标识：XXX-XX-XX-001

版本：001

XXXXX

项目技术文件

单组播协同直播系统

软件详细设计说明

承 担 部 门：北京邮电大学

完 成 日 期：2023-6-28

文档修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修改内容描述 | 修改人 | 日期 | 备注 |
| 001 | 初次提交 | 徐祖云 | 2023-6-28 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 次

[1 引言 1](#_Toc301)

[1.1 编写目的 1](#_Toc14068)

[1.2 背景 1](#_Toc28283)

[1.3 术语及缩略语 1](#_Toc6273)

[1.4 参考资料 1](#_Toc22292)

[2 程序系统的结构 1](#_Toc6683)

[3 程序1（标识）设计说明 1](#_Toc30390)

[3.1 程序描述 2](#_Toc18946)

[3.2 功能 2](#_Toc17972)

[3.3 性能 2](#_Toc6921)

[3.4 输入项 2](#_Toc32060)

[3.5 输出项 2](#_Toc15809)

[3.6 用户界面 2](#_Toc27463)

[3.7 算法 2](#_Toc26663)

[3.8 流程逻辑 2](#_Toc10532)

[3.9 接口 2](#_Toc10120)

[3.10 存储分配 2](#_Toc28602)

[3.11 限制条件 2](#_Toc30391)

[3.12 尚未解决的问题 2](#_Toc1525)

[4 程序2（标识）设计说明 3](#_Toc25293)

# 引言

## 编写目的

随着互联网的高速发展以及终端软硬件计算能力不断提高,全景视频直播相关业务迅速扩张。全景视频作为VR的呈现方式之一,因其360度视角、高分辨率的特点在各个领域得到应用。在带来全新视觉体验的同时也伴随着视频数据量的巨大提升，为了满足全景视频的观看需求，更好的为客户提供服务，现结合原有直播系统的优点和特点，开发一套采用 单组播协同、quic丢包重传的直播系统。本系统从传输层予以优化，使整个直播系统观看稳定性得到较大提高，通过重新优化内部结构，使系统的可扩充性得到极大提高。

本说明书给出直播系统的设计说明，包括最终实现的系统各个程序模块的结构、算法等。

**目的在于:**

为编码人员提供依据；

为修改、维护提供条件；

项目负责人将按计划书的要求布置和控制开发工作全过程；

项目质量保证组将按此计划书做阶段性和总结性的质量验证和确认；

**本说明书的预期读者包括:**

项目开发人员，特别是编码人员;

软件维护人员;

技术管理人员;

执行软件质量保证计划的专门人员；

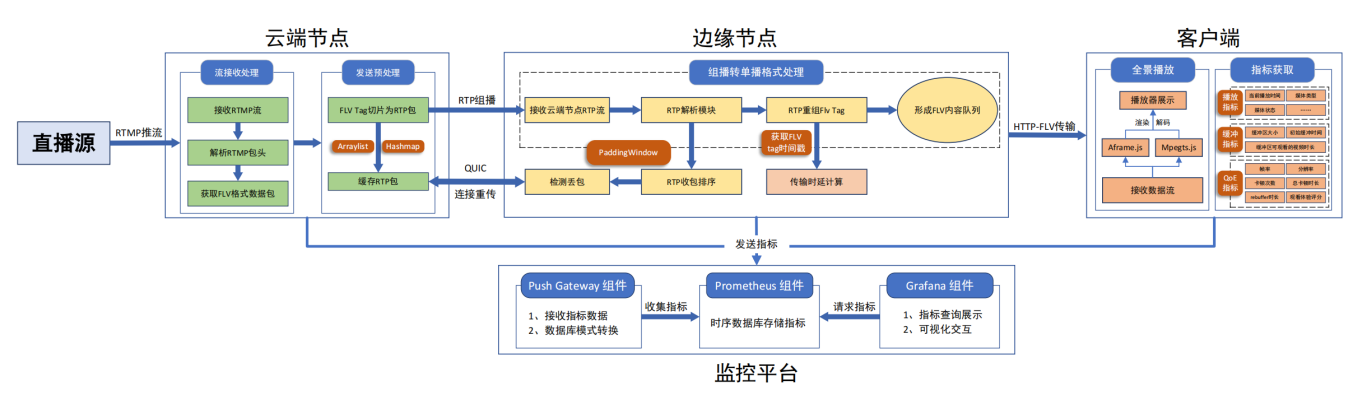
参与本项目开发进程各阶段验证、确认以及负责为最后项目验收、鉴定提供相应报告的有关人员；

