# 微信小程序网关设计方案

目录

[微信小程序网关设计与实现 1](#_Toc26098376)

[需求 1](#_Toc26098377)

[应用场景 1](#_Toc26098378)

[小程序仅加入会议 1](#_Toc26098379)

[小程序与一个APP 1](#_Toc26098380)

[小程序与多个APP 1](#_Toc26098381)

[两个小程序 1](#_Toc26098382)

[多个app与多个小程序 1](#_Toc26098383)

[模块 1](#_Toc26098384)

[微信网关结构图 1](#_Toc26098385)

[交互流程 1](#_Toc26098386)

[小程序加入会议流程 1](#_Toc26098387)

[多个小程序请求同一app视频不包括声音流程 1](#_Toc26098388)

[小程序请求视频流程 1](#_Toc26098389)

[接口 1](#_Toc26098390)

[MCS 与callroutegw相关接口 1](#_Toc26098391)

[加入会议 wxjoinconf 1](#_Toc26098392)

[退出会议  wxquitconf 1](#_Toc26098393)

[请求视频 wxreqvideo 1](#_Toc26098394)

[取消视频 wxcancelvideo 1](#_Toc26098395)

[请求视频 wxreqavsync 1](#_Toc26098396)

[mcs 和wxmgw交互 1](#_Toc26098397)

[数据 1](#_Toc26098398)

[测试 1](#_Toc26098399)

# 需求

1.小程序加入会议，其他客户端能看到小程序会议的视频与声音

2.小程序需要其他客户端视频与声音

3.其他客户端能看到小程序会议音视频

4.可以多个小程序加入会议

5.小程序可以获取其他小程序的会议音视频

# 应用场景

小程序加入会议有多种场景

## 小程序仅加入会议



这种场景应用是小程序加入会议后，只需要会议声音，不需要看其他的视频

## 小程序与一个APP



小程序与app加入会议后，小程序需要看到app的视频，同时与需要会议的声音， app也需要看到小程序的视频，同时听到会议的声音

## 小程序与多个APP



小程序与多个app加入会议， 每个app需要会议声音与其他app与小程序的视频，小程序需要会议声音与所有app的视频

## 两个小程序



两个小程序加入会议后，小程序1需要会议声音与小程序2的视频， 小程序2需要会议声音与小程序1的视频

## 多个app与多个小程序



多个小程序与多个app加入会议后， 每个app或小程序需要看到其他所视频以及会议的声音

# 模块

## 微信网关结构图



**微信网关程序**主要功能将小程序推到直播服务器的音视频流，拉过来之后，将音视频解复用，再将音频与视频分别推送给会议系统的mas, 这样小程序就加入会议开会。同时微信网关与会将会议中其他app的音视频拉过来之后，再将复合成rtmp流推到直播服务器，然后小程序将可以拉到其他app的音视频

每个小程序加入同一会议后，将这个会议的声音返回给小程序，这路声音是从会议mas中传送过来，rtpRecver收到之后将放入该小程序的audio Pub与video pub中，如果小程序只需要声音，flv muxer订阅该通道的audio, 转发到直播服务器，小程序再从直播服务器拉流。如果同时小程序需要其他app视频，需要通过mas将app视频传送过来，flv muxer将音频与视频复后成一路rtmp推送到直播服务器。

小程序请求视频有两种情况，一种时请求app的视频需要通过mas传送过来，另外一种时同会议中其他小程序的视频，因为其他小程序视频是通过微信网关拉过来，再转发到mas, 所以其他小程序的视频，在转发mas 需要放到video pub, 如果有其他小程序需要该视频，直接订阅之后，再通过flv muxer复后之后发送到直播服务器

同一会议的所有小程序音视频处理，需要确保在同一个微信网关程序处理，集群环境也需要这样

## 交互流程

### 小程序加入会议流程



### 多个小程序请求同一app视频不包括声音流程



### 小程序请求视频流程



#### 请求普通成员视频:

1只需要视频： 如果成员已经在wxgw, 直接到直播服务器拉流，否则将该成员的会议视频从mas推流到wxgw. 同时推流到直播服务器，再从直播服务拉流

2.需要视频同时需要混音, 首先走只需要视频流程，同时将该成员视频与小程序成员的混音合同一路直播流，推流直播服务器，再拉流

#### 请求小程序成员视频:

1. 只要视频，

方案1:在wxgw检查有没有该小程序rtmp2rtp, 如果没有创建一个，然后解复合video, audio, 然后将video到推直播服务器，

方案2:直在从直播服务器拉小程序参与的rtmp流，包括有AV, 小程序播放时，把A禁用

2.需要混音

wxgw拉流后解复合video, audio, 再将video与小程序混音复合成rtmp流

# 接口

## MCS 与callroutegw相关接口

### 加入会议 wxjoinconf

eg:  
{  
    "id": "3",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",

   "method": "wxjoinconf",  
    "params": {

         "confId": "123" ,

          "crgwid":"53001",                              //对应callroute的编号

    }  
}

response  
{  
     "id": "3",

     "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":"wxjoinconf",  
     "result": {  
         "audiossrc": "503677616",

