密　　级：绝　密

项目编号：P-XDP-201907

文档代号：P-XDP-DD-1.0

# 云盘原型（XDP）系统设计

## Software Design of XDP

V 1.0

南京捷帝科技有限公司

2019 年 07 月

**版本控制**

| 版本号 | 日期 | 修改人 | 说明 |
|---|---|---|---|
| V 1.0 | 2019-07-25 | 夏曹俊 | 文档建立、初始化 |
| | | | |
| | | | |
| | | | |
| | | | |

# 目　录

# 1. 引言

## 1.1. 编写目的

为了验证技术可行性，开发一个云盘的原型系统。

本文档是 XDP 系统设计文档，作为进行编码的依据。本文档将作为后续软件实现工作的设计蓝图；同时也为公司内部及客户的相关人员所阅读，作为他们理解本项目的一份重要文献。

## 1.2. 背景

软件设计是整个软件开发过程中的一个重要环节，对概要设计进行细化，需考虑具体实现语言、实现平台，同时还要考虑各种非功能性的需求，如性能等。在此基础上，给出子系统中任务的部署情况，各任务之间的交互情况。

软件详细设计是软件开发过程中的一份重要文档，是进行编码的前提条件。

## 1.3. 范围

完成云盘的客户端和服务端，支持并发，支持文件上传、下载和目录列表刷新。

## 1.4. 术语

| 中 文 名 称 | 英　　文 | 缩写 | 含　　义 |
|---|---|---|---|
| 线程池 | Thread Pool | TP | 基于 libevent 搭建的线程池系统 |
| 事件 | Event | | |
| | | | |
| | | | |

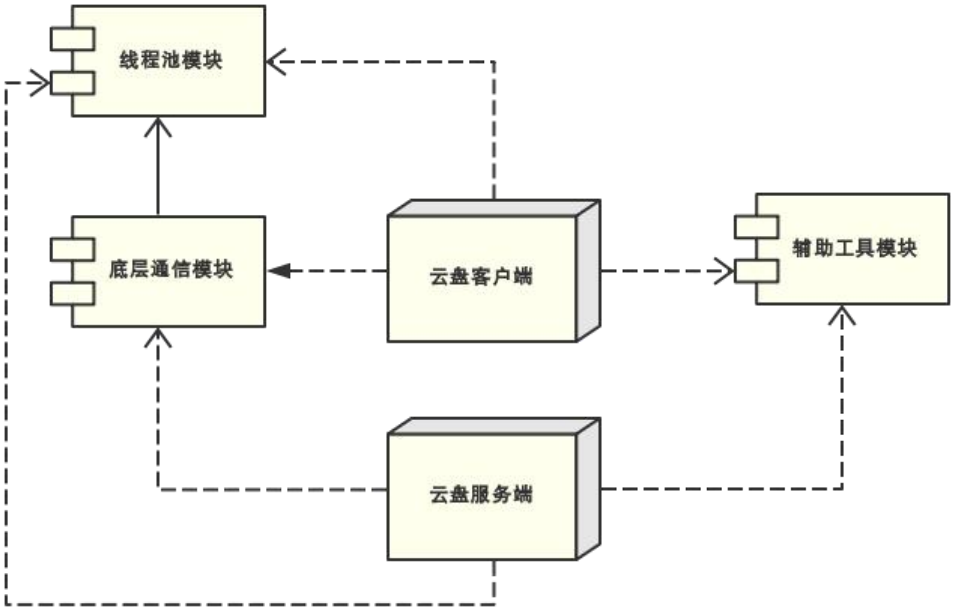
## 1.5. 参考资料

《计算机软件产品开发文件编制指南》（GT/B 8567-88）

# 2. 子系统划分及说明

XDP 系统划分为如下几个模块

1. 辅助工具模块：主要负责公共的工具，字符串处理和目录获取
2. 线程池模块：创建线程池和任务
3. 底层通信模块 ：客户端和服务端通信模块
4. 云盘客户端：前端上传下载和目录显示界面
5. 云盘服务端：后端，接收多用户并发访问，基于线程池



