开发相关技术及解决方案

经团队全体成员共同努力，我们完成了一个不同设备间分享传输文件信息的软件系统。系统可以借助互联网、局域网进行文件互传，也可于无网络情况下在移动终端间进行文件互传。系统开发过程中主要解决的关键问题及创新点总结如下：

一、web 服务器端解决方案

Web 服务端主要为 App 提供文件中转服务支持，并为未安装 App 的用户提供简单的 Web 界面的文件中转服务。后端采用三层架构进行设计，降低层次之间的耦合度，主要使用的技术是 c3p0+dbutils+JavaBean+Servlet+JSP，并在中转过期文件及时清理，服务器临时文件清理，数据库清理，多用户并发安全方面进行了考虑，结合二维码，并模仿微信 PC 客户端的扫描登录原理对用户的操作进行简化，争取在服务器端为用户提供高质量的服务。

1．文件中转服务

服务器负责中转客户端发送的文件，考虑到这个文件中转服务的即时性，只对文件暂时保管 5 分钟，但是这个参数写在配置文件中，日后可以根据需要随时修改文件暂存时间。对于文件和用户的绑定关系，客户端在上传文件时，服务器会根据设计好的算法为文件分配一个 5 分钟内有效的文件 id，并且为了防止其他恶意用户随机试验文件 id 而获取其他用户的文件，在用户上传文件要求提供一个简单的验证密码，也是 5 分钟有效期。用户上传完成之后可以打开 app 网站，输入这个文件 id 和验证密码很方便的获取文件，如果手机端安装了 App 客户端，打开客户端扫一扫网站首页的二维码，也可以很方便的获取文件，省去了输入文件 id 和验证密码的麻烦。

2．服务器对暂存文件的存放

服务器首先会用 UUID 算法为上传的每个文件分配一个唯一的文件名，并在数据表中保存了文件的原文件名，服务器根据生成的 UUID 文件名计算出哈希码，

根据哈希码将上传的所有文件分散到服务器上传文件夹根目录的 256 个文件夹中，防止一个文件夹中存放太多文件导致的文件检索速度的降低，并且用户查找文件时，可以根据数据表中的文件 UUID 名直接定位到指定的文件夹中找到文件，达到了分散文件和快速检索的目的。

3．文件 id 算法：

服务器事先在数据表生成了十万条记录，相当于十万个中转服务空槽，每当用户上传文件，就会从这些空槽中找出一个可用的分配给当前用户，并用这个空槽在表中的主键 id 和随机数生成文件 id，和当前文件绑定。

4．可用的空槽

数据表中有两个标记位，一个标记当前空槽是否可用，另一个标记空槽申请时间，一旦空槽标记位未被占用或者当前时间超过申请时间 5 分钟，便视为这个空槽可用。

5．多用户并发

数据表操作进行了事务管理，保证一个空槽被申请时，这个空槽不会被其他用户占有，并且使用了简单的验证密码保护用户文件，考虑到系统设计初衷只为文件保存 5 分钟，因此验证密码为了用户输入方便规定为 6 位数字以下的简单密码，在短时间内能够有效的保护住用户的文件。

6．文件中转服务策略

文件大小限制：单个文件大小限制 10M，多个文件总大小限制 50M。

上传文件列表查看：服务器中转文件只保存 5 分钟，用户在文件上传之后 5 分钟内，可以根据文件上传时服务器返回的文件 id 和验证密码查看自己的上传文件列表，五分钟内任意次数有效。

文件下载：用户上传的每个文件有一次下载机会，用户下载过后，该文件将会从用户的上传文件列表中删除，服务器文件也将删除，从一定程度上防止用户文件被恶意用户窃取，并保护服务器存储空间，也和该项目的设计理念一致（提供简单一次性文件中转服务，不提供文件存储服务）。

7．临时文件清理

该文件中转服务对文件仅提供一个中转服务，用户上传文件之后，通过文件 id 和验证密码获取文件列表，可以下载列表中的文件，下载后把数据表中的该文件 id 数据重置，以提供下次其他用户重复使用，而用户上传的文件会在用户下载请求完成之后自动删除，二维码文件也会在扫描成功之后自行删除。

