# P2P修改及新功能开发说明

P2P新功能的开发主要包括P2P修改点、P2P点播的开发、P2P log数据统计以及P2P升级功能调试

# 1.P2P修改点

p2p修改点使用宏“SM\_MODIFY”控制管理，使用“SM\_DBG”控制debug输出信息，修改点主包括如下：

1）接口中添加上报NAT类型的参数

涉及到文件DBInfo.h，UserManager.cpp，LogManager.cpp

2）针对letv不能播放的修改

letv的源不是正常的参数，主要体现在，一是在HTTP的header中没有ContenLen参数，二是在header和content之前只有一个“\r\n”，这些都需要特殊处理

涉及到文件HttpGet.cpp，HttpPeer.cpp

3）修改tracker的log文件不每天生成一个的问题

涉及到文件LogManager.cpp，UserManager.cpp

4）修改获取peer的版本信息的接口

针对不同的版本的peer，只需修改宏“VERSION\_STR”加以区分，并保存在version.txt

涉及到文件HttpsvrHandler.cpp，Main.cpp

5）针对peer的httpserver的端口的修改

peer的httpserver的端口是默认的是14188，但如果当前端口被使用的话，会逐步增加试探，直到成功，修改把成功的端口保存到port.txt文件中

涉及到的文件HttpsvrHandler.cpp，Main.cpp

6）获取url中参数的接口

获取开始和结束都指定之间的参数的接口

涉及到的文件HttpClient.cpp，HttpsvrHandler.cpp

# 2.P2P点播功能开发

## 2.1 p2p系统针对直播与点播的差异

分析p2p系统针对直播与点播的差异如下：

1）针对分片，直播的分片是一直在更新，而点播的分片是定死的，更新分片的策略；

2）针对存储，直播不需要存储在本地，点播由于数据量大，需要存储在本地，是否缓存以及动态切换的策略；

3）针对分配的内存空间，直播不需要很大，点播需要很大的空间，存储空间的动态调整策略；

4）针对存储下载信息，直播不需要存储下载信息(包括已下载的和正在下载的)，点播需要，以便下次播放更快；

5）针对取下载块的策略问题，直播需要实时，取下载块时会出现抢夺、少取保证流利播放等问题，点播不需要，策略的动态切换；

6）针对上报tracker分享文件，直播只需要上传节目的hashid，而点播还需要上传节目的hashid和节目分片的hashid；

7）针对tracker，目前直播只收集的是针对节目id的p2p源，而点播还需是节目的分片id的p2p源，tracker的取源策略；

8）针对缓存文件清除问题，是按已缓存文件的大小还是近缓存文件的个数;

9）针对下载的跳跃问题，直播是不能快进的，点播可以随时快进，下载的动态调整策略；

## 2.2 P2P点播目前的现状

1.点播可以缓存指定大小的视频数据，会自动维护缓存

2.点播才有缓存，直播不保留缓存

3.点播可以获取已缓存视频的时间，直播没有。--边看边缓

4.peer每次打开，都会向tracker上报自己已缓存视频，关闭会向tracker上报删除

5.视频刚开始时（缓存视频文件小于2），会只下载前1、2个视频，之后（60s），会自已启动下载多个（4个）

6.点播，当当前视频文件的p2p源多于一定数时（比如10），会关闭http连接，只启用peer下载

## 2.3 p2p点播功能开发点

### 2.3.1 tracker的上报

修改tracker上报时添加文件类型，主要区分点播vod与直播live，上报共享文件、上报删除文件、上报开始（停止）下载/下载列表、请求文件源、上报下载失败，这样tracker端能精确的区分出上报的信息，并方便区分管理；

note：以上都只上报文件类型为PLAYTYPE\_VOD且hash类型为HT\_URLDL或PLAYTYPE\_LIVE且hash类型为HT\_URL2

添加接口：

int PTL\_ReportShareFile(const hash\_t& hash, int filetype);

int PTL\_ReportShareFile(const hash\_t& hash, int filetype, uint64 size);

int PTL\_ReportRemoveFile(const hash\_t& hash, int filetype);

int PTL\_ReportStartDownloadFile(const hash\_t& hash, int filetype);

int PTL\_ReportStartDownloadList(const hash\_t& hash,int filetype,const string& url);

int PTL\_ReportStopDownloadFile(const hash\_t& hash, int filetype);

int PTL\_RequestFileSource(const hash\_t& hash, int filetype);