# 3. 消息格式定义

## 3.1. 消息格式

XDM 系统中所有消息将具有以下的统一格式：

| 消息类型 | 消息长度 | 消息体 |
|---|---|---|

消息各字段解释如下：

| 字段类型 | 长度(字节) | 描述 |
|---|---|---|
| 消息类型 | 4 | 标识本消息的类型 见消息类型定义 |
| 消息长度 | 4 | 消息体的总字节数 |
| 消息体 | | 消息体的具体内容 |

## 3.2. 消息结构体

```
struct XCOM_API XMsgHead
{
    MsgType type = MSG_NONE;     //消息类型
    int size = 0;                //消息内容长度（字节）不包含消息头
};
struct XCOM_API XMsg :public XMsgHead
{
    char *data = 0;              //用于存储消息内容
    int recved = 0;              //已经接收了消息内容字节数
};
```

## 3.3. 消息体描述

为了消息代码的简洁，消息体以二进制和普通文本存储，字段用半角逗号,隔开，每条记录用半角分号;隔开。

示例：

File1.zip,1024;File2.mp4,40960;File3.cpp,256;

## 3.4. 消息类型定义

消息类型不区分系统，直接以枚举存储类型（实际系统会替换为 protobuf，并根据系统编号），具体的类型定义将模块的消息列表说明

```
enum MsgType
{
    MSG_NONE = 0,
    MSG_OK,
    MSG_ERROR,
    MSG_GETDIR,
    MSG_DIRLIS，
    MSG_UPLOAD_INFO,
    MSG_UPLOAD_ACCEPT,
    MSG_UPLOAD_COMPLETE,
    MSG_DOWNLOAD_INFO,
    MSG_DOWNLOAD_BEGIN,
    MSG_DOWNLOAD_COMPLETE,
    MSG_MAX_TYPE
};
```

# 4. 辅助工具模块

## 4.1. 外部视图

### 4.1.1. 外部接口

#### 4.1.1.1. StringTrim

```
//////////////////////////////////////////////////////////////////////////
/// @brief 去除字符串头尾的空格
/// @param s 传入的字符串，会被修改
/// @return 返回修改后的字符串 s
std::string& StringTrim(std::string &s);
```

#### 4.1.1.2. StringSplit

```
//////////////////////////////////////////////////////////////////////////
/// @brief 切割字符串
/// @param str 传入的字符串
/// @param splitc 切割的字符
/// @return 切割后的数组
std::vector<std::string> StringSplit(const std::string &str, char splitc);
```

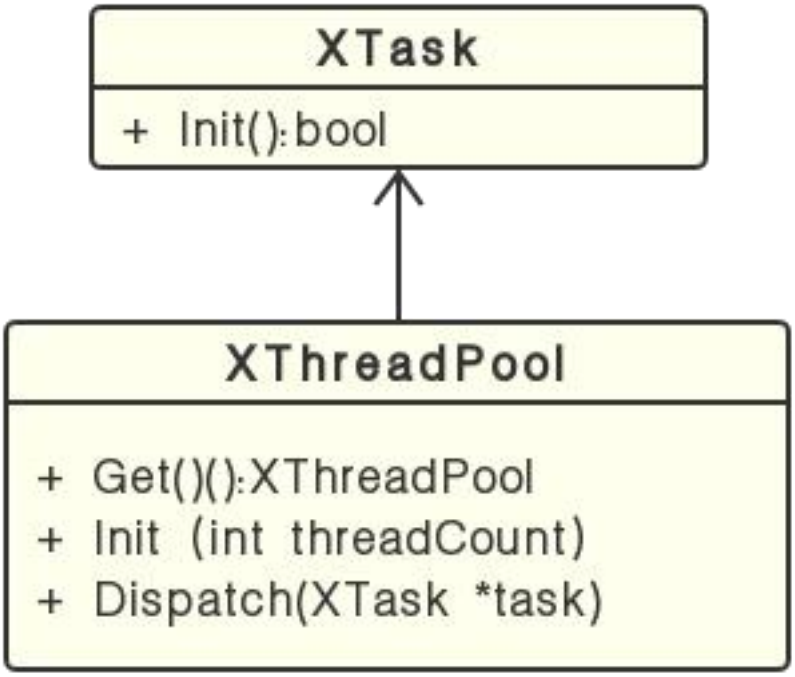
#### 4.1.1.3. GetListData

```
//////////////////////////////////////////////////////////////////////////
/// @brief 获取目录中文件列表，支持 windows 和 linux 目录，暂时不返回目录
/// @param path 目录的路径
/// @return 文件名 1，文件大小 1;文件名 2,文件大小 2;文件名 3,文件大小 3;
std::string GetListData(std::string path);
```