对于用户上传好的文件但是没下载，或者没被扫描成功的二维码文件，系统使用一个定时任务在每天凌晨自动扫面所有文件，并删除存留超过 5 分钟的所有文件以清理垃圾。

8．Web 页面服务与 App 服务

服务器使用一个 http 参数标记向服务器发送请求的是浏览器还是客户端，并对此进行相应的响应信息加工，若是浏览器，将会使用 JSP 技术生成动态网页返回给浏览器，若为 App 客户端，则返回主要信息的 json 数据，方便客户端进行解析。

二、Android 客户端解决方案

1．客户端网络模块

Android 的 App 客户端采用了 HttpClient 作为主要网络交互模块，并对此进行了封装，使用 gson 解析 json 数据的部分也封装在 http 网络模块的解析模块中，采用面向接口编程，降低了模块的耦合度，可以根据业务需要很方便地替换 gson 或者 httpclient 模块为其他模块。

2．客户端之间文件传输模块

采用 Android4.0 之后提出来的 WiFi p2p 技术，与传统的蓝牙传输，热点传输相比，具有传输速度快，更稳定。该技术虽然依赖 Android 设备的 WiFi 模块，但是在文件的传输过程中 Android 设备依旧可以保持 WiFi 对于网络的连接，并不会出现断网的情况。

4．局域网传输模块

采用ftp形式，即在Android设备开启ftp服务器，PC通过ftp客户端（Windows文件管理器）来进行文件管理。

三、系统的创意创新点

1．Android客户端UI采用Google自Android5.0开始提出的Material Design标准模式，对比Android5.0之前的系统，界面有非常大的改善。

2．Android设备之间传输文件采用Android4.0之后新添技术标准WiFi P2P技术，该技术传输距离可达300米，速度理论可达250Mbps。

3．在有网传输的情况下，Android设备和PC设备之间采用扫描二维码的方式以减少用户的手动输入文件获取信息。

具体过程如下：

网页文件上传：通过网页上传文件之后服务器会反馈回二维码，客户端只要扫描该二维码就能获取服务器返回的文件 id 和用户设置的验证密码，若用户摄像头损坏，还能根据网页显示的文件id和验证密码手动输入来获取文件。

网页文件下载：手机客户端上传的文件，服务器返回文件 id 和验证密码给客户端，这里仿照微信PC客户端扫描登录的原理，网站首页的二维码和服务器生成的一个唯一 id 绑定，当客户端扫描二维码时，会通过客户端把这个二维码id和客户端上的文件id及验证密码在服务器进行绑定，而在首页上，通过js对这个二维码id向服务器发送http轮询，及时检查二维码id的绑定状态，当查询到二维码id和一个文件id及验证密码绑定之后，自动跳转到文件id指定的上传文件列表供用户下载。

4．服务器端文件存储根据文件名的哈希码打散，有效解决同一文件夹下文件过多导致的效率问题，检索的时候可以通过文件名重新快速定位文件路径，两全其美。

5.服务器端网页采用http轮询的方式实现类似微信PC端的扫码逆向登录，只要是用户手机摄像头没问题的情况下，都可以用扫描二维码的形式简化用户的操作方式，提升用户体验。

6．服务器端数据表和文件id设计和生成，采用预先生成的十万条记录作为

文件中转空槽，每个记录的主键 id 作为生成文件 id 的决定性因素，这样保证了同一时间段内的有效文件 id 的唯一性，并减少了数据表记录添加导致的事务对整个数据表的锁定，提高一定的并发性能。并且在从数据表中获取空槽采用以主键最低优先的策略，尽量提高数据库查询性能。

四、开发与运行环境

1．服务器端：

开发环境：IntelliJ Idea14+JDK8+Tomcat8+Mysql5.5+JavaEE7

运行环境：Centos6.5 64 位+Tomcat8+JDK8+MySQL5.5

2．客户端开发环境：

Android Studio 1.5，Jdk_1.8

五、客户端参考的主要开源项目及文档：

[1] https://github.com/florent37/MaterialViewPager

[2] https://github.com/zxing/zxing

[3] https://github.com/nexes/Android-File-Manager

[4] https://github.com/ppareit/swiftp

[5] google 官方文档