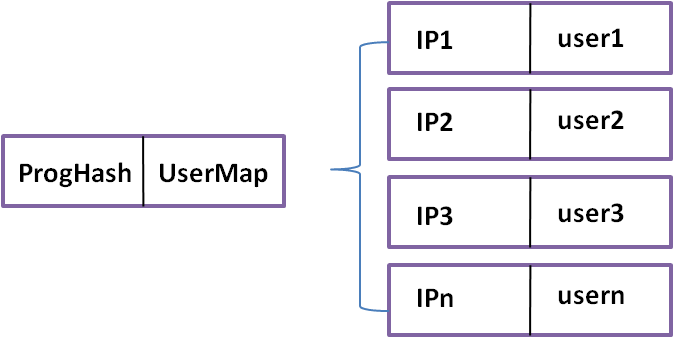
int PTL\_ReportDownloadWrong(const hash\_t& hash,const hash\_t& newHash, int filetype);

主要涉及到文件Tracker.h，Tracker.cpp

### 2.3.2 tracker端source的搜索管理

#### 2.3.2.1数据格式

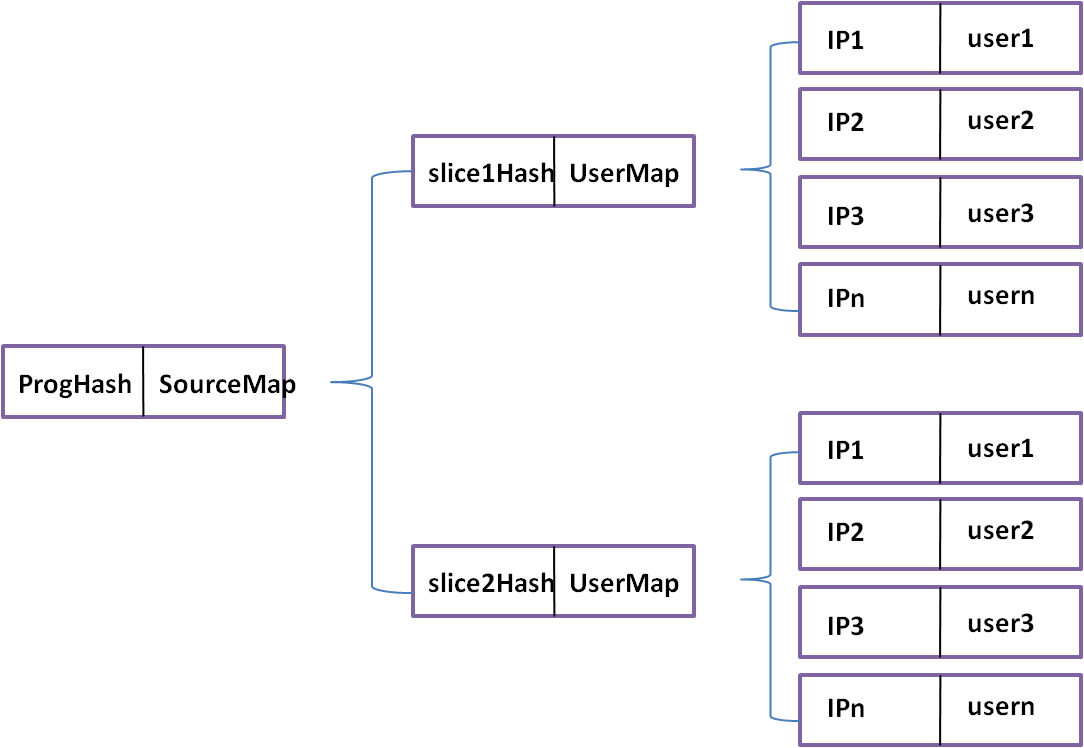
直播的源记录格式typedef map<hash\_t,SourceInfo\*> SourceMap，key是节目的hash值，value是源的信息，其中有个userMap（multimap<uint32,UserInfo\*> UserMap），是以ip为key的map，这样可以通过节目hash值得到source源的信息，进一步可以得到p2p源的信息；



其中user格式map<hash\_t,uint32> sourceMap; //共享文件列表map<hash,urlflag>，这样也就知道了每个p2puser的所看的节目；

