系统、数据框方向相关知识有机的结合在了一起，并通过编写代码极大地加深了理解。Protobuf的使用则使学生与世界领先的开源社区产生了联系，了解到了State-of-the-art、cutting-edges的现状。

**2 需求分析**

**2.1** 上传文件

用户可以通过客户端程序将本地文件上传、存储到云端服务器。

**2.2** 下载文件

用户可以通过客户端程序预览到已经保存在服务器端的所有文件，并选择相应文件，将其从云端服务器下载到本地。

**2.3** 暂停上传/下载任务

当一个上传/下载任务被创建之后，用户可以选择暂停该任务并安全的退出程序。

**2.4** 暂停上传/下载任务

当一个上传/下载任务被暂停之后，用户可以选择重启该任务，继续上传/下载的过程。

**2.5** 取消上传/下载任务

当一个上传/下载任务被创建之后，在该任务完成之前，用户可以选择取消该任务，永久性的终止该上传/下载的过程，并清除一切产生的中间文件。

**2.6** 删除文件

用户可以通过客户端程序预览到已经保存在服务器端的所有文件，并选择删除其中的部分文件。

**2.7** 重命名文件

用户可以通过客户端程序预览到已经保存在服务器端的所有文件，并选择修改其中部分文件的名字。修改文件存储路径也被看做是重命名文件的一种特例情况。

**2.8** 创建文件夹

用户可以通过客户端程序预览到已经保存在服务器端的所有文件，并可以选择在适当位置创建新的文件夹。

**2.8** 权限控制

我将上传与下载的权限分离，允许用户只拥有上传文件的权限，允许用户只拥有下载文件的权限，允许用户拥有上传/下载的完整权限。

**2.8** 客户端IP限制

更加细致的安全控制是可行的，服务器端可以检查客户端的IP地址，可以通过适当配置，只允许部分合法IP地址访问（白名单机制），或者禁止部分非法IP地址访问（黑名单机制）。

**3 应用层协议设计**

**3.1** 概述

在Fishield项目中，客户端生成恰当的Fishield\_Request，序列化之后发送给服务器端，服务器端在接收到请求后首先进行反序列化操作，还原成原本的Fishield\_Request数据结构。然后进行相应处理，并将处理结果写入一个新创建的Fishield\_Response数据结构中，经序列化后发送给客户端作为相应。

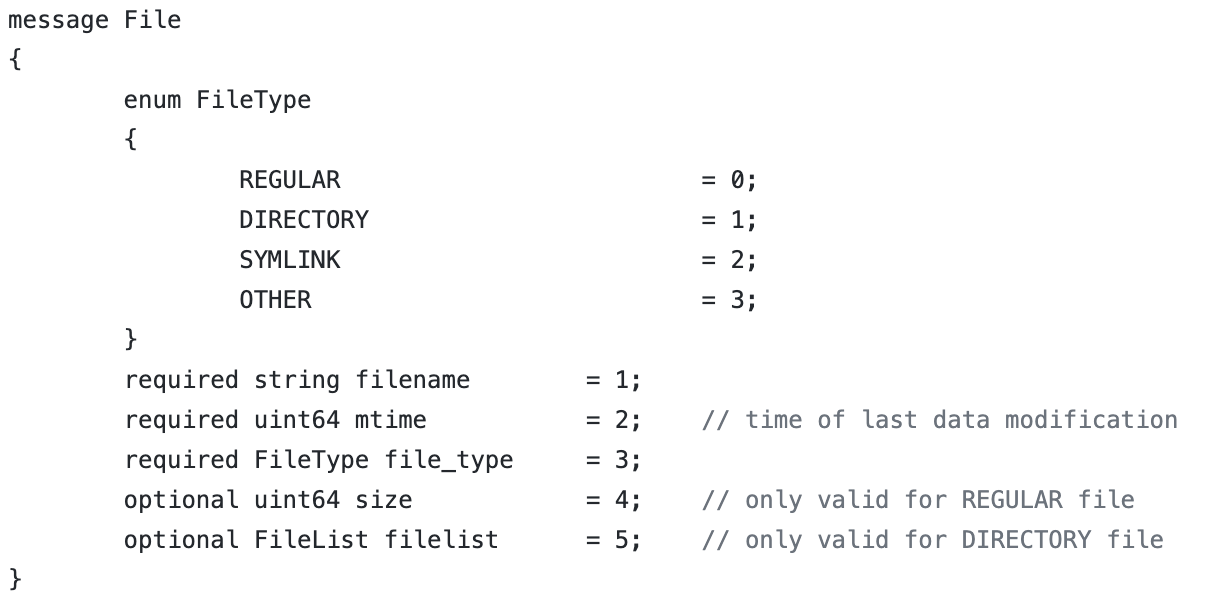
这里提到的Fishield\_Request和Fishield\_Response就是Fishield项目的应用层协议，它们是我定义好的两种数据结构，其中包含了大量的控制信息。它们的内在结构通过Protobuf的proto2语言可以及其简单的定义出来。然后Protobuf可以通过proto2语言所定义的结构生成相应的C++类，客户端/服务器端代码调用该C++类。Protobuf在其中起到了一个间接层的作用，其优势是简洁。用proto2语言定义结构比用C++语言直接定义要简单得多，并且Protobuf会自动帮我们生成相应的序列化/反序列化方法，可以极大地简化随后的C++代码。

