

QOS-FEC-NACK 传输库评估报告(V1.0.0)

www.mediapro.cc

一、QOS-FEC-NACK 传输库

QOS-FEC-NACK 是一套集 FEC 前向纠错、QOS、NACK 选择性重传、JitterBuff、码率自适应等技术于一体的实时音视频传输解决方案。方案基于私有的 UDP 协议，以库或源码的方式提供用户，其接口简洁可快速集成到用户现有音视频系统之中。方案使用 C++ 开发支持 Windows、Android (JNI)、ios、Linux 系统，支持 X86、ARM 32 位、64 位平台。

QOS-FEC-NACK 库特别适合在需要在弱网（4G、Wifi）下进行可靠、实时音视频传输的领域。它具有以下特点：

- A、自适应 FEC 冗余度，根据当前网络状况自适应调整冗余度，提高抵抗力同时避免带宽浪费。
- B、自适应 FEC Group 配置，对不同时刻、不同特征的媒体数据使用不同的 FEC Group 策略，在不增加抖动的前提下提升连续丢包的抵抗力。
- C、选择性 NACK 重传，只对预期无法恢复的数据包发起重传，最大限度避免拥塞和抖动。
- D、自适应 NACK 等待时间选取，内置 NACK 信令、数据处理，对外层黑盒。
- E、无延时的乱序、重复包处理。
- F、JitterBuff 自适应缓存处理，抵抗网络抖动，提供流畅输出。
- G、码率/帧率自适应建议输出
- H、IDR 帧请求建议输出
- I、丢帧冻结机制支持
- J、提供丢包率、码率、RTT 等上下行基础统计数据获取
- K、针对嵌入式等资源受限平台设计，资源占用低，运行效率高。编码规范、代码精简、注释完善，不依赖于任何第三方开源或闭源库。

本文档使用一个点对点投屏 DEMO 对方案进行各项弱网模拟测试，研究各种弱网特征下传输层的表现情况。文档最后对行业内优秀解决方案：腾讯云、声网 Agora WebRtc 等进行了研究。

二、实验环境

1、网络环境

为了保证测试环境一致性和可重现性，我们将在较好的网络环境下借助第三方弱网模拟工具，模拟各类网络情况。同时也会使用信号较弱的 wifi 搭建真实的弱网环境。

这里列举 Windows 平台上常用的两款弱网模拟工具：

(A) Network Emulator for Windows Toolkit

该工具是一款 windows 平台弱网模拟工具，使用说明可以参考以下链接：

<https://blog.csdn.net/lluoZh2015/article/details/50545159>

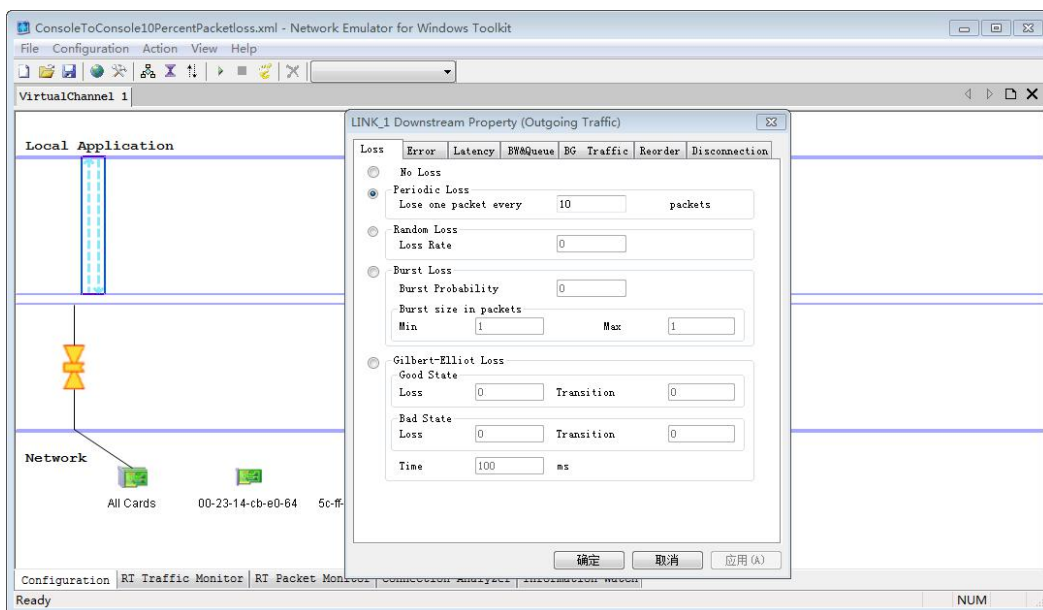


图 1 Network Emulator for Windows Toolkit 工具

软件提供：丢包、包篡改、延时、带宽限制、乱序、断网等功能。注意事项：

1、我们按照上面链接搭建测试工程后，发现实际未生效。解决办法是加载程序安装包自带的样例 XML 配置（ConsoleToConsole2PercentPacketloss.xml），在此配置基础上修改为自己的测试需求。