由于直播是都在看同一节目，而且是时时更新的，这样能保证从节目的hash key获取的p2p源一定是用户需要的源，而点播不一定，由于点播的切片比较的多且固定，如果也是按照同样的策略，这样不能保障所得到的源一定你所需要的片，根据这点设计记录点播源的数据结构，再加一层索引方便快速的搜索p2p源；

typedef map<hash\_t,SourceMap\*> SourceMapvod，key是节目的hash，value是map<hash\_t,SourceInfo\*>，他的key是节目的片段的hash值，这样可以直接的得到某个节目的某个片段的p2p源，能迅速的定位；



格式map<hash\_t, map<hash\_t, uint32> > sourceMapvod；这样了就知道了每个p2puser所有的节目以及节目的片段；

添加接口：

void source\_clear\_ex();

int urldlsource\_delete(const hash\_t &tth,UserInfo\* user);

int url2source\_delete(const hash\_t &tthpl, UserInfo\* user);

int source\_delete\_vod(UserInfo\* user);

int source\_delete\_vod(const hash\_t &tth,UserInfo\* user);

int source\_add(const hash\_t &tth,int sourcetype, uint64 size,uint32 block\_size,UserInfo \*user);

int source\_find(const hash\_t &tth,int sourcetype, UserInfo \*user,UserInfo \*arr[],uint64& size,uint32& block\_size,int max\_count = 100);

int get\_source\_num(int& file\_num\_live,int& source\_num\_live, int& file\_num\_vod,int& source\_num\_vod);

void print\_user\_source(UserInfo \*user);

void print\_sources();

主要涉及到的文件SourceService.cpp，SourceService.h

#### 2.3.2.2 peer源的搜索

直播的搜索是按节目搜索，点播的搜索是按照节目的片段搜索，在tracker端由节目片段，索引到节目，再找到片段，再找到片段的p2puser

主要涉及到的文件DownloadList.cpp，Download.cpp

### 2.3.3 peer端关闭/开启http的策略

live的最大下载任务个数使用宏“DOWNLOAD\_ACTIVE\_NUM\_LIVE”控制，当前是2，vod点播的个数从是配置文件中读取的SettingSngl::instance()->get\_downloadlist\_active\_partnum，当前是4，但刚开始播放时，为保障第一个片段的正常迅速播放，使用策略是，如果已下载完成列表小于2时，只分配一个下载任务，当已下载完成列表大于2时再恢复；

关键接口：

void check\_create\_download()

主要涉及到的文件DownloadList.cpp

### 2.3.4下载分配任务的超前分块

直播注重的是获取视频的时时性，能正常播放即可，获取数据不能太慢，也不必过快。但点播就不一样了，由于数据是早已经生成好的，点播除了获取数据的速度可以再快些，早下载可以早分享，那p2p数据分享就多下载些；

在分配给视频下载时的，从p2p源获取时多分配些。

关键接口：

int assign\_job(Peer \*peer,bool allowReqBlockTable/\*=true\*/)

主要涉及文件Download.cpp

### 2.3.5下载列表的更新问题

直播的节目列表即m3u8列表在不断的更新，但点播的列表是固定的不需要更新，这样就在针对是vod时，在列表下载开始的时候先下载完整的列表，下载完列表后，停止列表的刷新的操作，避免重复且费时费流量的操作；

关键接口：

DownloadList(void)

int start()

void on(HttpGetListener::Response,HttpGet\* hg,int result,const string& save\_path,const char\* orig\_rsp,int len)

涉及到的文件DownloadList.cpp，DownloadList.h

### 2.3.6更换节目的前一节目的处理

直播更新新节目的处理是，把旧的列表关掉，并把相关的数据清除，但点播不能这样，点播只能关掉列表，但不能删除前一个节目的数据，即即使不看前一个节目，但前一个节目自己已经下载下来了，就要继续给其它peer提供

void clear\_cache()

int delete\_file(const hash\_t& hash,bool isDelPhy, bool isclear)

int delete\_fileinfo\_noerase(const hash\_t& hash,bool delfile);

int delete\_downinfo\_noerase(const hash\_t& hash,bool delfile);

int delete\_readyinfo\_noerase(const hash\_t& hash,bool delfile);

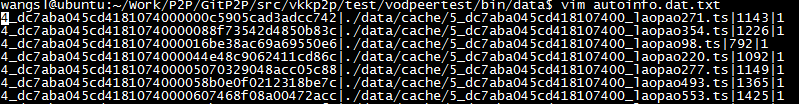
主要涉及到的文件DownloadList.cpp，DownloadList.h，DownloadManager.cpp，FileStorage.cpp，FileStorage.h