# 5. 线程池模块

## 5.1. 外部视图

### 5.1.1. 外部接口




#### 5.1.1.1. XThreadPool::Get

```
//////////////////////////////////////////////////////////////////////////
/// @brief 获取 XThreadPool 的静态对象 （静态函数）
/// @return XThreadPool 静态对象的指针
//////////////////////////////////////////////////////////////////////////
static XThreadPool* Get()
```

#### 5.1.1.2. XThreadPool::Init

```
//////////////////////////////////////////////////////////////////////////
/// @brief 初始化所有线程并启动线程，创建号 event_base ，并在线程中开始接收消息
void Init(int threadCount);
```

#### 5.1.1.3. XThreadPool::Dispatch

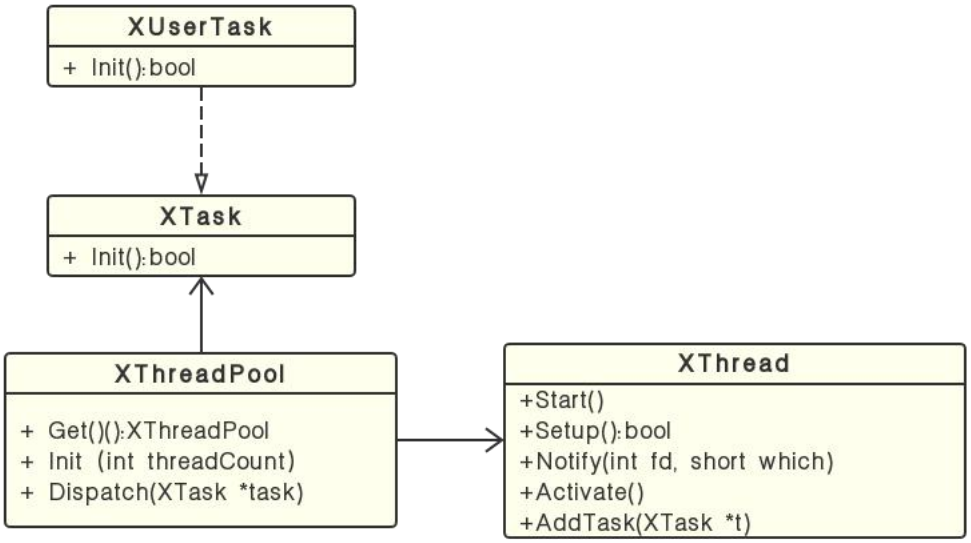
```
//////////////////////////////////////////////////////////////////////////
/// @brief 分发任务到线程中执行，会调用 task 的 Init 进行任务初始化
///        任务会轮询分发到线程池中的各个线程
/// @param task 任务接口对象，XTask 需要用户自己继承并重载 Init 函数
void Dispatch(XTask *task);
```

#### 5.1.1.4. XTask::Init

```
//////////////////////////////////////////////////////////////////////////
/// @brief 用户的具体任务实现函数，用户需要继承此接口类，并重载实现此方法
/// @param 初始化是否成功
virtual bool Init() = 0;
```