2、如上图所示，测试项可以作用于本机输入也可以作用于本机输出。比如对 Outgoing 的丢包设置将对本机发出的包进行丢包，适合在发送端使用。而对 Incoming 的丢包设置将对本机收到的包进行丢包，适合在接收端使用。对待测应用程序而言，在发送端丢包还是接收端丢包没有差异，本次实验均在发送端进行弱网测试。

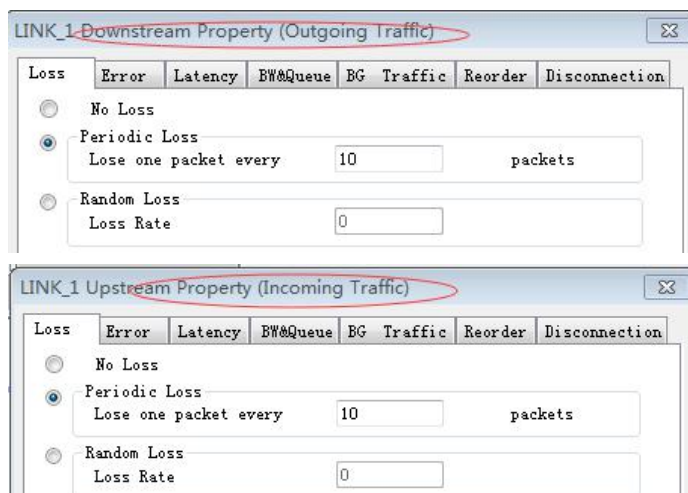


图 2 在发送和接收处模拟弱网

（B）Clumsy

Clumsy 是一款小巧而功能强大的开源弱网模拟工具，支持 windows 平台，可用于模拟：丢包(Drop)、延时(Lag)、重复(Duplicate)、乱序(Out of order)、篡改(Tamper)、抖动(Throttle)等。其项目地址：

<http://jagt.github.io/clumsy/cn/index.html>

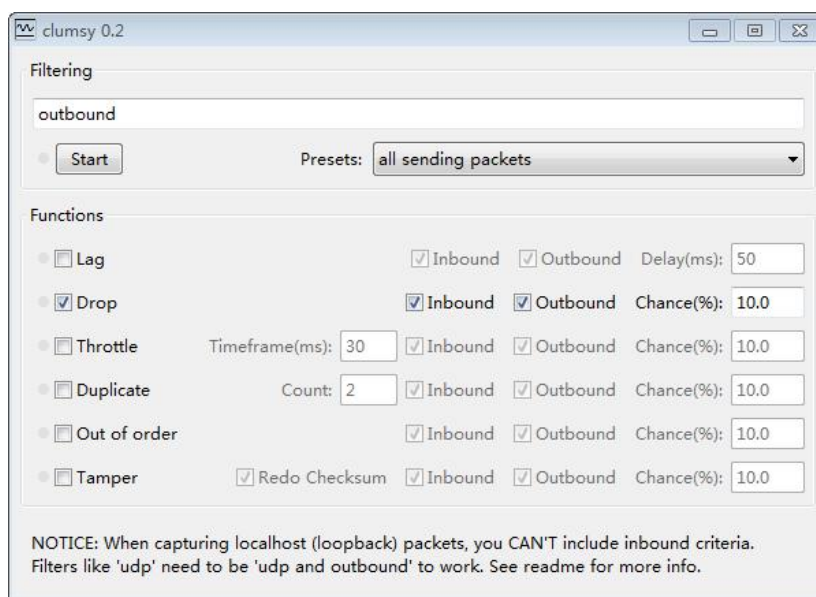


图 3 Clumsy 对所有发送的包按 10%进行丢包处理示意

本次测试主要使用 **Clumsy**，相比使用 **Network Emulator for Windows Toolkit**，二者测试结论基本一致。

2、测试 DEMO 说明

本次测试使用 windows 平台下的桌面投屏 DEMO，DEMO 分为发送端和接收端，发送端采集自身桌面和扬声器音频，压缩后通过 QOS-FEC-NACK 点对点 SDK 发往接收端，后者解码并渲染输出，从而实现屏幕共享功能。

A、接收端

该组 DEMO 的功能与投屏类应用类似，我们首先启动接收端 DEMO。进入 UDP-AVClient-ScreenPlay 文件夹，双击启动 AVClient.exe 即可。



图 4 接收端 DEMO 启动界面

接收端启动后，将显示其接收的 IP 和端口，发送端可以使用该信息进行投屏。

当发送端码流到来时，接收端将使用一个新的窗口“Remote Video”显示远端画面，如下图所示：



图 5 接收端独立的窗口展示远端画面

注意：“Remote Video”窗口是一个全屏窗口，用户可以自行在底部任务栏切换。当远端停止音视频传输时，该窗口内容无更新，且不会响应鼠标事件，只能底部切换。

接收端文件夹下的 AVClient.ini 文件为其配置文件，对配置文件的修改需要重启客户端方能生效。配置文件包括如下几项：

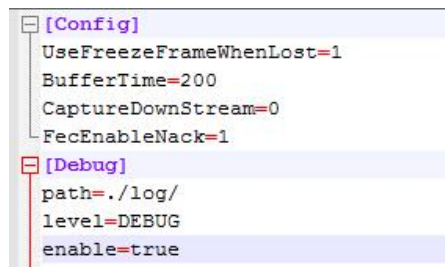


图 6 接收端配置文件

UseFreezeFrameWhenLost 表示当出现视频丢包无法恢复时，为了不展现出花屏而将画面冻结，直至完整的关键帧到来再继续送显。该值一般设置为 1 开启。