### 2.3.7本地存储的自动管理

本地存储的自动化管理思路是，新建文件时添加文件的记录，并计算出文件列表中的存储大小，若大于临界值则会删除掉一些文件直到有足够的空间用于存储；

删除策略是先删除hash中考前的节目（节目的所有文件），若除了当前的节目，其它节目都删除完了空间还不够的话，则删除当前节目比较先前的文件片直到空间足够（当前一次删除4个片段）

所有新建的文件记录在"autoinfo.dat.txt"文件中，格式"%s|%s|%d|%d"，分别是文件的hash，文件的路径，文件的score（记录的访问文件的先后顺序），文件的类型（直播还是点播）；这样score越小的文件越早，删除时也就先删除



关键接口：

int add\_autoinfo(const hash\_t& hash,int filetype,const string& path);

void remove\_autoinfo(const hash\_t& hash, int filetype);

void on\_read\_refer(const hash\_t& hash, int filetype);

bool get\_autocache\_vodmap(AutoMapvod& mp)

int auto\_clear\_cache(const hash\_t& hashExclude)

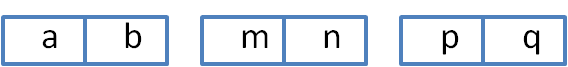
主要涉及到的文件FileAutoCache.cpp，FileAutoCache.h

### 2.3.8 动态获取缓存时间

为支持播放时能显示缓存的进度，即满足边看边缓，添加动态获取已缓存视频时间的接口，主要思想是获取当前能播放的视频的绝对时间，分析下以下两种情况的缓存时间

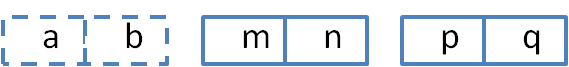
1）视频正常

视频正常已经缓存到q块，则q块绝对时间就是已缓存的时间



2）视频的开头一部分视频可能已经删除掉

视频已经缓存到q块，但视频开头的一部分视频已经被删除掉（可以由于空间不足的问题，删除了，比如a、b块），虽然已经被删除掉，但已缓存的时间的还是q块的绝对时间，这样还是能直播快进到q块的。



获取缓存时间的接口“http://127.0.0.1:14188/playlist/cachetime?type=x&url=playlist\_url”

主要接口：

void response\_playlist\_cachetime(int fd,const string& cgi,const string& head)

int downloadlist\_getcachetime(const string url,int& time)

int get\_downloadcachetime(const string& url, int& time)

int get\_downloadcachetime(int& time)

主要涉及到文件：

HttpsvrHandler.cpp，Interface.cpp，DownloadListManger.cpp，

# 3.P2P log数据统计

P2P log的含义参考Tracker分析.docx中的“6.LOG系统”部分，数据统计部分使用python提取出用户的数据信息及实时的统计信息，这个只讲述如何部署、输出结果，如何实现可以参见脚本“p2pDataHandleTask.sh”和“p2pDataHandle.py”

## 3.1 部署及使用说明

在服务器“103.244.165.202”中使用指令编译定时任务，"$crontab -e"，如下红色即其中的内容：

# m h dom mon dow command

59 23 \* \* \* /home/tvfan/P2P/trackerbin/p2pout/p2pDataHandleTask.sh

Note：即每天23:59分执行脚本“p2pDataHandleTask.sh”，p2pDataHandleTask.sh脚本内容如下：

#!/bin/bash

datestr=`date +%Y-%m-%d`

python /home/tvfan/P2P/trackerbin/p2pout/p2pDataHandle.py /home/tvfan/P2P/trackerbin/tracker\_log /home/tvfan/P2P/trackerbin/p2pout $datestr

即执行python处理程序，参数分别是要处理的log目录、输出结果的目录、要处理的文件的日期，从中可以看出shell及python脚本位于目录“/home/tvfan/P2P/trackerbin/p2pout/”中，输出的结果在目录“/home/tvfan/P2P/trackerbin/p2pout”，每天更新。

Note:

