f d d标识：XXX-XX-YH-001

版本：001

XXXX

项目技术文件

XX系统

软件用户手册

承 担 部 门：

完 成 日 期：

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 | 2023-6-29 | 缺少监控运行说明 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 次

[1 引言 1](#_Toc24833)

[1.1 编写目的 1](#_Toc7520)

[1.2 背景 1](#_Toc9481)

[1.3 术语及缩略语 1](#_Toc1195)

[1.4 参考资料 1](#_Toc15137)

[2 软件综述 1](#_Toc11296)

[2.1 软件的特点 1](#_Toc7856)

[2.2 软件运行环境 1](#_Toc11849)

[2.3 软件的基本组成和操作概述 2](#_Toc25045)

[3 软件安装与初始化 2](#_Toc6044)

[4 使用指南 2](#_Toc24247)

[4.1 XXXX功能 2](#_Toc774)

[4.2 XXXX功能 2](#_Toc20584)

[4.3 出错处理 3](#_Toc962)

[5 附录 3](#_Toc30691)

# 引言

## 编写目的

该文档的目的是描述本直播系统项目的用户使用说明，其主要内容包括:

1>运行环境

2>安装与配置

3>操作说明

本文档的预期的读者是:

1>项目开发人员，特别是编码人员;

2>软件维护人员;

3>技术管理人员;

4>执行软件质量保证计划的专门人员；

5>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；

6>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

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，根据 Zink Michael 等人 的调研结果，传输一个4k的360°全景视频到客户端并允许用户全方位的观看所需要的数据率是400Mb/s，是传统平面高清视频的10倍以上，因此在不同网络状况下尽可能保障视频质量、提高用户体验质量成为全景视频在线传输面临的巨大挑战。而传统的网络架构在海量访问下存在网络拥塞、网络丢包和延时响应等问题，这严重影响了用户观看全景视频的体验。在此背景下，内容分发网络技术凭借代理访问、边缘缓存、负载分担、就近访问等特点，具备提升响应速度、提高系统稳定性、降低主链路负载压力的效果，有广泛的应用前景。我们提出了一种单组播协同传输、嵌入QUIC重传机制的CDN直播网络架构，以探索未来直播网络发展的新方向。

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。

## 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 |  |

## 参考资料

列出需要使用的参考资料，如：

1. 本项目的经核准的计划任务书或合同、上级机关的批文；
2. 属于本项目的其他已发表的文件；
3. 本文件中各处引用的文件、资料、包括所要用到的软件开发标准。 列出这些文件资料的标题、文件编号、发表日期和出版单位，说明能够得到这些文件资料的来源。

| 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 |

表 1‑2 参考资料一览表

# 软件综述

## 软件的特点

基于quic重传的单组播协同直播系统是一个使用go语言编写的高性能流媒体服务器，使用了Rtmp、Rtp、Http-flv等传输协议，运行在linux/windows系统上，主要由云端、边缘、客户端三部分组成，推流者通过将视频流推送到云端服务器，云端通过单组播协同方式发送至边缘服务器，边缘服务器对丢失的包通过quic重传获取并提供http-flv服务；客户端通过就近向边缘服务器获取媒体流，从而获得稳定的观看体验。

通过这个系统为减少了直播服务器传输过程中的网络带宽消耗，可以大大减轻传输网络的拥塞状况，并保证媒体数据传输的完整性，提高用户的观看体验。本系统在设计过程中也充分注重了复用技术的运用。提高系统的灵活性、可扩充性和可维护性。

## 软件运行环境

### 硬件环境

客户端：双核cpu 3.0G及以上，4G内存；安装google chrome浏览器，最好拥有独立显卡。

服务器配置如下:

表 2‑1 服务器配置一览表

| **序号** | **硬件项名称** | **硬件要求** | **数量** | **备注** |
| --- | --- | --- | --- | --- |
|  | 云端服务器 | 8核心CPU，32G内存， 集成双千兆以太网口 ，双冗余电源 | 1 |  |
|  | 边缘服务器 | 8核心CPU，32G内存， 集成双千兆以太网口 ，双冗余电源 | 2 | 可以是大于1的任意数量 |

### 软件环境

表 2‑2 软件环境一览表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 软件需求名称/标识 | 具体要求 | 优先级 | 备注 |
|  | 云端服务器 | supervisor、go1.18+、win/mac/linux | 高 | 用于进程常驻后台 |
|  | 边缘服务器 | supervisor、go1.18+、win/mac/linux | 高 | 用于进程常驻后台 |