合作各方有关部门的复杂人;项目负责人和全体参加人员；

## 背景

本项目由北京邮电大学计算机学院和中国电信广州研究院共同提出，由北京邮电大学网络体系结构研究院硕士研究生开发。

本直播系统由三大部分组成：用于接收直播源推流的云端节点，接收云端节点数据并向用户提供服务的边缘节点，用于播放全景视频并收集性能指标的播放器。有关本系统与其他系统的相关联系如下：



## 术语及缩略语

列出本文件中用到的专门术语的定义和外文首字母组词的原词组：

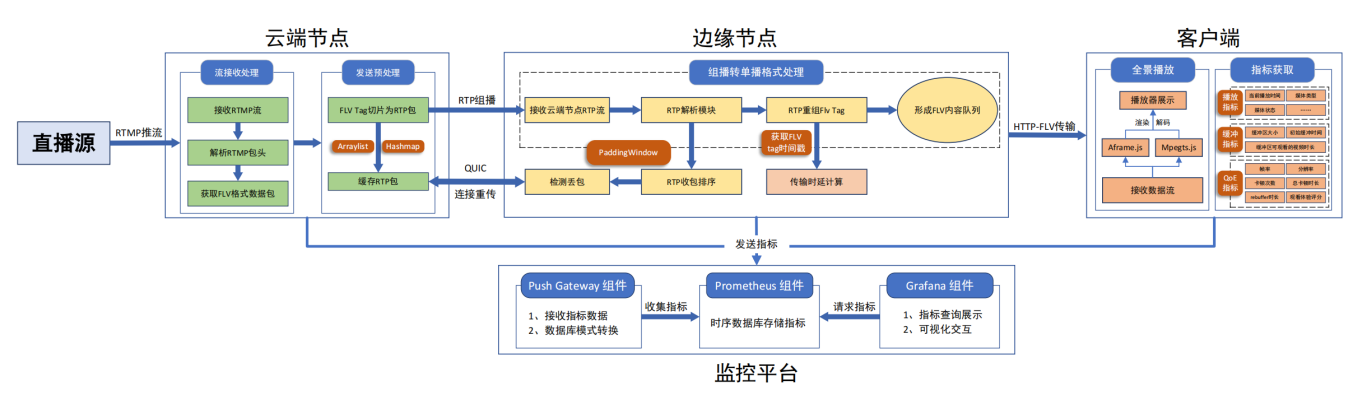
表 1‑1 本文档使用的术语或缩略语一览表

| 序号 | 术语或缩略语 | 解释 | 备注 |
| --- | --- | --- | --- |
|  | 本系统 | 基于quic重传的单组播协同直播系统 |  |

表 2‑1 参考资料一览表

| 序号 | 文档标识 | 文档名称 | 版本号 | 备注 |
| --- | --- | --- | --- | --- |
| 1. | 《软件工程——原理.方法与应用》-001 | 《软件工程——原理.方法与应用》 | 001 | 史济民 等 高等教育出版社2002.2 |
| 2. | Adobe’s Real Time Messaging Protocol-001 | Adobe’s Real Time Messaging Protocol | 001 | https://github.com/melpon/rfc/blob/master/rtmp.md |
| 3. | RTP: A Transport Protocol for Real-Time Applications-001 | RTP: A Transport Protocol for Real-Time Applications | 001 | https://www.rfc-editor.org/rfc/rfc3550.html |
| 4. | QUIC: A UDP-Based Multiplexed and Secure Transport-001 | QUIC: A UDP-Based Multiplexed and Secure Transport | 001 | https://datatracker.ietf.org/doc/html/rfc9000 |
| 5. | Adobe Flash Video File Format Specification  Version 10.1-001 | Adobe Flash Video File Format Specification  Version 10.1文档 | 001 | http://download.macromedia.com/f4v/video\_file\_format\_spec\_v10\_1.pdf |
| 6. | HTTP Live Streaming-001 | HTTP Live Streaming文档 | 001 | https://www.rfc-editor.org/rfc/pdfrfc/rfc8216.txt.pdf |