安装软件：

$sudo apt-get install python-pip //安装python安装软件的软件

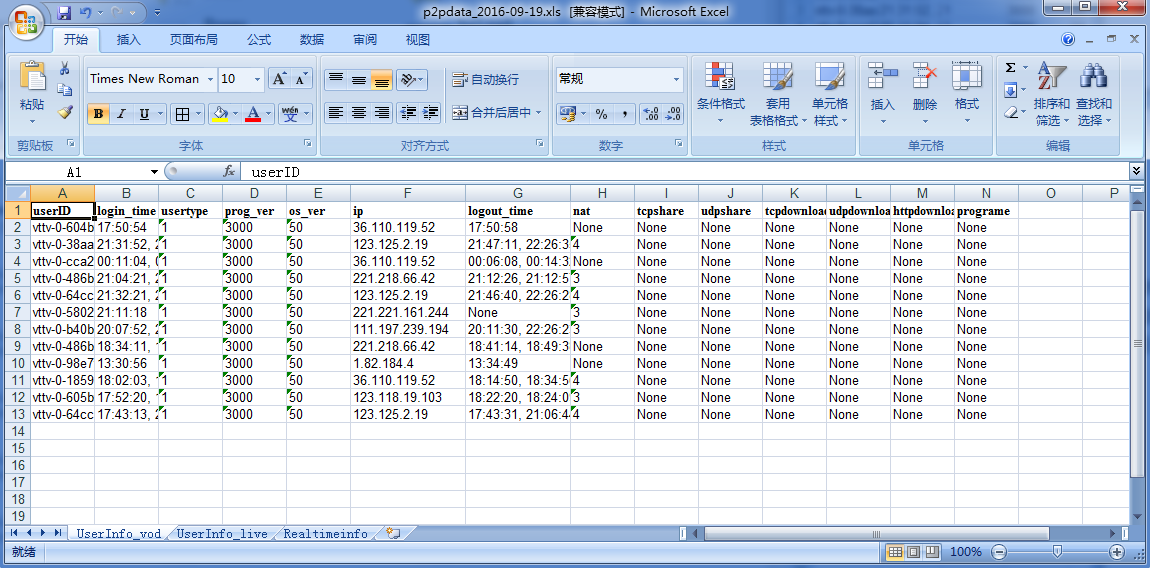
$ sudo pip install xlrd //安装读excel的软件

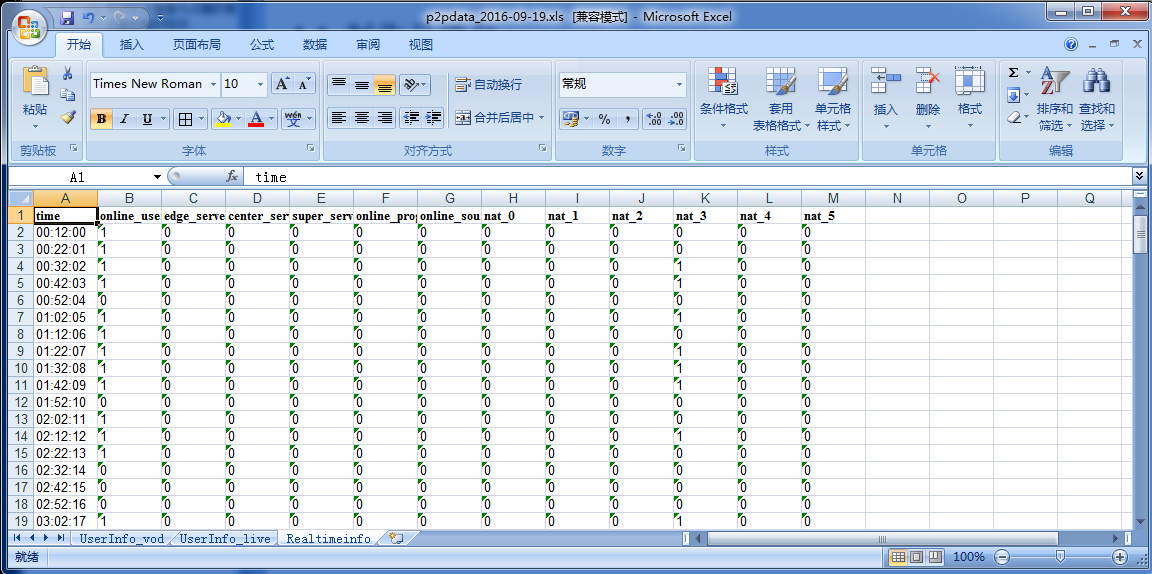
$ sudo pip install xlwt //安装写excel的软件

## 3.2输出结果

### 3.2.1 Excel输出结果示例

p2p\_xxxxxx.xls





表中字段含义如下：

UserInfo Sheet记录的是用户的统计信息，分为点播与直播两个sheet页

userID:用户的ID，以设备的mac命名，类似“vttv-0-38aa3ced2181-4942”，其中‘0’表示是用户边的tracker的ID，“38aa3ced2181”表是mac值，最后四位是第一次使用时分配的随机值