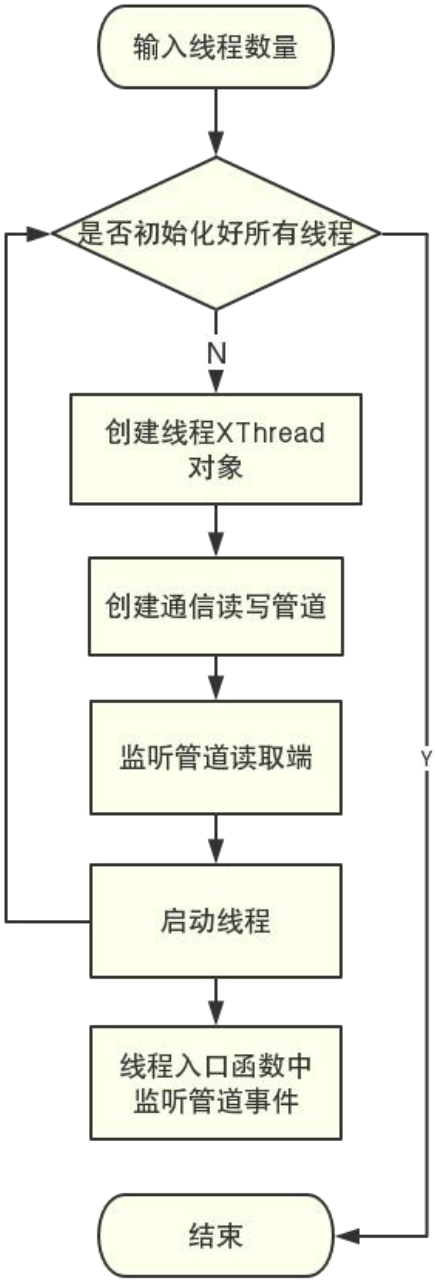
## 5.2. 内部视图

### 5.2.1. 类定义

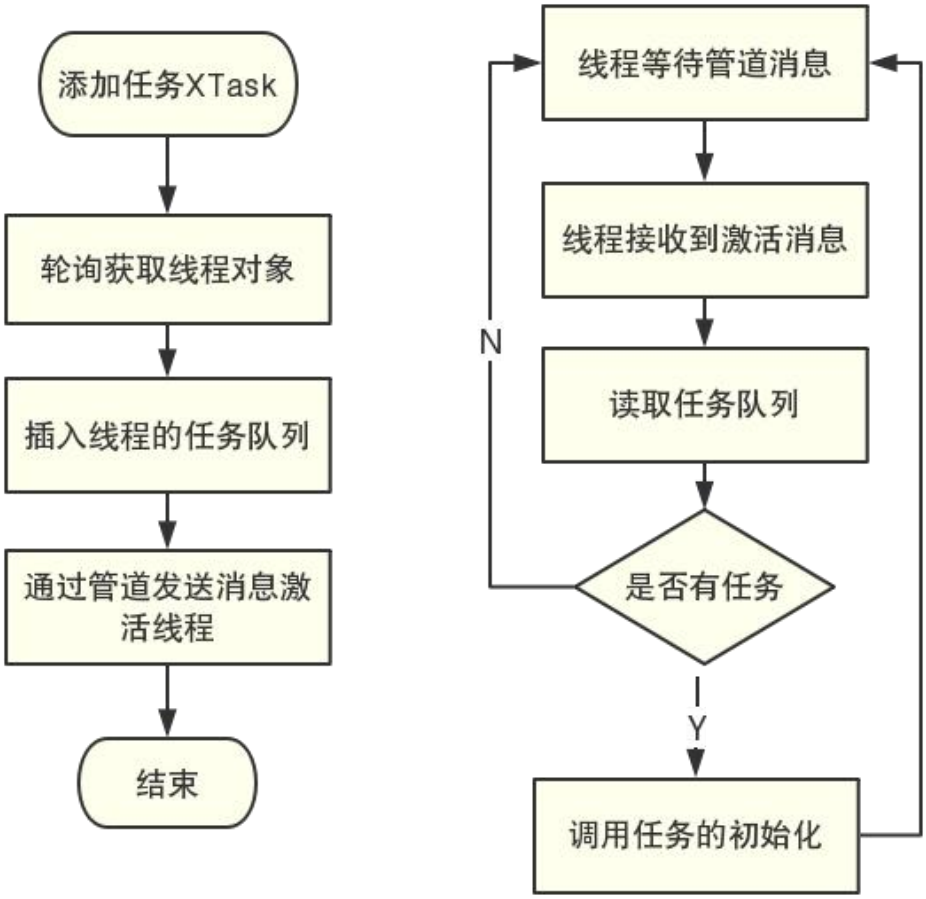


## 5.2.2. 接口操作实现

### 5.2.2.1. 线程池初始化流程图



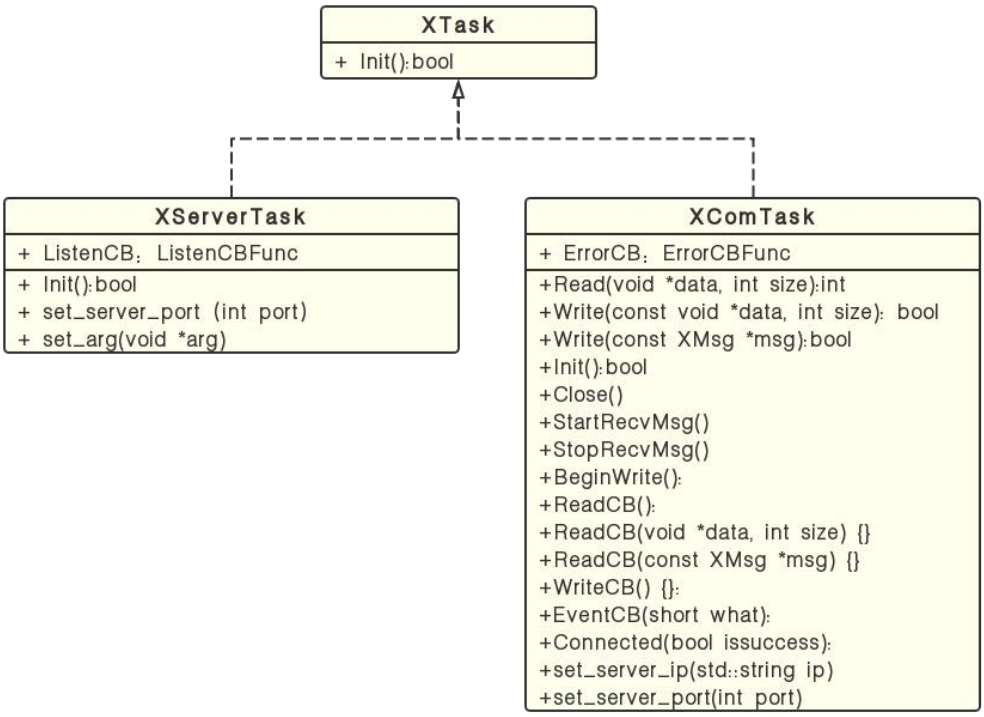
#### 5.2.2.2. 线程池任务处理流程



# 6. 底层通信模块

## 6.1. 外部视图

### 6.1.1. 外部接口



# 7. 云盘客户端

## 7.1. 外部视图

### 7.1.1. 发送消息列表

#### 7.1.1.1. MSG_GETDIR

**消息名称：**请求目录中文件列表

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_GETDIR | 消息体长度，包含字符串\0 | 目录路径 |

**消息体示例：**

./server_root

#### 7.1.1.2. MSG_UPLOAD_INFO

**消息名称：**请求上传文件

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_UPLOAD_INFO | 消息体长度，包含字符串\0 | 文件名和文件相对路径，文件大小 |