**3.2** 文件数据结构

文件数据结构包含文件名、文件最后修改时间、文件类型、文件大小、子文件列表五个字段。

与Unix系统定义保持一致，文件有普通文件、文件夹、符号链接和其他文件四种类型。文件大小只有在文件类型是普通文件时有意义，子文件列表只有在文件类型是文件夹时有意义。

图3.1 文件数据结构定义



**3.3** Fishield\_Request数据结构

Request数据结构的核心是RequestType，即请求类型。其他字段均为可选的补充参数，都是只有针对某个特殊的请求类型，某几个字段才是有效的。

* 登录请求（LOGIN）

在该类型下，用户名（username）和密码（password）字段是有效的。客户端程序将用户输入的用户名信息封装到Fishield\_Request数据结构中，并发送给服务器。服务器查询数据库相关数据，进行校验并返回校验结果。

* 请求文件列表（FILELIST）

在该类型下，服务器路径（remote\_path）字段有效。经校验路径有效后，服务器端程序将服务器路径所代表的位置下所有子文件列表返还给客户端。该请求主要用于向用户展示云端服务器所存储的内容列表。

* 创建文件夹（MKDIR）

在该类型下，服务器路径（remote\_path）字段有效。经校验路径有效后，服务器端程序在服务器路径所指定的位置创建一个空的文件夹。

* 上传文件请求（UPLOAD）

在该类型下，一个新的上传任务会被创建。此时文件名（filename）和包裹数量（packet\_no）字段有效。为了实现有效的断点续传功能，让用户可以选择将任务暂停、在一段时间后重启。我将每个文件分成若干定长的包裹（Packet），并逐一发送。这样通过跟踪发送的包裹编号、已发送的包裹数量以及包裹总数，就可以在应用层实现可暂停的可靠传输。在用户点击暂停某个任务的时候，客户端程序会在发送完当前正在发送的包裹后停止发送。

* 下载文件请求（DOWNLOAD）

在该类型下，一个新的下载任务会被创建。此时服务器路径（remote\_path）和文件名（filename）字段有效。此后下载任务开展的时候，文件同样会被切分成一个个定长的包裹（Packet）逐一从服务器端发送。

* 发送包裹（SEND\_PACKET）

在该类型下，客户端向服务器端发送一个上传任务的包裹（Packet）。此时，任务编号（Task\_id）和包裹（Packet）字段有效。任务编号是用于区分多个不同的任务的，每个上传/下载任务都有一个唯一的任务编号。包裹（Packet）则包含实际发送的数据。

* 接收包裹（RECEIVE\_PACKET）

在该类型下，客户端通知服务器端重启一个已经暂停的下载任务，此时任务编号（Task\_id）有效。任务编号用于指定需要重启的任务。

* 删除文件（REMOVE）

该类型用于删除云端服务器上存储的某个文件。此时服务器路径（remote\_path）和文件名（filename）字段有效。

* 重命名文件（RENAME）

该类型用于将云端服务器上的某个文件重命名，这里的重命名是广泛意义上的重命名，包含文件的位置移动。此时服务器路径（remote\_path）、文件名（filename）和新文件路径（new\_path）字段有效。新文件路径用于指定修改后的文件路径。