login\_time：记录的是用户登陆tracker的时间

usertype：用户的类型，client，center或server，目前只使用clent类型，即‘0’

prog\_ver：peer的版本类型

os：peer的系统类型

logout\_time：用户退出tracker的时间

tcpshare：用户tcp数据的分享量

udpshare：用户udp数据的分享量

tcpdownload：用户tcp数据的下载量

udpdownload：用户udp数据的下载量

nat：用户的nat类型，0~5

programe：用户观看节目的hash值和开始时间

Realtimeinfo Sheet记录的服务器统计统计的用户的类型

time：统计时间，每10分钟统计一次

online\_users：在线用户数

edge\_servers：edge server在线数（未使用）

center\_servers：center server在线数（未使用）

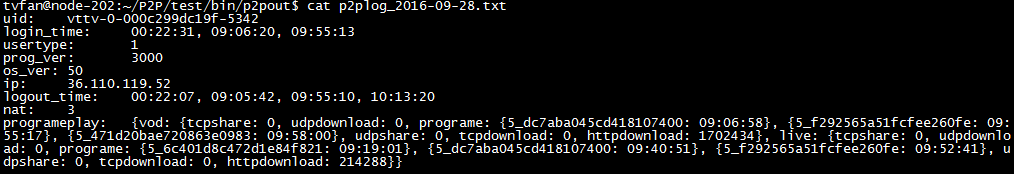
super\_servers：super server在线数（未使用）

online\_progs：当前在观看的节目数

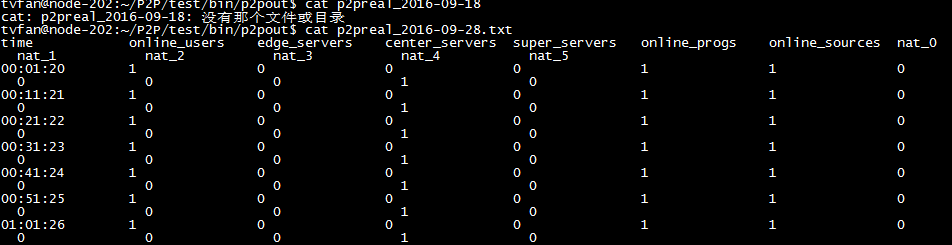
online\_sources：当前观看节目的数据源数，对于点播，若一个用户观看了2个节目，则online\_progs为2，online\_sources为2。online\_sources/online\_progs值越大，即实际用户越多，分享效果即越好

nat\_x：nat类型为x的统计值，x取[0,5]

### 3.2.2 TXT输出结果

p2plog\_xxxxx.txt

p2preal\_xxxxxx.txt



# 4.P2P升级功能

## 4.1 制作升级包

制作升级包的目录“tools/upgrade”，制作流程如下：

1）拷贝新生成的"vkkp2p\_android.e"到目录“tools/upgrade”中；

2）执行命令"./generateUpgrade.sh"，就会生成升级包“vodpeerUpgrade.tar.gz”

3）将升级包拷贝到webserver，即"/server/tomcat/tomcat\_p2p/webapps/ROOT/p2p/update @http://103.244.165.201:4000 "

