moCloud--服务器设计文档

# 历史

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Description** |
| V1.0.0 | WuJinlei | 20180226 | Create |
|  |  |  |  |

# 目录

[moCloud--服务器设计文档 1](#_Toc8292)

[历史 1](#_Toc26190)

[目录 2](#_Toc18047)

[1.概述 4](#_Toc30484)

[2.应用场景 4](#_Toc32009)

[2.1.初始化 4](#_Toc17020)

[2.2.文件管理相关场景 5](#_Toc20704)

[2.2.1.初始化 5](#_Toc7025)

[2.2.2.获取文件信息 5](#_Toc25782)

[2.2.3.读文件--下载、播放、读异常处理 5](#_Toc12137)

[2.2.4.写文件--上传文件 6](#_Toc14200)

[2.2.5.删除文件 7](#_Toc5063)

[2.3.客户端管理相关场景 7](#_Toc23187)

[2.3.1.初始化 7](#_Toc1905)

[2.3.2.收到新的连接 7](#_Toc32708)

[2.3.3.处理客户端的业务请求 8](#_Toc20979)

[2.3.4.主动清除客户端 8](#_Toc16926)

[2.3.5.广播消息 8](#_Toc13543)

[3.序列图 8](#_Toc27739)

[4.上下文图 9](#_Toc18816)

[5.模块图 9](#_Toc20163)

[6.内存和数据 9](#_Toc13146)

[6.1.线程 9](#_Toc2338)

[6.2.数据库 9](#_Toc6927)

[7.类图 9](#_Toc4524)

[8.流程图 9](#_Toc13847)

[9.头文件 9](#_Toc18367)

[10.开发阶段 9](#_Toc3836)

[10.1.M1--创建模型、支持基本功能 9](#_Toc6618)

[10.2.M2--上传、下载、续传文件 10](#_Toc6127)

[10.3.M3--删除文件、在线播放文件 10](#_Toc16464)

[11. 遗留问题 10](#_Toc9022)

[12.测试 10](#_Toc23157)

[12.1.功能测试 10](#_Toc30779)

[12.2.性能测试 10](#_Toc27228)

[12.3.边界测试 11](#_Toc12431)

# 1.概述

moCloud定位为一个私有云系统，包含了稳定运行的云服务器，和可以正常使用的云客户端；

Server运行在树莓派上，client实现在手机上。两者可以通过局域网互联，是moCloud正常交互的前提，后续可以考虑将服务扩展至公网。

由于树莓派本身的硬件性能和存储size有限，因此设计方案是：

1. Server端存放一套数据文件；
2. 所有的客户端看到的都是这一套数据；
3. 客户端分两种权限：管理员和普通用户；普通客户端只能上传文件、下载文件、播放文件、获取文件列表等；管理员除此之外，还可以删除文件；

本文档，介绍moCloud--服务器端的设计方式，实现流程，可以为后续参与该项目的人提供参考，也可以为后续维护提供依据。

# 2.应用场景

服务器端的主要工作，可以概述为如下几个方面：

* 客户端的管理：包括接收客户端的请求，密钥协商，响应客户端的其他控制指令，管理客户端的心跳包机制，等等；
* 本地文件的管理：包括同步更新所有的本地文件，文件发生变化时广播到所有客户端，支持文件的上传、下载、播放，等；

## 2.1.初始化

服务器启动后，通过“初始化”的操作，开始服务：

1. 读取配置文件，得到“云文件目录”的路径；
   1. 配置文件读取失败，或不符合规则，返回错误，退出运行；
2. 执行“文件管理模块”的初始化，详见2.2.1中相关描述；
   1. 如果初始化失败，退出运行；
3. 执行“客户端管理模块”的初始化，详见2.3.1中相关描述；
   1. 如果初始化失败，退出运行；
4. 读取配置文件，创建socket，监听端口，等待客户端连接；