* 取消任务（CANCEL）

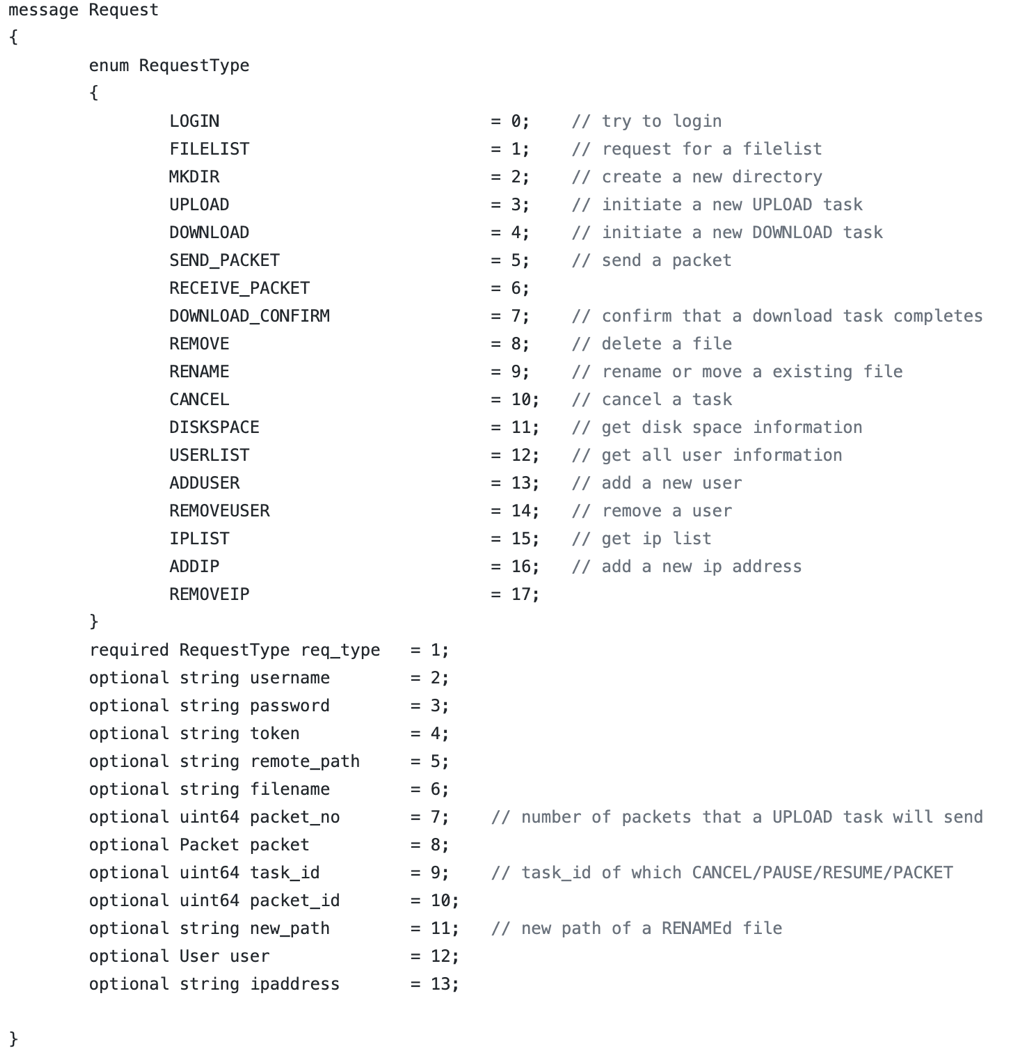
该类型用于取消某个现存的、未完成的上传/下载任务。此时任务编号（Task\_id）有效，用于指定需要取消的任务。

图3.2 Fishield\_Request结构定义

* 获取磁盘容量信息（DISKSPACE）

该类型用于获取云端服务器的存储空间信息，主要包括磁盘总容量、已使用容量以及未使用容量大小。

**3.3** Fishield\_Response数据结构

Response数据结构的核心是ResponseType，即响应类型。其他字段均为可选的补充参数，都是只有针对某个特殊的响应类型，某几个字段才是有效的。

* 成功（SUCCESS）

该类型代表请求合法且处理成功。是最常见最普通的类型。

* 用户不存在（NOSUCHUSER）

使用不存在的用户名登录会出发此错误。

* 非法路径（ILLEGALPASSWD）

下载文件/新建文件夹/重命名文件/删除文件时如果给出的路径非法会出发此错误。

* 未知错误（UNKNOWN）
* 无响应错误（NORESPONSE）

无法链接到服务器或链接超时会触发此错误。

* 非法请求（ILLEGALREQUEST）

出现未定义的请求会出发此错误

* 无权限错误（NOPRIVILEGE）

Fishield项目可以对用户采取更加细粒度的权限划分，如果某用户没有上传权限并尝试上传文件，或没有下载文件却尝试下载文件会触发此错误。

**4 代码架构设计**

**4.1** 概述

整体代码一共可以大题分为数据库交互代码（DB\_Manager）、服务器端代码（Server）、调度器代码（Scheduler）、任务传输代码（Task）和客户端代码（Client）五部分。