**消息体示例：**

test.mp4

#### 7.1.1.3. MSG_DOWNLOAD_INFO

**消息名称：**请求下载文件

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_DOWNLOAD_INFO | 消息体长度，包含字符串\0 | 文件名和文件相对路径 |

**消息体示例：**

test.mp4

#### 7.1.1.4. MSG_DOWNLOAD_COMPLETE

**消息名称：**下载文件已经完成

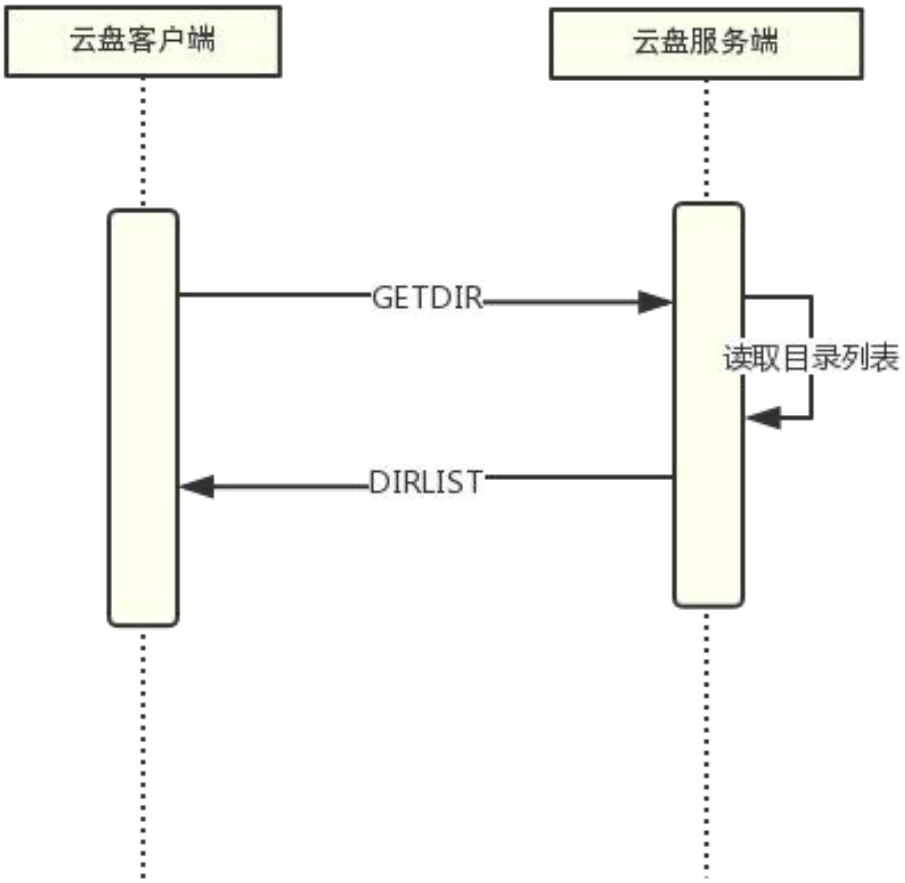
**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_DOWNLOAD_COMPLETE | 消息体长度，包含字符串\0 | OK |