# 程序系统的结构



# rtmp流转flv功能设计说明

## 程序描述

该程序监听rtmp流，将从rtmp流中得到的rtmp消息解析出类型、数据、时间戳等信息，区分不同的流，最后封装为flv Tag

## 功能

该程序应具有将rtmp消息转换为flv tag数据，并能够区分不同的rtmp流。

## 性能

该程序需要尽快的执行，从而保证媒体数据实时的交付到边缘节点，从而减少直播的时延；需要对不同种类的rtmp消息数据都能够执行，并保证数据的准确性；需要不断的循环执行直至所有的流都关闭。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输入方式 | 数量和频度 | 来源 |
| --- | --- | --- | --- | --- | --- | --- |
| rtmp消息 | message | \*RtmpStream.Message | - | 消息监听 | 无限 | 用户推流 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Flv Tag数据 | flv\_tag | []byte | - | 内部传递 | 无限 | Rtp切片发送 |

# Flv切片发送程序设计说明

从本章开始，逐个地给出各个层次中的每个程序的设计考虑。以下给出的提纲是针对一般情况的。对于一个具体的模块，尤其是层次比较低的模块或子程序，其很多条目的内容往往与它所隶属的上一层模块的对应条目的内容相同，在这种情况下，只要简单地说明这一点即可。

## 程序描述

该程序将从Rtmp流中得到的Flv Tag通过Rtp单组播的方式发送到所有的边缘节点，对长度超过Rtp最大载荷的Tag分片处理、标记后发送。

## 功能

该程序应具有将Flv数据正确切片并标记、进行Rtp封装（设置ssrc和marker等）、通过Udp发送到指定单播和组播地址的功能。

## 性能

该程序需要尽快的执行，从而保证媒体数据实时的交付到边缘节点，从而减少直播的时延；需要对不同大小的Flv数据都能够执行，并保证数据的准确性；需要不断的循环执行直至所有的流都关闭。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输入方式 | 数量和频度 | 来源 |
| --- | --- | --- | --- | --- | --- | --- |
| Flv Tag数据 | flv\_tag | []byte | - | 内部传递 | 无限 | 用户推流 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp数据包 | rp | \*rtp.DataPacket | 任何 | 内部传递 | 无限 | rtp缓存和udp socket |

## 算法

详细说明本程序所选用的算法，具体的计算公式和计算步骤。

|  |
| --- |
| **算法1:** flvTag切片发送 |
| **输入:** 字节数组flv\_tag |
| **while** next flv\_tag **do**  **if** len(flv\_tag) <= MAX\_RTP\_PAYLOAD\_LEN **then**  rp ← Rtp包(initialization)  rp.marker ← true  rp.payload ← flv\_tag  发送rp  **else**  slice\_num ← len(flv\_tag)%MAX\_RTP\_PAYLOAD\_LEN + 1  **for** i=0 to slice\_num **do**  rp ← Rtp包(initialization)  **if** !last slice **then**  rp.marker ← false  **else**  rp.marker ← true  **end if**  rp.payload ← flv\_tag[i\*MAX\_RTP\_PAYLOAD\_LEN,  (i+1)\*MAX\_RTP\_PAYLOAD\_LEN]  发送rp  **end for**  **end if**  **end while** |

## 存储分配

在构造Rtp数据包的过程中会产生临时的数据包，一般在10个以内，用于减少频繁创建和销毁Rtp数据包造成的性能开销，一个Rtp包通常在100到1200个字节。

## 限制条件

本程序只会在有直播推流的时候才会执行。

## 尚未解决的问题

# Rtp包缓存程序设计说明

## 程序描述

该程序将Rtp发送的包放入通过arraylist/hashmap建立缓存中，为后期quic重传服务提供支持。

## 功能

该程序应将Rtp数据包顺序的放入队列中，设置最大长度，将新的包从队头送入队列，从队尾删除旧的包，并能够快速的根据流的ssrc和指定的seq查找到对应的Rtp包。

## 性能

该程序需要尽快的执行，从而保证重传数据实时的交付到边缘节点，从而减少直播的时延；需要对不同的ssrc和seq都能够正确执行，并保证数据的准确性；需要不断的循环执行直至所有的流都关闭。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输入方式 | 数量和频度 | 来源 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp数据包 | rp | \*rtp.DataPacket | 任何 | 内部传递 | 无限 | 程序创建 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp队列 | rtpQueue | \*rtpQueue | 任何 | 内部传递 | 无限 | 等待quic服务访问 |