## 软件的基本组成和操作概述

系统由云端-边缘-客户端三部分组成，在云端服务器接受直播/点播源的rtmp全景视频流，并转换为flv格式视频流，通过RTP组播切片发送至边缘服务器同时缓存一定数量的数据包；边缘节点主要由RTP缓存模块、flv解析模块和http-flv服务模块组成，通过对数据包的排序和重传以应对复杂的网络环境；客户端通过就近向边缘服务器获取媒体流，减轻云端服务器的负载，获得稳定的观看体验。

# 软件安装与初始化

**3.1云端节点安装**

**方式一 —— 下载编译好的可执行文件**

**[Releases](https://github.com/NOMADxzy/Rtp_Http_Flv/releases)**

**方式二 —— 从源码编译**

1. 由于依赖了**net/rtp**，所以需首先编译[GoRtp](https://github.com/wernerd/GoRTP)库到当前go环境中，方法如下：

git clone https://github.com/wernerd/GoRTP

复制 **rtp**到 go根目录**/src/net**下 ，然后执行：

go build net/rtp

go install net/rtp

找到**src/net/rtp/sessionlocal.go** ，修改 **maxNumberOutStreams = 10000**（增大最大出流数量，默认值只有5条）

2. cd Rtmp\_Rtp\_Flv && go get

3. cd server && go build -o cloud

**3.2边缘节点安装**

**方式一 —— 下载编译好的可执行文件**

**[Releases](https://github.com/NOMADxzy/Rtp_Http_Flv/releases)**

**方式二 —— 从源码编译**

1. cd Rtp\_Http\_Flv

2. 安装依赖go get

3. 编译go build -o edge

4. 若要开启https，需生成相关证书密钥，否则将以http模式服务，具体命令如下：

- mkdir certs

- cd certs

- openssl genrsa -out server.key 1024 [生成服务器私钥]

- openssl req -new -key server.key -out server.csr [根据私钥和输入的信息(域名)生成证书请求文件]

- openssl x509 -req -in server.csr -out server.crt -signkey server.key -days 3650 [用私钥和证书请求文件生成证书]

**3.3客户端安装**

git clone https://gitee.com/vesper0407/my-mpegtsjs-lesson

# 目标分支不是master

cd my-mpegtsjs-lesson && git checkout feat/new-ui-dev

# 安装依赖 mpegts.js自带的

npm install

# 可修改源码之后重新编译 不建议这么做 耗时 可以在原型上扩展 参考mpegtsjs-metric的实现

npm run build:prod

# 运行之后会在根目录dist下生成打包好的文件

# 全景播放器运行

cd ./demo/panorama\_live

# 依赖安装

npm install

# 编译打包

npm run build

# 根据index.html中的注释 下载js/css到本地并按路径引入

# 目前可以直接使用dist下的静态资源

wget ...

# 开启本地服务测试 浏览器打开

npm run client

# 桌面应用程序运行(optional)

npm run start-electron

注意区分**develop**和**prod**两种环境，prod需要编译的bundle.js，保证各浏览器的兼容性

# 使用指南

由于云端节点启动后会将自身信息发送到边缘节点，以减少边缘节点的配置操作，所以最好按照如下顺序启动：边缘、云端、客户端

## 边缘节点启动

1. 开发环境运行

./edge [-udp\_addr 239.0.0.1:5222] [-disable\_quic]

可通过命令行设置参数，也可以通过配置文件设置(推荐)，(优先级 flag命令行 > config.yaml配置文件 > default默认配置)，详见3>

1. 生产环境运行

推荐使用[supervisor](https://zhuanlan.zhihu.com/p/147305277)

启动supervisor supervisord -c /etc/supeivisord.conf

在supervisord.d中新建子进程配置**edge.conf**如下:

|  |
| --- |
| [program:edge]  environment=PATH="$PATH"  directory=/root/work  command=/root/work/edge\_linux\_v4.0.3  autorestart=true  autostart=true  stderr\_logfile=/root/work/edge.err.log  stderr\_logfile\_maxbytes=100MB  stderr\_logfile\_backups=20  stderr\_capture\_maxbytes=10MB  stdout\_logfile=/root/work/edge.out.log  stdout\_logfile\_maxbytes=100MB  stdout\_logfile\_backups=10  stdout\_capture\_maxbytes=4MB  startsecs=5 |