Server

Client

Control Message through

TCP Sockets

DB\_Manager

Read or Write Data

to Dastabase

Task

Scheduler

Manage

Manage

Create

File Data through

TCP Sockets

图4.1 代码结构图

**4.2** 数据库交互代码模块（DB\_Manager）

所有与数据库相关的交互代码全部通过数据库交互代码模块执行。其他模块不直接与数据库产生交互。Fishield使用的是mysql数据库，数据库中主要存储了用户密码信息以及各项上传/下载任务的完成进度。

**4.3** 服务器代码模块（Server）

服务器代码模块是整个项目的核心，这个模块与其他所有模块都有交互。

服务器模块与客户端模块通过TCP Socket传递上一节提及的Fishield\_Request以及Fishield\_Response数据。这部分信息主要是控制信息，起到发起/终止/修改任务的作用。

在接到客户端发来的新建上传/下载任务的请求后，服务器端代码模块会创建一个任务对象（Task），并将这个对象交给调度器模块（Scheduler）统一调度管理。并将相应数据通过数据库交互模块（DB\_Manager）写入持久化的Mysql数据库。

在调度器模块（Scheduler）的调度下，任务（Task）对象会与客户端建立起TCP Socket通信渠道，并在该Socket中发送/接收所指定的文件。

**4.4** 调度器模块（Scheduler）

调度器模块拥有一个任务池（Task Pool），其中记录了所有当前处于活跃状态的任务。调度器模块会分离出一个线程通过轮询（Polling）的方式检查每一个任务，并根据每个任务各自不同的状态作出调整、更新调度数据。

**4.5** 任务模块（Task）

任务模块负责真正完成上传/下载文件的过程。任务模块负责与客户端程序建立TCP通信管道，并传输文件。传输文件的时候将文件切分成若干定长的包裹（Packet），以包裹为最小单位依次传递。并在每次传递完一个包裹的时候更新任务状态，检查包裹是否正确传递，实现应用层的可靠数据传输以及断点续传功能。

A screenshot of a cell phone

Description automatically generated

图4.2 任务状态

在实现中，每个任务从开始到完成，一共被分为了16个可能出现的状态，如图4.2所示。其中包括开始上传（UPLOAD\_INIT）、上传中（UPLOADING）、已上传（UPLOADED）、上传已暂停（UPLOAD\_PAUSED）、上传暂停中（UPLOAD\_PAUSING）、上传继续等状态（UPLOAD\_RESUME）、开始下载（DOWNLOAD\_INIT）、下载中（DOWNLOAD ING）、已下载（DOWNLOADED）、下载已暂停（DOWNLOAD\_PAUSED）、下载暂停中（DOWNLOAD\_PAUSING）、下载继续等状态（DOWNLOAD\_RESUME）、取消中（CANCELING）、已取消（CANCELED）、失败中（FAILING）、已失败（FAILED）等。

**4.6** 客户端模块（Client）

客户端模块一般负责向服务器端发起任务，并协助任务模块代码实现应用层的可靠数据传输。

**5 客户端UI界面设计**

**5.1** 概述

为了给用户更加流畅的使用体验，我提供了在界面中修剪所选取的图片的功能。用户可以在程序界面中自由的对所选图片进行剪裁、缩放和保存三种操作

**5.1** 概述

为了给用户更加流畅的使用体验，我提供了在界面中修剪所选取的图片的功能。用户可以在程序界面中自由的对所选图片进行剪裁、缩放和保存三种操作

**5.1** 概述

为了给用户更加流畅的使用体验，我提供了在界面中修剪所选取的图片的功能。用户可以在程序界面中自由的对所选图片进行剪裁、缩放和保存三种操作

**参考文献**

[1] [OpenCV](https://opencv.org/)

[2] [OpenCvSharp - .NET Framework wrapper for OpenCV](https://github.com/shimat/opencvsharp)

[3] [Getting Started With OpenCvSharp 3](https://www.codeproject.com/Tips/1085960/Getting-Started-With-OpenCvSharp)

[4] [[Cpp] SIFT and SURF](https://github.com/shimat/opencvsharp/wiki/%5BCpp%5D-SIFT-and-SURF)

[5] [OpenCV GUI基本操作，回调函数，进度条，裁剪图像等](https://blog.csdn.net/wangyaninglm/article/details/42032825)

[6] [SIFT算法](https://blog.csdn.net/lyl771857509/article/details/79675137)

|  |
| --- |
| 【结论】： |
| 【小结】： |
| 指导老师评语及成绩 |
| 【评语】：  成 绩： 指导老师签名：  批阅日期： |