## 算法



## 存储分配

在构造Rtp队列时根据配置文件指定缓存的Rtp包数量（5000），并在队列中长期维持这个数量的Rtp包，一个Rtp包通常在100到1200个字节。

## 限制条件

本程序只会在有直播推流的时候才会执行写入缓存，在收到quic重传请求时执行查找缓存的操作。

## 尚未解决的问题

# Rtp收包排序与丢失重传设计说明

## 程序描述

该程序将通过Udp收到的Rtp包（可能存在乱序和丢失）重新组织成有序的数据包队列，对丢失的包通过quic重传获取。

## 功能

该程序应将Rtp数据包顺序的放入队列中，根据不同的ssrc送入不同的队列，在队列中：

1>边缘节点收到的包可能是丢失的、乱序的，需要将这些包组织起来有序的交付给Rtp解析模块，并检测丢失的包，等待一段时间后发送重传获取。

2>设计一个队列，始终只记录队列头的序号FirstSeq，每当有新包到来时根据FirstSeq计算应当在队列中的相对位置并放入，这样得到一个顺序排列、一些位置为空的rtp队列，队列中越靠后的rtp包越新。

3>设置一个大小为PaddingWindowSize的窗口，将队列尾部往前窗口大小内的区域为等待区（较新的包），窗口大小外的区域为重传区 （较旧的包）。

4>每次从队列头依次取rtp包，对于重传区的包，如果不存在则直接发送重传请求，直至将重传区的包都取到。

5>接着从等待区依次取rtp包，存在则取出，若遇到空则等待队列填满等待区。

6>这样确保每个没收到的包都有一定的等待时间，防止其仅仅是乱序到达而非丢失，并减少了不必要的时延。

## 性能

该程序需要尽快的执行，从而保证媒体数据实时的交付到客户端，从而减少直播的时延；需要对不同的ssrc和seq都能够正确执行，并保证数据的准确性；需要不断的循环执行直至所有的流都关闭；需要对各种极端情况（无法重传获得丢失的包、包数据发生错误）都有一定的处理机制。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输入方式 | 数量和频度 | 来源 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp数据包 | rp | \*rtp.DataPacket | 任何 | 内部传递 | 无限 | udp层解析 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp队列 | rtpQueue | \*Queue | 任何 | 内部传递 | 无限 | 等待flv解析模块访问 |

## 算法

|  |
| --- |
| **算法2:** Rtp排序与重传 |
| **输入:** 新的Rtp数据包rp  **输出:** 有序的Rtp数据包pkt |
| **while** next rp **do**  firstSeq ← rtpQueue[0].Seq;  rtpQueue[rp.Seq - firstSeq] ← rp;  **for** len(rtpQueue) > 等待区大小 **do**  p ← rtpQueue[0];  **if** p为空 **then**  重传p;  **end if**  输出 p  **for** rtpQueue[0]不为空 **do**  p ← rtpQueue[0];  输出p  **end for**  **end while** |



## 存储分配

在构造Rtp队列时根据配置文件指定窗口大小的Rtp包数量（100），程序正常运行时队列会低于这个窗口长度（具体取决于乱序的情况和丢失包重传的速度），一个Rtp包通常在100到1200个字节。

## 限制条件

本程序只会在有收到云端节点Rtp流数据时才会执行。在边缘节点性能不够时（过多的推流数量和过大的码率）无法正常完成，会放弃quic重传并主动丢包以确保系统不会崩溃。

## 尚未解决的问题

# Rtp重组Flv Tag设计说明

## 程序描述

该程序将通过Rtp队列中取得的有序Rtp包重新组合解析成有效的Flv Tag数据包，交付http-flv模块向客户端提供访问。

## 功能

该程序应将Rtp数据包根据marker位重组为切片前的数据，获得完整的flv内容队列，送入http-flv/hls服务模块中。

1. 对于上层rtp队列交付的有序Rtp包，需要将切片的flvTag重组为完整的Tag，并送入flv内容队列。

2>设置一个数据结构`FlvRecord`作为tag缓存，根据第一个切片中存放的TagSize信息，创建相应大小的buffer，将接下来收到的切片依次顺序放入buffer中，直至遇到marker=1。

3>对于一些复杂情形的健壮性处理，如果某个包重传也无法获取，则丢弃该包所属的flvTag缓存，并设置jumpToNextHead为true丢弃接下来的所有属于该Tag的Rtp包，跳到下一个flvTag开始解析（会造成图像花屏）。