BufferTime 表示接收 Jitter buff 缓存毫秒数，为了抵抗网络传输、FEC 恢复、QOS 乱序恢复、NACK 重传等行为带来的抖动，接收端需要加入缓存以保障视频的流畅性。流畅性和实时性（时延）是一对矛盾的指标，Jitter buff 必然将引入一定延时，当前默认为 200ms。

CaptureDownStream 表示是否开启录制功能，若开启则将接收到的音视频流录制到 TS 文件。测试过程中建议关闭。

FecEnableNack 表示本端视频是否开启 NACK 功能，建议开启以提高视频抗丢包能力。

Debug 组下是日志相关的配置，比如 **path** 指定日志文件存放的路径，**level** 表示当前期望输出的日志级别，常用级别有 DEBUG, INFO, WARN, ERROR。**enable** 表示是否启动日志功能，建议启用。

在后面的测试过程中，将会对部分配置进行调整，查看调整前后的效果对比。

B、发送端

发送端为 UDP-AVClient-ScreenCap 目录下的 AVClient.exe，在启动前我们需要先对其进行配置，配置文件为 AVClient.ini

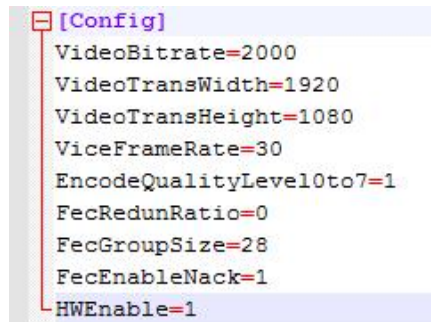


图 7 发送端配置文件

VideoBitrate 表示发送端使用的视频编码码率，单位 kbps，设置为 3000 即表示 3Mbps

VideoTransWidth 表示发送端使用的视频编码宽度，**VideoTransHeight** 表示视频编码高度，**ViceFrameRate** 表示视频编码帧率（本程序使用 Direct 桌面采集，在性能较低的机器上采集帧率无法达到 30fps，编码帧率仍然会按 30fps 配置编码器）

RemotelpAddr 表示接收端的 IP 地址，请按自己接收端实际情况进行配置。

EncodeQualityLevel0to7 表示当采用 X264 软编码时，使用的 preset 级别，0 表示 ultrafast，1 表示 superfast，2 表示 veryfast，3 表示 faster，4 表示 fast，5 表示 medium，6 表示 slow，7 表示 slower。等级越高同等码率下的图像质量越好，但 CPU 占用也越高，请根据自身机器配置而定，建议设置为 1。

HWEnable 表示是否启用硬编码，程序支持 Intel QSV 硬编码和 Nvidia 硬编码，相比 X264 能获得更低的 CPU 占用。不过硬编码的缺点是灵活性不足，无法支持传输层 IDR 帧请求机制。

FecRedunRatio 表示上行 FEC 使用的冗余度，比如设置为 30 时表示使用 30% 的上行冗余，设置为 0 时表示使用自动冗余度。

FecGroupSize 表示上行 FEC 使用的 group 标准分组大小。为了获得最佳效果，分组大小建议与码率想匹配。512Kbps 以下建议设置为 8，512Kbps~1Mbps 建议设置为 16，1Mbps~2Mbps 建议设置 24，2Mbps~4.5Mbp 建议设置 28，4.5Mbps 以上建议 34。**注：**当关闭自动 group 策略时，每个 group 大小均为该值设定值。当开启传输层自动 group 策略时，将产生非均匀大小的 group，此时该值用来表示最小的 group 大小。

FecEnableNack 表示是否启用 NACK 选择性重传机制。收发双方均开启时方能生效，建议双方均开启以提高系统抗丢包能力。

系统使用的 IDR 帧间隔说明：当使用 X264 编码时，发送端使用 5 秒一个 IDR 帧。当使用硬编码时，发送端使用 3 秒一个 IDR 帧。

启动发送端后进入如下界面，输入接收端展示的投屏 IP 端口组合即可开始连接。

注意：收发双方并无 TCP 连接，这里的连接可以理解为本地 UDP 资源的创建。

注意：因发送端需要使用管理员权限运行，否则可能无法加载桌面采集驱动而无法达到最佳性能。



图 8 发送端启动界面

连接后，客户端将进入下图所示的待共享屏幕状态。可以点击主界面启动按钮或者使用悬浮球来启动桌面共享。启动后，接收端就能看到发送端的桌面并能听到发送端播放的音乐了。



图 9 发送端开始共享桌面

三、测试项说明

说明：同市面上各大实时视频服务商一样，DEMO 也提供丢帧冻结机制，这样用户无法察觉到丢帧带来的花屏，从而获得更好的用户体验。因此本次测试中，丢包最终将体现为画面卡顿。DEMO 提供了发送端码率自适应功能，传输层根据当前的网络状况实时调整发送帧率，从而达到间接调整码率的目的。相比直接调整编码码率，调整帧率有如下优点和缺点：