Note：

1）新生成的"vkkp2p\_android.e"的生成日期要与程序中的“VERSION\_STR”保持一致；

2）升级包的组成，readme.txt和vodpeer.e，readme.txt记录升级程序的信息，vodpeer.e是升级程序。



readme.txt示例

## 4.2升级包的部署

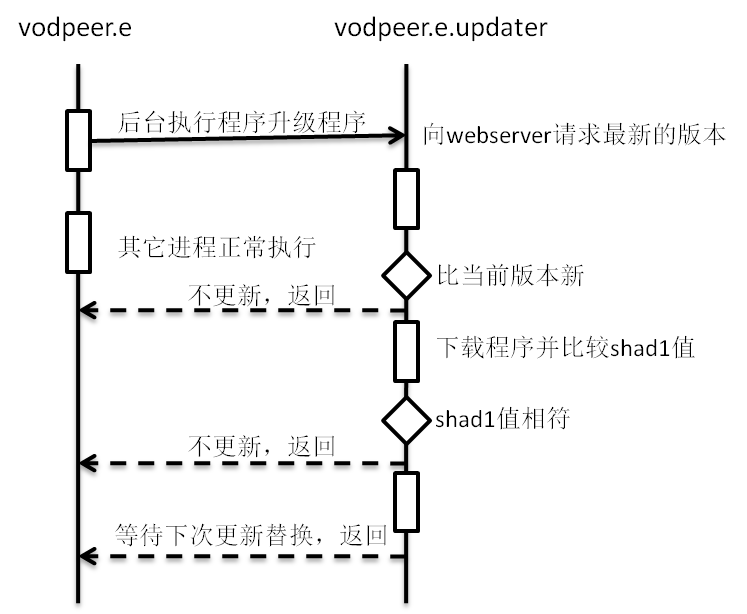
将升级包拷贝到指定目录下，并解压，根据readme.txt信息修改vodpeer\_update.ini文件，文件的格式如下：



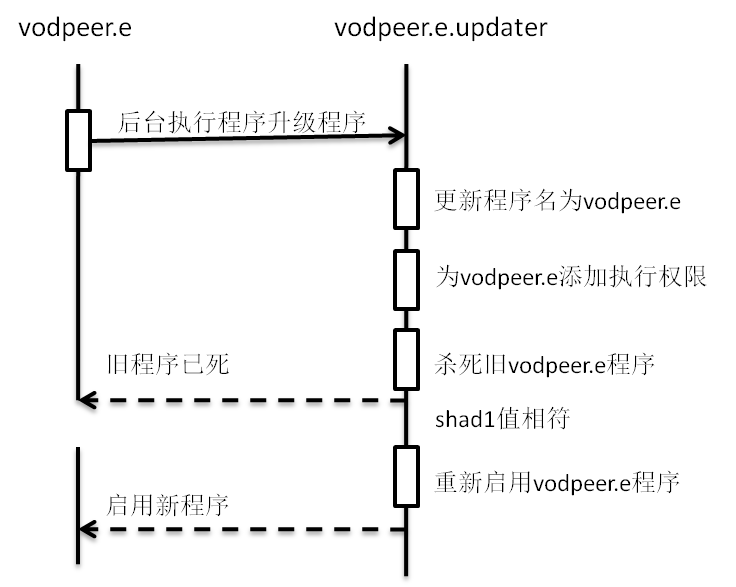
Note：尽量将原程序备份

## 4.3 升级流程

当前目录下无vodpeer.e.newversion情况 ，流程如下：



当前生成vodpeer.e.newversion情况 ，流程如下：



# 5 P2P新开发点展望

## 5.1 tracker的分流

目前tracker的容纳用户10W，即是如果p2p的使用用户为10w，若超过了10w，会影响10w之外的用户的体验，解决些问题分为两个方法：

1）tracker的级联

多搭建几个tracker及一个总的tracker，让几个tracker与总的tracker交互，peer与分散的tracker互动，但这样会有两个问题要考虑：一是分tracker与总tracker的通信，总的tracker要实现所有分tracker的功能并要管理、统筹分tracker，总tracker的负担比较重，并且peer与tracker的通信要“peer->分tracker->总tracker->分tracker->peer”，而之前的是“peer->tracker->peer”，复杂性提高，调用的实时性及效率打折扣；二是peer选择tracker的问题，这个问题在peer的用户分流中提。

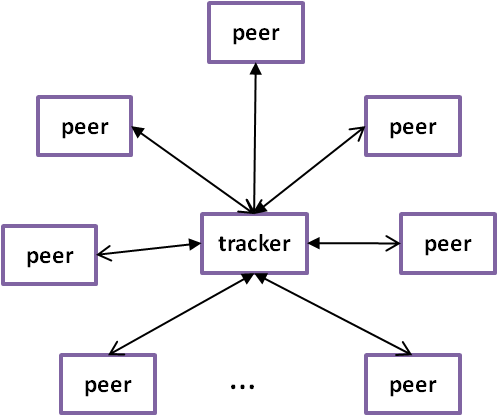
2）peer的用户分流

根据用户的ip不同而连接不同的tracker，即实现不同ip段（区域）用户连接不同的tracker，既实现用户的分流，降低tracker的压力，又与tracker的取源策略相吻合。