## 2.2.文件管理相关场景

### 2.2.1.初始化

1. 传入的“云文件目录”是否存在：
   1. 目录不存在，创建；
   2. 路径有效，但路径并不是指向一个目录，而是一个文件或其他格式文件，返回错误，初始化失败；
2. 目录下的子目录是否符合规范：
   1. 约定的子目录不全，补全；
   2. 有其他的子目录，不处理；
   3. 子目录的路径指向的不是目录，而是文件或其他格式，返回错误，初始化失败；
3. 依次获取各个目录下的各个文件，判断文件是否在数据库中已经存在：
   1. 不存在，新建一条记录，并更新到数据库中；
   2. 存在，判断基本信息是否一致，不一致的，将本地文件的信息更新到数据库中；
   3. 本地文件不存在，但数据库中存在的，从数据库中删除；

### 2.2.2.获取文件信息

支持：获取所有文件的信息；获取一种类型的所有文件信息；获取单个文件的信息；

以获取单个文件的信息为例：

1. 查看文件是否存在于本地存储中：
   1. 不存在的话，返回错误；
2. 读取文件信息，并组织成需要的格式，并返回；

### 2.2.3.读文件--下载、播放、读异常处理

读文件涉及到的业务，包括了下载文件，播放文件；

下载文件和播放文件，由客户端管理模块的每个客户端对象，记录自己正在下载那个文件、下载的进度，下一次要下载的offset和size等；文件管理模块，只负责根据传入的offset和size，从文件中读取一部分内容，返回；

需要处理一种异常情况：假如server运行过程中，出现某一个文件被删除了，或者被人更换了文件体导致size变小了等，读取会失败，一旦读取失败了，我们要refresh本地存储的同时，也要广播给所有客户端，让他们重新获取文件列表，停止对这个文件的下载、播放等任务；这个过程称之为“读异常处理”；

1. 要读取的文件是否存在：
   1. 如果文件不存在，但是在数据库中有该文件的信息，说明是在本次服务器启动后，运行期间，文件被删除了，1.清空数据库中的本条文件记录；2.返回错误之后；3.触发“读异常处理”机制，告知所有client重新获取一次文件信息列表；
2. 状态是否是“deleting”：
   1. 如果状态是deleting，说明管理员马上要删除该文件了，不再支持读文件了，返回错误值；
3. 是否是第一次读文件：
   1. 是：打开文件，保存文件句柄，读的用户数量更新为1；
   2. 否：读的用户数量加1；
4. 根据传入的offset和size，读文件：
   1. 读文件失败，判断本地存储中文件信息和当前的文件信息，是否一致，如果不一致，说明服务器运行期间文件被修改过，需要：1.更新本地存储中的本条文件记录；2.返回错误；3.触发“读异常处理”机制，告知所有client重新获取一次文件列表；
5. 将读取到的内容，返回，一次读取完成；
6. 如果收到了“读结束”的标记：
   1. 如果已经是最后一个读用户了，直接关闭读句柄；
   2. 如果不是最后一个读用户，读的用户数量减1即可；

### 2.2.4.写文件--上传文件

写文件针对的是上传文件的场景，上传文件，要上传不同名字的文件，而且上传过程中将文件后缀命名为“.uploading”，上传文成后，由客户端管理者将该文件，重命名为正确的名字；具体流程参见客户端管理模块针对上传文件的处理；

1. 文件在数据库中，是否存在：
   1. 存在，说明文件已经被创建，判断写文件的用户是否和本次写文件的用户相同，
      1. 相同，读取文件句柄，执行步骤5；
      2. 不同，是另一个client试图上传同名文件，返回错误，告知client已经有用户上传该文件了，上传完成前不能重复上传；
   2. 不存在，说明是第一次写该文件，执行步骤2；
2. 文件在本地存储是否存在：
   1. 如果已经存在了，不能够再次写，必须更换其他名字后再写；因此返回失败；
3. 以写模式打开文件：
   1. 打开文件失败，返回失败；
4. 更新文件信息到本地存储；
5. 将传入数据，按照传入的offset、size，写入文件：
   1. 如果出现了跳block的情况，中间的block先写入全0值占位；

如果收到写完成的指令，将文件重命名，取消“.uploading”后缀，并更新到本地存储中；

### 2.2.5.删除文件

删除文件的操作，只有管理员权限才可以做，用户不可以做；

1. 查看发起者的权限：
   1. 普通用户权限，不允许删除文件，返回错误值；
2. 查看该文件在本地存储中是否存在；
   1. 不存在，返回错误；
3. 更新该文件的状态到deleting；
4. 删除文件；
5. 更新本地存储；
6. 触发“读文件异常”机制，告知所有client更新文件列表；