## 性能

该程序需要尽快的执行，从而保证媒体数据实时的交付到客户端，从而减少直播的时延；需要对不同的ssrc都能够对应到唯一的path（客户端访问地址），并保证数据的准确性；需要不断的循环执行直至所有的流都关闭；需要对各种极端情况（flvTag解析失败，遭到破坏等）都有一定的处理机制。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输入方式 | 数量和频度 | 来源 |
| --- | --- | --- | --- | --- | --- | --- |
| Rtp数据包 | rp | \*rtp.DataPacket | 任何 | 内部传递 | 无限 | Rtp队列 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Flv数据包 | FlvTag | []byte | 任何 | 内部传递 | 无限 | 交付http-flv和hls模块 |

## 算法

|  |
| --- |
| **算法3:** Rtp重组flvTag |
| **输入:** 有序的Rtp数据包rp  **输出:** flv Tag字节数组flvTag |
| **while** next rp **do**  marker ← rp.marker;  pos ← 0  **if** marker为0 **then**  **if** rp为初始分片 **then**  payload ← rp.payload;  TagSize ← 解析playload;  flvTag ← 长度TagSize的字节数组(initialize);  flvTag[0:len(payload)] ← payload;  pos ← len(payload)  **else**  flvTag[pos:len(payload)] ← payload;  pos ← pos + len(payload)  **end if**  **else**  flvTag[pos:len(payload)] ← payload;  pos ← 0  **return** flvTag  **end if**  **end while** |

## 存储分配

在构造flv队列时会对每个客户端创建相应的http响应、管道、tag队列等，数量取决于访问流的客户端数量，每个flv tag的大小取决于tag的种类（音频或关键帧视频和非关键帧视频）和媒体流的清晰度，音频tag大小通常为几百字节。720p视频关键帧的tag大小通常为10K～100K字节，非关键帧的tag大小通常为2000～10000字节。

## 限制条件

本程序只会在有收到云端节点Rtp流数据时才会执行。在边缘节点性能不够时（过多的客户端数量）无法正常完成，会主动丢包以确保系统不会崩溃。

## 尚未解决的问题

# http-flv服务设计说明

## 程序描述

该程序维护每条流的初始段信息和内容队列，并监听http端口，收到客户端的访问请求后向其源源不断的提供对应的http-flv格式的媒体流。

## 功能

程序在收到每条新流时保存其初始段信息，同时开启http服务，收到客户端请求后解析path信息确定相应的流ssrc，找到该ssrc解析Rtp队列得到FlvTag数据，先返回flv头和预先保存的流初始段，再将音视频数据其以flv流的格式源源不断的写入httpResponser。

## 性能

该程序需要尽快的执行，从而保证媒体数据实时的交付到边缘节点，从而减少直播的时延；存在大量用户时需要对每个客户端都能够执行，并保证数据的准确性；需要不断的循环执行直至所有的流都关闭。

## 输入项

给出对每一个输入项的特性，包括名称、标识、数据的类型和格式、数据值的有效范围、输入的方式。数量和频度、输入媒体、输入数据的来源和安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Flv Tag数据 | flv\_tag | []byte | - | 内部传递 | 无限 | flv服务队列 |
| 初始段 | initialSegment | SegmentPacket | flv流序列信息 | 内部传递 | 流数量 | 流信息缓存 |

## 输出项

给出对每一个输出项的特性，包括名称、标识、数据的类型和格式，数据值的有效范围，输出的形式、数量和频度，输出媒体、对输出图形及符号的说明、安全保密条件等等。

| 名称 | 标识 | 数据类型 | 有效范围 | 输出方式 | 数量和频度 | 去向 |
| --- | --- | --- | --- | --- | --- | --- |
| Flv Tag数据 | flv\_tag | []byte | - | 内部传递 | 无限 | 用户http响应 |
| 初始段 | initialSegment | SegmentPacket | flv流序列信息 | 内部传递 | 流数量 | 用户http响应 |

## 存储分配

流初始段会保存在缓存中，每条流有两个（音频和视频），初始段的大小在100字节以内。在构造flv队列时会对每个客户端创建相应的http响应、管道、tag队列等，数量取决于访问流的客户端数量，每个flv tag的大小取决于tag的种类（音频或关键帧视频和非关键帧视频）和媒体流的清晰度，音频tag大小通常为几百字节。720p视频关键帧的tag大小通常为10K～100K字节，非关键帧的tag大小通常为2000～10000字节。

## 限制条件

本程序只会在有收到云端节点Rtp流数据时才会执行。在边缘节点性能不够时（过多的客户端数量）无法正常完成，会主动丢包以确保系统不会崩溃。