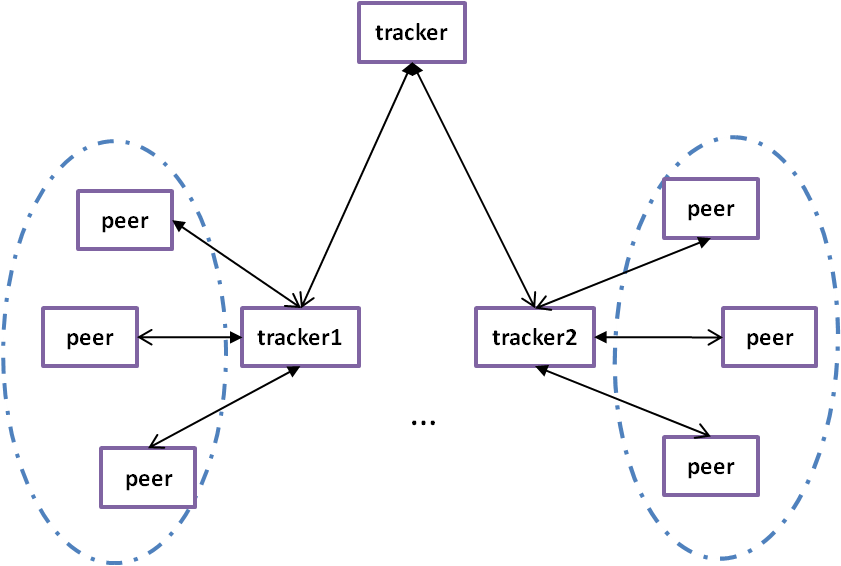
要实现此功能，可这样实现：peer获取自己的外网ip，根据本地的表（不同ip段不同的tracker的对应表），选择相应的tracker相连。这样关键点就在本地的表和获取自己的外网ip

1）获取自己的外网ip，只需要连接一个外网服务器，即可让服务器返回自己的外网ip，这个外网服务器的选择可自己定；

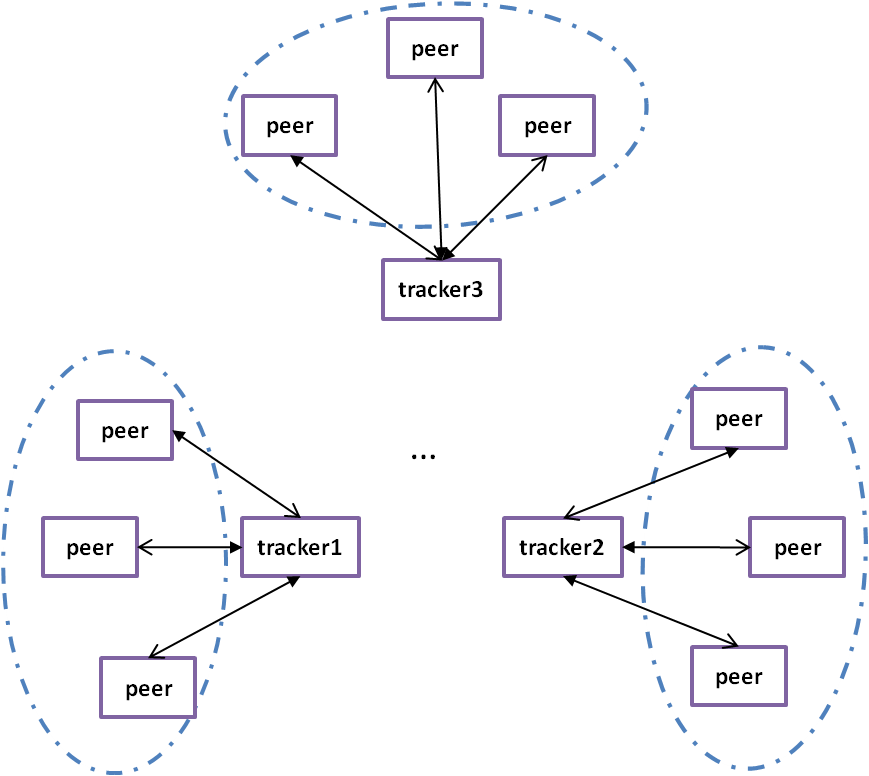
2）本地表，即tracker与ip段映射表，可通过做简单的统计，统计用户的ip分布，再做映射表；



当前tracker



tracker的级联



peer的分流

## 5.2 P2P服务器节点

开发服务器节点，类似超级用户super peer，这些peer的是运行在服务器上，带宽足够高，一直在运行，这样能保证用户永远能取到peer节点，类似最初级的CDN。