          "videossrc": "503677616",

         "pushurl":"",       //小程序推音视频流的rtmp url

         "pullurl":"", //小程序推音视频流所对应的拉流地址，

//小程序本身不使用，debug时用到

          "pullaudiourl":"",    //小程序拉会议音频的rtmp url

          "audioip": "192.168.1.123",       //音频mas地址  
          "audiortpport": "12345",        //音频mas rtp媒体端口

          "videoip": "192.168.1.123",       //视频mas地址  
          "videortpport": "12345",             //视频mas rtp媒体端口

     }  
}

### 退出会议  wxquitconf

eg:  
{  
    "id": "3",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  
    "method": "wxquitconf",  
    "params": {

         "confId": "123",

         "crgwid":"53001"                               //callroutegw编号如53001

    }  
}

response  
{  
     "id": "3",

     "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":"wxquitconf",  
     "result": {

     }  
}

### 请求视频 wxreqvideo

eg:  
{  
    "id": "3",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  
    "method": "wxreqvideo",  
    "params": {

         "confId": "123" ,

         "antissrc":"503677616",      //被请求视频的会议成员的ssrc

         "seq":"231ABfu13ad21",     //微信小程序携带的消息序列号,唯一标识一次请求

         "type":"1"                     //请求的类型，1：视频  2：屏幕共享

}  
}

response  
{  
     "id": "3",

     "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":"wxreqvideo",  
     "result": {

         "pullurl":"",

         "seq":"231ABfu13ad21",   //微信小程序携带的消息序列号,唯一标识一次请求

         "type":"1"                  //请求的类型，1：视频  2：屏幕共享

     }  
}

### 取消视频 wxcancelvideo

eg:  
{  
    "id": "3",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  
    "method": "wxcancelvideo",  
    "params": {

         "confId": "123" ,

         "antissrc":"503677616",        //被请求视频的会议成员的ssrc

         "seq":"231ABfu13ad21",      //微信小程序携带的消息序列号,唯一标识一次请求

         "type":"1"                      //请求的类型，1：视频  2：屏幕共享

    }  
}

response  
{  
     "id": "3",

     "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":"wxcancelvideo",  
     "result": {

          "seq":"231ABfu13ad21",         //微信小程序携带的消息序列号,唯一标识一次请求

          "type":"1"                //请求的类型，1：视频  2：屏幕共享

     }  
}

### 请求视频 wxreqavsync

eg:  
{  
    "id": "3",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  
    "method": " wxreqavsync,  
    "params": {

         "confId": "123" ,

         "antissrc":"503677616",      //被请求视频的会议成员的ssrc

         "seq":"231ABfu13ad21",     //微信小程序携带的消息序列号,唯一标识一次请求

"type":"1"                  //请求的类型，1：视频  2：屏幕共享

     }

}  
}

response  
{  
     "id": "3",

     "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":" wxreqavsync ",  
     "result": {

        "seq":"231ABfu13ad21",   //微信小程序携带的消息序列号,唯一标识一次请求

         "type":"1"                  //请求的类型，1：视频  2：屏幕共享

     }  
}

//所有请求如果失败，都使用下面的失败应答：

error message:  
{  
    "id": "1",

    "uuid": "14f5ac8e-5496-11e7-a38b-d93e113210c6",  //用户ID

    "method":"",                                                    //内容为请求的method值  
    "result": {  
         "code": "111",  
         "message": "Invalid format"  
      }  
}

## mcs 和wxmgw交互

struct Port {  
1:i32 audio,  
2:i32 video,  
}  
struct StartPushReq {  
1:string uuid,  
2:string pushUrl,  
3:i32 audioPayloadType,  
4:string audioCodec,  
5:i32 videoPayloadType,  
6:string videoCodec

7:string confId,

8:string reqUuid,

9:string weixin, 与成员是微信小程序成员  
}  
struct StartPushResp {  
1:Port port,  
2:i32 code,  
3:string message,  
}  
struct StopPushReq {  
1:string uuid,

2:string confId  
}  
struct StopPushResp {  
1:i32 code,  
2:string message,  
}  
struct StartPullReq{  
1:string uuid,  
2:string pullUrl,  
3:i64 ssrc,  
4:string masIp,  
5:i32 audioPort,  
6:i32 videoPort,  
7:i32 audioPayloadType,  
8:string audioCodec,  
9:i32 videoPayloadType,  
10:string VideoCodec,  
11:string pushAudioUrl,

12:string confId,  
}  
struct StartPullResp {  
1:i32 code,  
2:string message,  
}  
struct StopPullReq {  
1:string uuid,

2:string confId,  
}  
struct StopPullResp {  
1:i32 code,  
2:string message  
}

struct AvSyncReq {  
1:string uuid,

2:string confId,

3:string syncUuid  
}  
struct AvSyncResp {  
1:i32 code,  
2:string message  
}

# 数据

Mcs会议成员

普通成员: 小程序视频直播频道，推流与拉流地址

小程序成员: 混音直播频道，推流与拉流地址

Wxgw重要数据

Rtp2rtmpMap

Key: uuid

视频pubsub

音频pubsub

Flvmuxer flv复合器

Rtmp2rtpMap

Key: uuid 小程序uuid

Flvdemuxer: flv解复合器

视频pubsub

音频pubsub

小程序会议房间:

小程序参会者 rtmp2rtp

小程序参会者 rtp2rtmp

小程序请求者 rtp2rtmp

# 测试