## 2.3.客户端管理相关场景

客户端管理模块负责管理：

* 接收新的客户端连接请求；
* 管理所有的客户端连接，例如心跳机制；
* 为每个客户端提供单独的对象，处理其业务请求；
* 其他相关业务；

### 2.3.1.初始化

初始化操作，准备相应的数据结构，分配需要的内存，启动对应的线程等；

1. 清空存放所有客户端信息的map；
2. 启动“客户端状态管理”线程，该线程负责定时检查客户端的状态，如果状态更新为了invalid，说明心跳超时或者客户端主动发起了byebye的动作，需要清除该客户端的信息；
3. 启动接收连接的主线程，该线程负责accept新的客户端请求；

### 2.3.2.收到新的连接

1. 判断该IP是否已经存在于map中：
   1. 存在，步骤2；
   2. 不存在，说明是新的client的控制端口发来的连接请求，步骤3；
2. 查看该IP在map中，控制端口、数据端口的状态：
   1. 控制端口有效，数据端口也有效：该IP不能再发起connect请求，一个client只能启动一个客户端，返回错误；
   2. 控制端口有效，数据端口无效：本请求是该IP的数据端口发起的，将本次的请求加入map中，返回正确值；
   3. 与控制端口重复：不能重复使用同一个ip和端口发起connect命令，返回错误；
3. 将该请求加入到map中；
4. 创建一个clientMgr对象，该对象负责接收该client的所有控制指令，并处理之；

### 2.3.3.处理客户端的业务请求

TODO

### 2.3.4.主动清除客户端

清除客户端的动作，触发源分为两种：客户端异常了导致心跳包没有定时发送，服务器要清除它的信息；客户端不再登录server了，发送了byebye指令，server要清除它的信息；

1. 遍历map，找到状态为invalid的clientMgr对象；
2. 删除该对象；

### 2.3.5.广播消息

“读异常机制”需要能够广播消息给所有的client，告知他们重新获取文件列表；这里的广播就是这个意思。

1. 遍历map中各个客户端；
2. 状态为running的，说明已经做过密钥协商、登录了的，且有效的client，执行步骤3；否则，跳过该client，不予发送信息；
3. 组织消息体，发送到client；

# 3.序列图

# 4.上下文图

# 5.模块图

# 6.内存和数据

## 6.1.线程

## 6.2.数据库

# 7.类图

# 8.流程图

# 9.头文件

# 10.开发阶段

## 10.1.M1--创建模型、支持基本功能

第一阶段，主要创建客户端和服务器模型，保证正常的通信机制，和基本的业务模型，具体来说，主要是如下几个方面：

* 加解密模块：支持正常的加解密；
* 信息校验模块：支持需要的校验算法；
* 通信模块：client和server可以正常通信；
* 心跳包管理模块：客户端可以正常发送心跳包，服务器可以管理心跳包，两者都可以根据心跳包的正常与否，做出正确的反应；
* 基本功能之密钥协商：可以正常进行密钥协商；
* 基本功能之获取文件列表：client可以正确获取到server当前的所有文件的列表信息，并展示给用户；

1. MoCloudUtilsCrypt，moCloudUtilsCheck，打桩； OK
2. moCloudUtilsType，moCloudUtils，实现；OK
3. moCloudClient：
   1. Init: connectToServer; doKeyAgree; OK
   2. startHeartBeat : sayHiToServer; OK
   3. signUp : 注册; OK
   4. signIn : 登录；OK
   5. getFileList; OK
   6. unInit: sayByeByeToServer; freeAllResources;OK
4. moCloudServer：启动，接收client的connect请求，每一个client新建一个thread处理；
5. moCloudServer：文件管理模块；
6. moCloudServer：client管理模块，心跳包管理最重要；
7. moCloudServer：支持基本功能之密钥协商和获取文件列表；
8. 正式实现moCloudUtilsCrypt；
9. 正式实现moCloudUtilsCheck；

## 10.2.M2--上传、下载、续传文件

## 10.3.M3--删除文件、在线播放文件

# 遗留问题

# 12.测试

## 12.1.功能测试

## 12.2.性能测试

## 12.3.边界测试