更新配置supervisorctl update

遇到失败请重启supervisor supervisorctl reoload

1. 参数配置

Config.yaml

|  |
| --- |
| ./edge -h  Usage of ./main:  -udp\_addr string 监听udp的端口(:5222)#单播  (239.0.0.1:5222)#组播  -httpflv\_addr string 提供httpflv服务的地址(:7001)  -disable\_quic bool 是否停用quic重传(false)  -padding\_size int rtp队列的缓冲长度(300)  -queue\_chan\_size int 流的写入写出缓冲长度(100)  -record\_dir string 录制文件的存放目录(./record)  -pack\_loss float64 模拟丢包率(0.002)  -enable\_hls bool 开启hls服务(true)  -hls\_addr string hls服务地址(":7002")  -enable\_record bool 启动直播录制(false )  -cert\_file: string https证书(certs/server.crt)  -key\_file: string https公钥(certs/server.key)  -log\_level: string 日志等级(trace/debug/info/error)  -enable\_log\_file: bool 启用日志文件(false)  -protect bool 保护模式,出现大面积连续丢包时会放弃重传这些包,跳到下个有效包(true) |

## 云端节点启动

1. 开发环境运行

直接命令行启动./cloudserver 将会开启服务并监听`1935`、`4242`、`8090`端口（默认配置）

(无配置文件时、使用默认配置并在同目录下生成配置文件，主要配置client\_addr\_list指定边缘节点地址，也可以修改配置文件中的其他参数自定义运行，详细内容见3>)

1. 生产环境运行

推荐使用[supervisor](https://zhuanlan.zhihu.com/p/147305277)

启动supervisor supervisord -c /etc/supeivisord.conf

在supervisord.d中新建子进程配置cloud.conf如下:

|  |
| --- |
| [program:cloud]  environment=PATH="$PATH"  directory=/root/work  command=/root/work/cloud\_linux\_v4.0.3  autorestart=true  autostart=true  stderr\_logfile=/root/work/cloud.err.log  stderr\_logfile\_maxbytes=100MB  stderr\_logfile\_backups=20  stderr\_capture\_maxbytes=10MB  stdout\_logfile=/root/work/cloud.out.log  stdout\_logfile\_maxbytes=100MB  stdout\_logfile\_backups=10  stdout\_capture\_maxbytes=4MB  startsecs=5 |

更新配置supervisorctl update

遇到失败请重启supervisor supervisorctl reoload

1. 参数配置

config.yaml

|  |
| --- |
| rtp\_cache\_size: 5000 #云端节点缓存的rtp包数量，设置过小会影响重传  quic\_addr: :4242 #quic服务的监听地址  client\_addr\_list: #边缘节点的地址，会向这些地址发送数据  - 239.0.0.1:5222  - 127.0.0.1:5224  enable\_record: false #是否录制直播，开启时每条流都会产生相应的flv录制文件  rtp\_port: 5220 #rtp发送端口  rtmp\_addr: :1935 #rtmp监听端口  api\_addr: :8090 #http监听端口，主要向边缘提供流的名称、时间、数据量等信息  debug: false #为true时不向边缘发rtp数据，用于调试  log\_level: debug #日志等级，trace/debug/info/error  enable\_log\_file: true #启用日志文件，不启用时仅在命令行输出过程信息 |

## 推流

云端和边缘启动后，可以开始直播推流，测试阶段可以使用ffmpeg推流，命令如下：

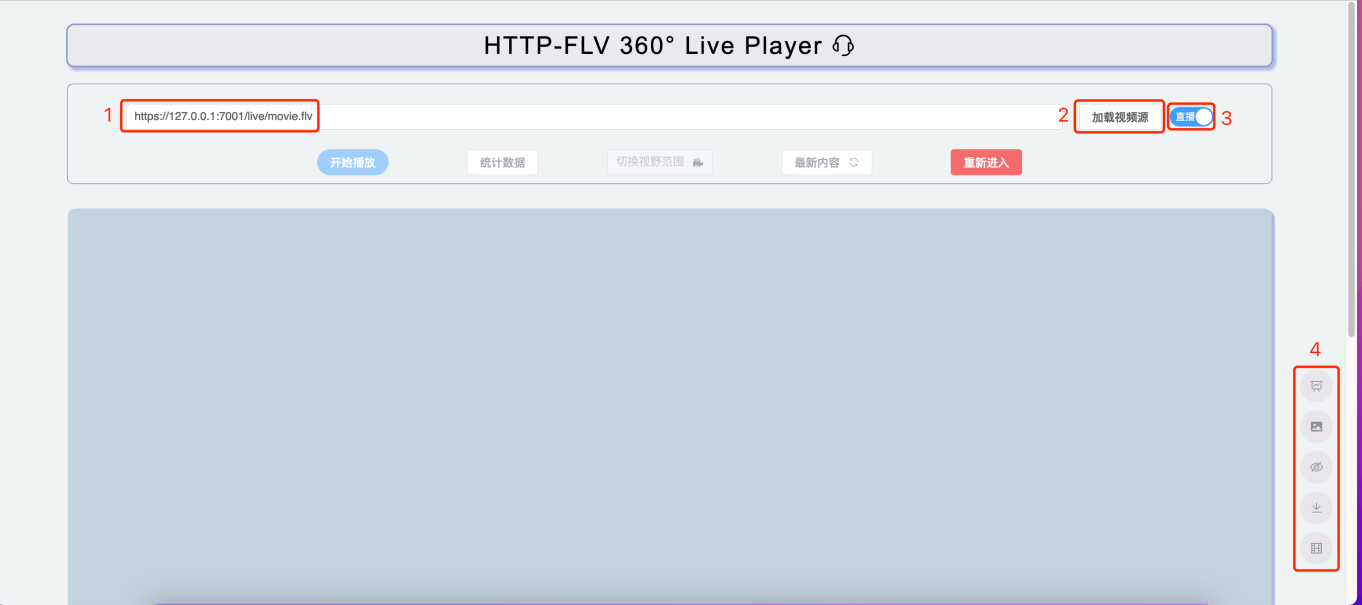
ffmpeg -re -stream\_loop -1 -i test.mp4 -vcodec libx264 -acodec aac -f flv rtmp://127.0.0.1:1935/live/movie