优点：

A、相比直接调整码率，更难察觉质量跳变。

B、无需适配各个平台的硬件编码器，各个平台均可以采用统一的帧率调整方案。

缺点：

1、画面流畅性受到影响

评测项：流畅度

关于流畅度，我们将分为以下几个级别：

A、画面流畅

B、偶尔微弱卡顿（附加：卡顿时长+频率描述）

C、明显卡顿（附加：卡顿时长+频率描述）

D、较长时间卡顿

评测项：延时

延时计算方式：在发送端打开毫秒精度秒表，接收端将看到秒表值，使用手机对二者屏幕拍照，计算二者差值得到总延时。整个系统中，延时主要有非传输层延时和传输层延时两部分组成。非传输层延时包括：采集、编码、解码、渲染引入的延时，本 DEMO 实际采集帧率无法达到恒定 30fps，对整体延时稍有影响。

传输层延时又分为：相对稳定部分和抖动延时部分。其中接收端 Jitter buff 程序加入的缓存延时属于相对稳定部分。网络线路传输延时、QOS 乱序等待时间、NACK 重传等待时间、FEC 恢复等待时间、画面冻结等待时间属于抖动延时部分，抖动延时只在该动作发生时引入，且动作完成后消失。QOS 乱序发生时才会引入等待，比如收到 1、2、4 号包，输出 1、2 后会进入等待，若此期间收到 3 号包则输出 3、4，若超出等待时间仍未收到 3 号包则直接输出 4 号包，即便后续收到 3 号包也将其丢弃。若当前丢包无法恢复时，即会触发 NACK 重传，接收方进入 NACK 等待，等待期间收到了重传包则输出，否则等待超时后退出。FEC 有 group 组的概念，冗余包位于组的尾部，前部媒体包的丢失需要等待尾部冗余包的到来方能恢复输出，因此 FEC 解码在丢包时也会引入抖动，FEC group 越大引入的抖动也越大，不过在同等冗余率下抗连续丢包的能力也越强。当 NACK、FEC 均无法恢复时，将冻结画面并请求远端发送 IDR，只有收到完整的 IDR 帧时才恢复送解码、渲染，这里也将引入抖动延时。编码器

Gourp 越大，“可能”需要的 IDR 等待时间越长（当接收端主动请求的 IDR 帧也出现丢包时，只能依靠编码器自身的周期性 IDR 帧。当接收端主动请求 IDR 帧传输成功时，等待时间和编码器自身的周期性 IDR 间隔无关。）

需要说明的是延时指标和流畅性指标往往是一对矛盾，播发端缓存的数据越多，流畅性越好，延时也越大，反之若缓存的数据较少或者不缓存，则延时更低，流畅性不足。传输层需要根据实际应用场景选择合适的策略（折中）。SDK 提供 API 供用户配置接收端的 Jitter Buff 缓存毫秒数。默认情况下使用 300ms 缓存，这是基于 300ms 的延时不会对双向音视频实时互动产生影响这一业内经验。

评测项：清晰度

DEMO 使用自适应帧率方式来间接实现码率自适应，因此图像质量与传输层无紧密关系，主要由用户指定的编码分辨率、码率、桌面画面内容决定。注：帧率降低时，帧间相关性降低，运动估计残差更大，同等码率下编码质量会稍弱。

四、测试结果

1、丢包测试

测试配置 A:

接收端使用 300ms 缓存，具体配置入下图所示：

[\[Config\]](#)

UseFreezeFrameWhenLost=1

BufferTime=300

FecEnableNack=1

发送端使用 720P 2Mbps 30fps，FEC 使用自动冗余，具体配置入下图所示

[\[Config\]](#)

VideoBitrate=2000

VideoTransWidth=1280

VideoTransHeight=720

ViceFrameRate=30

EncodeQualityLevel0to7=1

FecRedunRatio=0

FecGroupSize=28

FecEnableNack=1

HWEnable=0

画面内容：全屏播放影片

发送端使用 Clumsy 设置发送丢包 5%、8%、12%、20%、30%，为了排除遗留影响，每次修改丢包率均在发送端断开连接再重新连接。

5%丢包时，连续观察 20 分钟，画面流畅，较难感知丢包，延时稳定在 300ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

8%丢包时，连续观察 20 分钟，画面流畅，较难感知丢包，延时稳定在 300ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

12%丢包时，连续观察 20 分钟，画面流畅，较难感知丢包，延时稳定在 300ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

20%丢包时，连续观察 20 分钟，因码率自适应被触发，画面帧率逐渐下降，总体比较流畅，较低频率偶尔卡顿，卡顿时长约 300ms 左右（IDR 请求）。延时稳定在 330ms 左右。码率约 2.2Mbps。