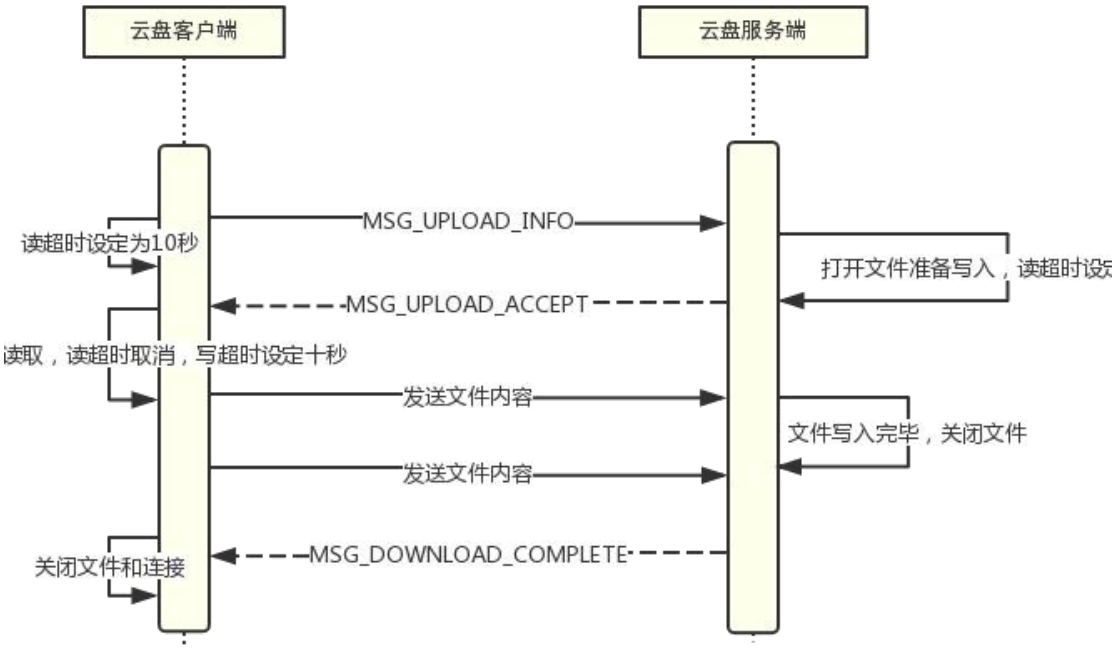
## 7.1.2. 操作顺序图

### 7.1.2.1. 目录获取顺序图

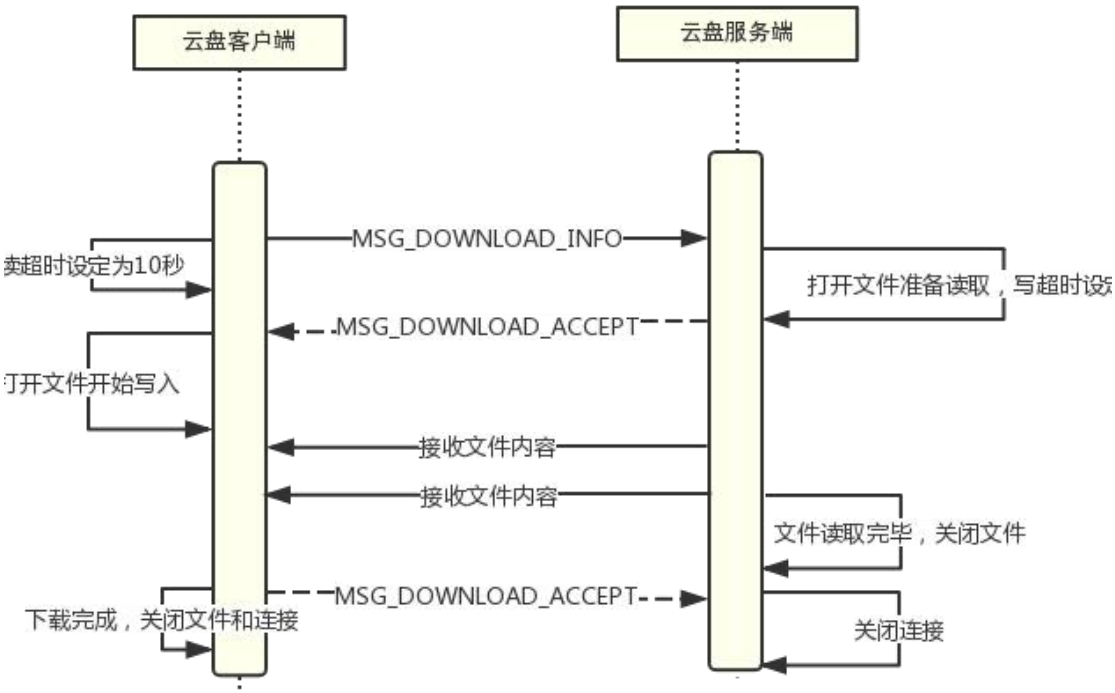
客户端设定接收超时 30 秒
服务端设定发送超时 10 秒



### 7.1.2.2. 文件上传顺序图

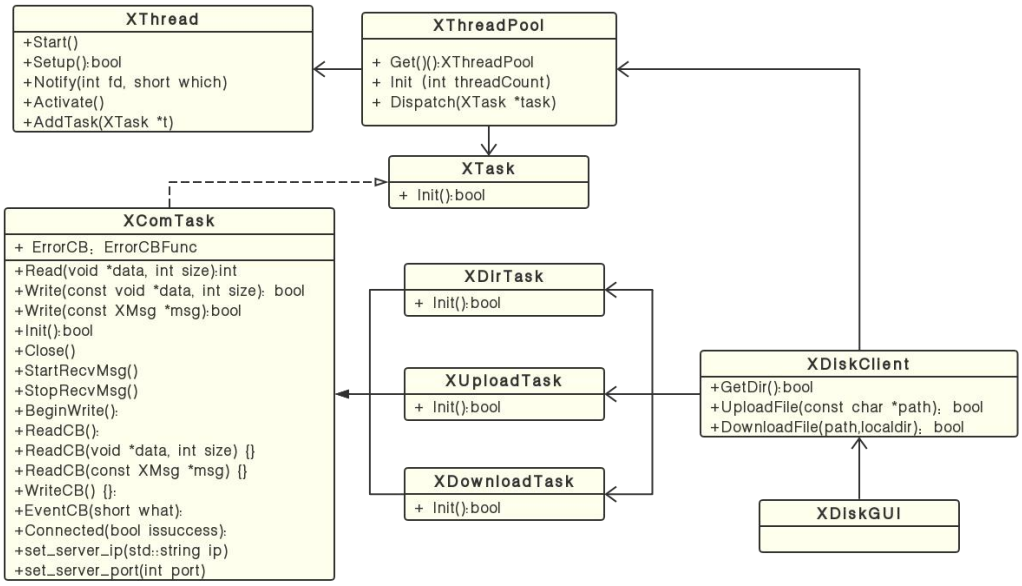


### 7.1.2.3. 文件下载顺序图



## 7.2. 内部视图

### 7.2.1. 类定义



# 8. 云盘服务端

## 8.1. 外部视图

### 8.1.1. 发送消息列表

#### 8.1.1.1. MSG_DIRLIST

**消息名称：**返回目录中文件列表

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_DIRLIST | 消息体长度，包含字符串\0 | 目录列表 |

**描述：**

消息体的内容包括以下字段：

1 文件名

2 文件字节数大小

文件 1 名称，文件 1 大小；文件 2 名称，文件 2 大小;

**消息体示例：**

File1.zip,1024;File2.mp4,40960;File3.cpp,256;

#### 8.1.1.2. MSG_UPLOAD_ACCEPT

**消息名称：**确认开始接收文件

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_UPLOAD_ACCEPT | 消息体长度，包含字符串\0 | OK |

#### 8.1.1.3. MSG_UPLOAD_COMPLETE

**消息名称：**确认接收文件完成

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_UPLOAD_COMPLETE | 消息体长度，包含字符串\0 | OK |

#### 8.1.1.4. MSG_UPLOAD_COMPLETE

**消息名称：**确认接收文件完成

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_UPLOAD_COMPLETE | 消息体长度，包含字符串\0 | OK |

#### 8.1.1.5. MSG_DOWNLOAD_ACCEPT

**消息名称：**确认可以开始下载

**格式：**

| 消息类型（4） | 消息长度(4) | 消息体 |
|---|---|---|
| MSG_DOWNLOAD_ACCEPT | 消息体长度，包含字符串\0 | 文件大小 |

### 8.1.2. 操作顺序图

见云盘客户端顺序图

# 8.2. 内部视图

## 8.2.1. 类定义