其中stream\_loop表示循环推流，如果视频为h264编码、音频为aac编码的话可以将libx264、aac改为copy减少编解码性能消耗

movie为直播的channel名，可以改为任意英文字符串

## 客户端使用

浏览器输入客户端地址后进入以下页面：



1. 视频地址栏，输入类似地址（https://127.0.0.1:7001/live/movie.flv）访问指定的视频流，该地址由rtmp推流的app和channel决定（例如：rtmp://127.0.0.1:1935/live/movie）
2. 加载视频按钮，在第三步选择直播后点击
3. 切换直播、点播，在本系统中使用直播模式
4. 监控组件，点击可显示相应的监控数据

# 附录

**技术支持信息：**

[livego](https://github.com/gwuhaolin/livego)

[GoRtp](https://github.com/wernerd/GoRTP)

[quic-go](https://github.com/quic-go/quic-go)

**相关基础知识：**

[RTMP](https://github.com/melpon/rfc/blob/master/rtmp.md)

[RTP](https://www.rfc-editor.org/rfc/rfc3550.html)

[QUIC](https://datatracker.ietf.org/doc/html/rfc9000)

[HTTP-FLV](https://ossrs.io/lts/en-us/docs/v4/doc/delivery-http-flv)

[HLS](https://www.rfc-editor.org/rfc/pdfrfc/rfc8216.txt.pdf)

**代码结构：**

|  |
| --- |
| **云端节点** |
| **- `defines.go`：基本配置项文件，包括flv格式用到的常量和rtp缓存、监听地址等参数**  **- `conn.go`：quic 流对象，用于重传丢失的 rtp 数据包**  **- `flv.go`：处理 flv 数据，包括构造 flvTag 以及读写 flv 数据**  **- `listQueue.go`：缓存 rtp 数据包的队列，通过arraylist实现【当前使用】，对外接口和 mapQueue 一致**  **- `mapQueue.go`：缓存 rtp 数据包的队列，通过hashmap实现，对外接口和 listQueue 一致**  **- `utils.go`：建立udp连接、quic连接等工具方法**  **- `cloundserver.go`：主要代码入口程序**  **- `metrics.go`：监控程序** |

|  |
| --- |
| **边缘节点** |
| **cache**  **- `cache.go`：主要是缓存 flvTag 的初始段 initialization segment，通常包含在首个音频和视频的 Tag 中，包含了媒体的基本信息，例如编解码格式以及采样率等，播放器必须拿到才能正确解码播放视频**  **- `RtpQueue.go`：用于缓存rtp包的队列，确保rtp包的有序和尽可能存在**  **- `FlvRecord.go`：解析flvTag的缓存，记录历史信息，每得到一个完整的tag后都会重新开始**  **- `App.go`：一个边缘节点实例，记录所有的流、udp连接等信息**  **certs**  **https服务的相关证书和密钥**  **configure**  **- 配置文件**  **container**  **- 一些协议的数据包格式，用来创建和解析不同协议的数据包**  **protocol/httpflv**  **- 提供httpflv服务的必要文件，向客户端传输数据包，主要数据结构是 flvWriter**  **protocol/quic**  **- quic 客户端，主要根据 sequence number 重传 rtp packet**  **utils**  **- tls、flv文件的读写、http请求等工具方法**  **main.go**  **- 主要代码逻辑，接收 rtp 数据, 解析、处理，提供http服务** |