30%丢包时，连续观察 20 分钟，因码率自适应被触发，画面帧率逐渐下降，有较明显的卡顿。延时稳定在 400ms 左右。码率约 2.0Mbps。

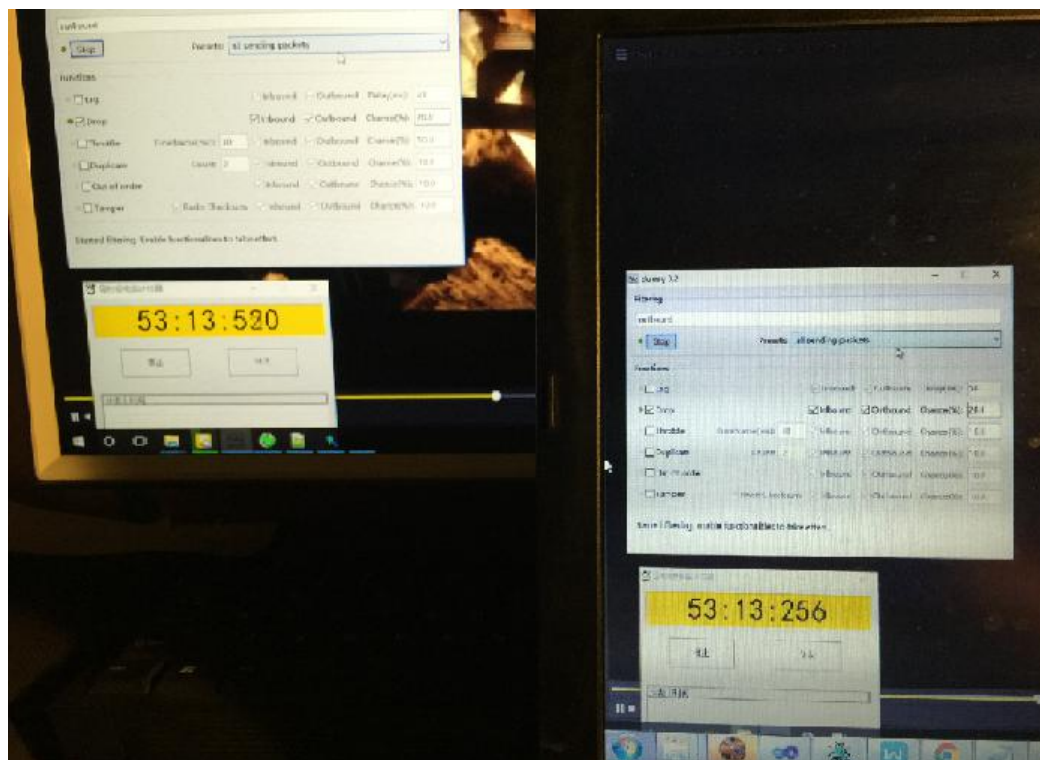


图 10 20%丢包时的延时情况

2、重复测试

测试配置 A，发送端使用 Clumsy 设置 Duplicate 发送重复率 5%、12%、20%、30%，每次重复 1 包（Count 设置为 2）。

5%重复包时，连续观察 20 分钟，画面流畅，延时稳定在 260ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

12%重复包时，连续观察 20 分钟，画面流畅，延时稳定在 260ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

20%重复包时，连续观察 20 分钟，画面流畅，延时稳定在 260ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

30%重复包时，连续观察 20 分钟，画面流畅，延时稳定在 260ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

可见单纯的重复包对系统影响很小。

3、乱序测试

测试配置 A，发送端使用 Clumsy 设置 Out of order 发送乱序率 5%、12%、20%、30%。

5%乱序包时，连续观察 20 分钟，画面流畅，延时稳定在 280ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

12%乱序包时，连续观察 20 分钟，画面流畅，延时稳定在 280ms 左右，冗余率基本维

持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

20%乱序包时，连续观察 20 分钟，画面流畅，延时稳定在 280ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

30%乱序包时，连续观察 20 分钟，画面流畅，延时稳定在 280ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

可见单纯的乱序包对系统影响很小。

4、延时测试

测试配置 A，发送端使用 Clumsy 设置 Lag 发送延时 50、100、200、400、600ms。

50ms 时，连续观察 20 分钟，画面流畅，延时稳定在 310ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

100ms 时，连续观察 20 分钟，画面流畅，延时稳定在 340ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

200ms 时，连续观察 20 分钟，画面流畅，延时稳定在 520ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

400ms 时，连续观察 20 分钟，画面流畅，延时稳定在 740ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

600ms 时，连续观察 20 分钟，画面流畅，延时稳定在 920ms 左右，冗余率基本维持在自动冗余度的下限 30%，码率平均约 2.4Mbps。

线路延时最终会叠加到总体延时之上，测试结果符合预期。

5、抖动测试

测试配置 A，分别进行以下几项测试

A、发送端使用 Clumsy 设置 Throttle 分别 5%、12%、20%、30%概率抖动 30ms。

测试结果：5%~30%概率 30ms 抖动对流畅性、延时无可感知的影响。

B、发送端使用 Clumsy 设置 Throttle 分别 5%、12%、20%、30%概率抖动 100ms。

测试结果：5%~30%概率 100ms 抖动对流畅性、延时无可感知的影响。

C、发送端使用 Clumsy 设置 Throttle 分别 5%、12%、20%、30%概率抖动 150ms。

测试结果：5%~30%概率 150ms 抖动对流畅性存在一定影响，几十秒一次的可感知微弱顿挫感。

D、发送端使用 Clumsy 设置 Throttle 分别 5%、12%、20%、30%概率抖动 200ms。

测试结果：5%~30%概率 200ms 抖动对流畅性存在较大影响，随着概率的增加，30%概率时一秒一次的可感知明显顿挫感。

抖动达到一定程度时，超出 Jitter Buff 抵抗力，将出现画面顿挫。目前版本暂未加入 Jitter Buff 能力自适应，用户设定 Jitter Buff 深度后，即根据设定值确定缓存深度，不会跟随媒体流实际抖动自适应调整。这样做有以下优缺点：

优点：系统延时稳定于用户设定值，不随网络抖动增长而增长。

缺点：网络抖动超出设定值时，播放器将出现卡顿。当实际抖动小于设定值时，仍然引入了设定值的延时。

抖动优化是后续传输层优化的方向之一。

6、极速测试

当设定接收端 Jitter Buff 缓存 0ms 时，即关闭接收端缓存功能，收到数据第一时间解码、第一时间渲染，此时无程序引入的延时，我们称之为极速模式。

设置接收端配置文件 BufferTime=0，不加入人为弱网措施时：

测试结果：连续观察 20 分钟，画面总体比较流畅，延时稳定在 94ms 左右。对于延时要求较高，而对于流畅性没有极高要求的场合可以使用极速模式。

7、断网测试

本 DEMO 收发双方并无 TCP 连接，断开一方网络后，另一方只是无法接收或发送媒体数据，待网络恢复后自行恢复。

五、竞品分析

实时音视频传输一直是多媒体通信的核心之一，腾讯云、网易云信以及部分新兴企业如声网、即构均提供了云解决方案。各家方案各有千秋，通过相互对比借鉴，能获得更多灵感。

1、腾讯云：<https://cloud.tencent.com/product/trtc>

网站介绍：腾讯实时音视频（Tencent Real-Time Communication，TRTC）是腾讯云基于 QQ 十多年来在音视频通话技术上积累，提供全平台互通高品质实时视频通话服务的一款产品；抗丢包率超过 40%，抗网络抖动超过 1000ms，即使在弱网环境下仍然能够保证高质量的音视频通信，确保视频通话过程顺畅稳定。

注意：腾讯云、阿里云等也提供基于 RTMP/HLS 的直播服务，这类基于 TCP 协议的直播并非本文所描述的音视频实时互动范畴，它们往往用于对延时要求不高的单向的音视频传输服务，其对弱网的抵抗力较差，在网络恶化时服务质量衰减迅速。本报告中，我们将以大牛直播 RTMP 传输为例，测试基于 TCP 的 RTMP 协议在弱网下的表现。

腾讯实时音视频同样使用私有的 UDP 协议，目前广泛应用于微信、QQ 等产品。腾讯云实时音视频官网提供了 DEMO 供用户测试，使用步骤大致如下，具体请参考腾讯文档说明：

<https://cloud.tencent.com/document/product/647>

A、用户需要先在腾讯云上注册实时音视频服务，并在控制台中指定音视频参数。

B、下载 iLive SDK 和配套 DEMO 工程，在 DEMO 代码中填写注册时生成的 APP ID 等参数并编译可执行程序。

用户在腾讯云后台中对音视频互动的详细参数进行设置（注意：腾讯云 SDK 并不在客户端进行音视频参数设置，而是在管理后台设置，客户端选择后台中的某组设置）。当前可供用户设置的分辨率最大到 720P，码率最高到 1500kbps，可能更高级别的参数需要人工客服申请。



图 11 腾讯云实时音视频控制台



图 12 腾讯云实时音视频能力设置

为了方便演示，我们在附件中提供了腾讯云 DEMO 的源码以及一个使用我们申请测试账号编译的可执行程序。程序包括一个发送端和一个接收端，发送端将采集系统默认的摄像头编码后发往腾讯云服务器，后者转发给接收端。

注意：为了搭建与我们 DEMO 类似的环境，我们使用虚拟桌面采集摄像头作为系统默认摄像头（运行我们 DEMO 后，将自动向系统注册一个名为 screen-capture-recorder 的虚拟摄像头，拔掉发送端机器的其他摄像头，确保让虚拟摄像头成为默认摄像头）。

注意：腾讯云并不提供 P2P 服务，所有音视频均走服务器转发，这可能是由其收费方式决定的，对于所有用户按使用分辨率、时长收费，一个发送端、一个接收端使用 1 小时，则按 2*1 小时计费。若提供 P2P 服务，并对用户之间的 P2P 传输收取费用则于情于理说不过去。

注意：腾讯云 SDK 将视频编解码和传输层一起封装。

正常情况下，使用 720P 1.5Mbps 码率，在无弱网模拟的情况下，连续观察 20 分钟，画面流畅，延时稳定在 140ms 左右，码率平均约 1.5Mbps。

丢包测试结果：

5%丢包时，连续观察 20 分钟，画面流畅性有一定下降（帧率下降导致，均匀丢帧，无长时间卡顿）延时稳定在 150ms 左右，码率平均约 1.6Mbps。

12%丢包时，连续观察 20 分钟，画面流畅性进一步下降（帧率下降导致，均匀丢帧）偶尔（几分钟）出现明显卡顿，延时稳定在 140ms 左右，码率平均约 1.6Mbps。

20%丢包时，连续观察 20 分钟，画面几秒一次频繁卡顿（卡住约 500ms 左右），帧率较低，延时稳定在 200ms 左右，码率平均约 1.6Mbps。

通过画质对比，腾讯云应该也是采用帧率自适应，并未降低码率，图像质量相对恒定。码率控制和用户购买的非常接近，毕竟这个涉及到自身成本。延时总体较低，接收端缓存较小，在丢包时因 NACK 引入的瞬间抖动导致画面易出现卡顿。IDR 帧间隔约为 1 秒。

抖动测试结果：

使用 720P 1.5Mbps 码率，30%概率引入 100ms 抖动，无丢包，此时画面流畅，延时扩大到 240ms 左右，说明腾讯检测到网络持续抖动后自适应的增加了接收端缓存时间。当关闭抖动模拟时，延时恢复到 140ms。观察到开启抖动模拟时，画面会持续 1~2 秒的卡顿期，然后延时加大，流畅性提升，说明在此期间进行了缓存的重新初始化动作。腾讯的接收缓存自适应调整的策略值得我们学习。

重复测试结果：

30%重复包时，连续观察 20 分钟，画面流畅性与未开启时基本一致，延时稳定在 140ms 左右，码率平均约 1.5Mbps。

乱序测试结果：

30%重复包时，连续观察 20 分钟，画面流畅性与未开启时基本一致，延时稳定在 140ms 左右，码率平均约 1.4Mbps。

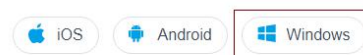
计费标准：<https://cloud.tencent.com/document/product/647/17157>

2、声网：<https://www.agora.io/cn>

网站介绍：SD-RTN 是专为实时传输设计的虚拟通信网络，基于 UDP，延时可控。专利算法，极大幅提升可靠性。全球部署近 100 个数据中心，国内数十家中小运营商全面覆盖，99.99% 高可用，连通率 99.9%。

声网实时音视频同样使用私有的 UDP 协议，基于 WebRTC 技术优化开发，SDK 将音视频编解码和传输层封装在一起。

Agora Video Call 体验多人音视频通话品质



下载 Agora App 即刻体验

搭建项目前，可下载 Agora App，快速体验品质



Agora Live 体验多人连麦直播品质

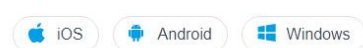


图 13 声网 DEMO 下载

注意：我们选择多人音视频通话 DEMO，Wireshark 截包发现该 DEMO 在同一网段内走的是 P2P（虽然走的 P2P，但仍然按量收费）。附件中已经包括了测试 DEMO 执行程序。

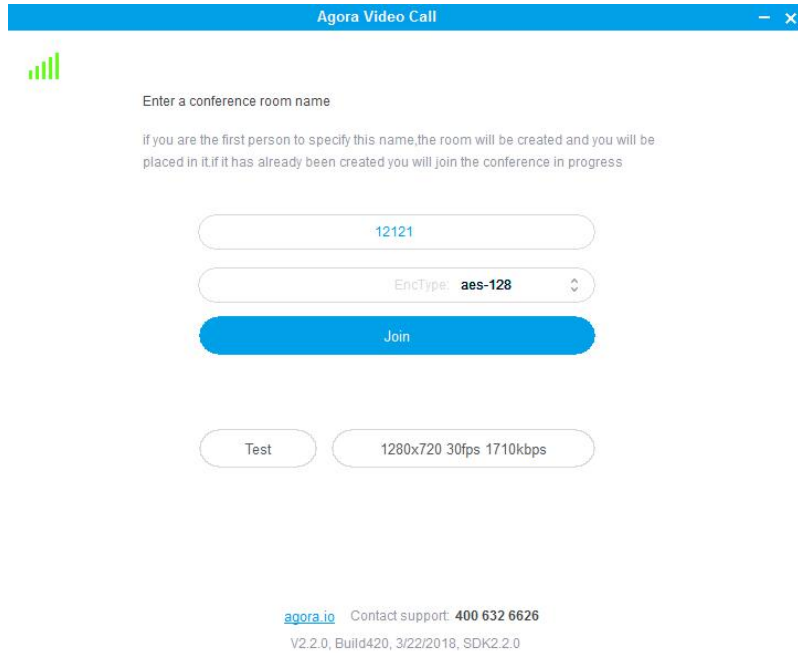


图 14 声网 DEMO 启动界面

在启动 DEMO 后，选择 720P 30fps 分辨率（这个分辨率为摄像头通讯时的分辨率，本次测试我们使用屏幕共享，后者将默认使用桌面的分辨率进行通讯，帧率最高只能设置 15fps），并填写一个房间号，收发双方使用相同的房间号即可进行互通。

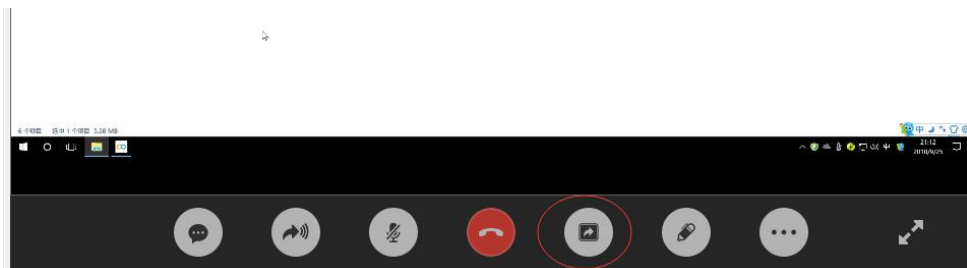


图 15 使用屏幕共享功能

我们将桌面分辨率设置为 720P，DEMO 将自动使用 720P 约 800Kbps 码率（DEMO 未提供对桌面共享分辨率、码率的设置功能），在无弱网模拟的情况下，连续观察 20 分钟，画面因帧率 15fps 的缘故流畅性一般，延时稳定在 240ms 左右。

丢包测试结果：

5%丢包时，连续观察 20 分钟，画面流畅性有一定下降（帧率动态变化，最低到 8fps，丢帧不够均匀，无长时间卡顿）延时稳定在 400ms 左右，码率平均约 800Kbps。

12%丢包时，连续观察 20 分钟，画面不流畅（帧率下降导致，非均匀丢帧），延时稳定在 550ms 左右，码率平均约 800Kbps。

20%丢包时，连续观察 20 分钟，画面很不流畅（非均匀丢帧，帧率在 8~13fps），延时稳定在 400ms 左右，码率平均约 800Kbps。

抖动测试结果:

使用 720P 800kbps 码率, 30%概率引入 100ms 抖动, 无丢包, 此时画面流畅性基本不受影响(受 15fps 缘故, 流畅性一般), 延时扩大到 450ms 左右, 码率基本不变。说明声网检测到网络持续抖动后增加了接收端缓存时间。当关闭抖动模拟较长时间后, 延时仍然维持在 400ms 左右。这点与腾讯的处理策略不同。

重复测试结果:

30%重复包时, 连续观察 20 分钟, 画面流畅性与未开启时基本一致, 延时稳定在 220ms 左右, 码率平均约 800kbps。

乱序测试结果:

30%重复包时, 连续观察 20 分钟, 画面流畅性与未开启时基本一致, 延时稳定在 220ms 左右, 码率平均约 800kbps。

计费标准: <https://www.agora.io/cn/price/>

附加: QOS-FEC-NACK 15fps 720P 800Kbps 版本测试

因为声网 DEMO 限制, 只能达到 15fps, 为此我们将 QOS-FEC-NACK DEMO 也使用同等环境。发送端配置如下:

[Config]

VideoBitrate=800

VideoTransWidth=1280

VideoTransHeight=720

ViceFrameRate=15

EncodeQualityLevel0to7=1

FecRedunRatio=10

FecGroupSize=28

FecEnableNack=1

HWEnable=0

我们使用 720P 15fps 800Kbps 编码, 使用固定 10%的冗余度。

接收端配置如下:

[Config]

UseFreezeFrameWhenLost=1

BufferTime=300

FecEnableNack=1

在无弱网模拟的情况下, 连续观察 20 分钟, 画面因帧率 15fps 的缘故流畅性一般, 延时稳定在 320ms 左右。

5%丢包时, 连续观察 20 分钟, 画面流畅性基本不受影响(帧率稳定于 15fps, 无长时间卡顿)延时稳定在 330ms 左右, 码率平均约 900Kbps。

12%丢包时, 连续观察 20 分钟, 画面流畅性基本不变, 偶尔有短暂顿挫感(帧率稳定于 15fps, 无长时间卡顿)延时稳定在 400ms 左右, 码率平均约 900Kbps。

20%丢包时, 连续观察 20 分钟, 画面很不流畅(均匀丢帧, 帧率最低降到 8fps)。延时稳定在 400ms 左右, 码率平均约 900Kbps。

六、总结

QOS-FEC-NACK 方案可以在保障实时性前提下，有效提升音视频在弱网下的传输效果，相比业内领先的解决方案，传输效果在某些指标上已经较为接近。相比业内以云服务方式提供服务，QOS-FEC-NACK 提供开放的接口，可以方便的加入到私有网络中。方案以库或源码方式提供，相比昂贵的按时收费成本更低。我们将持续优化，以更高的要求、更严苛的环境促进 QOS-FEC-NACK 方案的进步。