Live Video Broadcasting

FAQ

Product Introduction

Contents

FAQ

Basics of LVB

Last updated：2018-07-24 16:09:24

1. What are Push, LVB and VOD?

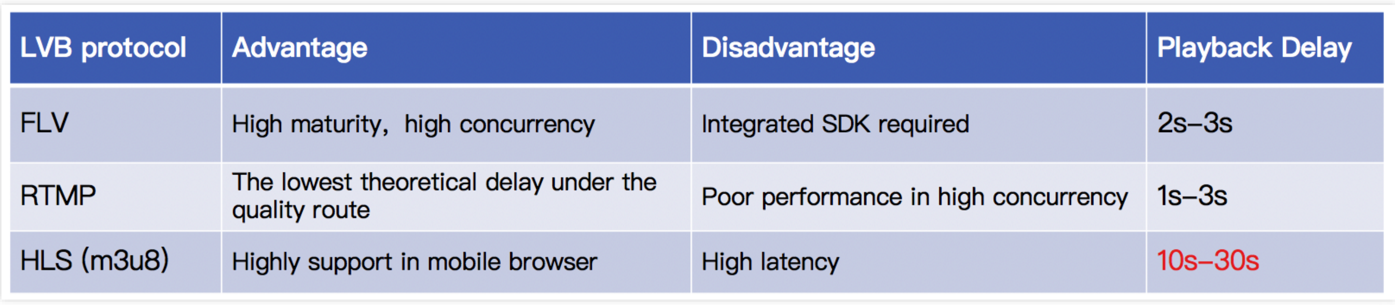
- **Push**: This refers to the process in which VJs push local video and audio sources to Tencent Video Cloud servers. It is also known as "RTMP Publishing" in some cases.
- **LVB**: LVB video source is generated in real time. It is only meaningful if someone pushes the live streams. Once the VJ stops broadcasting, the LVB URL becomes invalid, and since the live streams are played in real time, no progress bar is displayed on the player during the playback.
- **VOD**: VOD's video source is a file on cloud, which can be played at any time as long as it has not been deleted by the provider (such as Youku Tudou, iQIYI and Tencent Video). Since the entire video file is stored on the server, a progress bar is displayed during the playback.

2. What are the popular LVB protocols?

Three LVB protocols are commonly used: RTMP, FLV and HLS.

- **RTMP**: The RTMP protocol can be used for both push and live broadcasting. It involves breaking large video and audio frames into smaller fragments and transmitting them as small packets over the Internet. RTMP supports encryption, and thus provides a good privacy. However, the complicated fragmentation and reassembling bring about some unforeseeable stability issues to RTMP in case of a high-concurrency scenario.
- **FLV**: FLV protocol is mainly implemented by Adobe Systems. It simply places header information to the large video and audio frame headers. This simplicity makes it a sophisticated format in terms of delay control and high-concurrency performance. Its only disadvantage is the limited capability on mobile browsers. However, it's suitable for LVB on mobile Apps.
- **HLS**: HLS is a solution from Apple Inc. It splits videos into 5-10s fragments and manages them with an m3u8 index table. Since the videos downloaded on clients are complete data files with a 5-10s duration, the video smoothness can be ensured, but this leads to a great delay (typically the delay is around 10-30s when HLS is used). HLS is supported better than FLV on iPhone and most Android

browsers, so it is often used for sharing URLs in QQ or WeChat's "Moments".

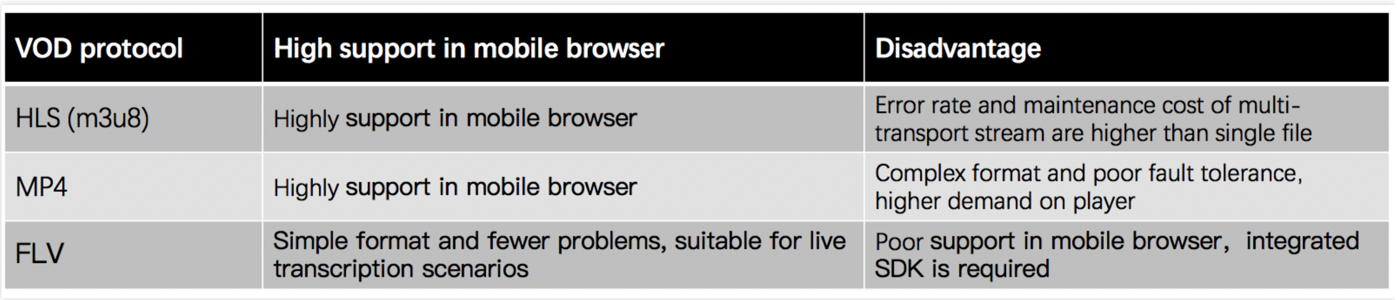


LVB protocol	Advantage	Disadvantage	Playback Delay
FLV	High maturity，high concurrency	Integrated SDK required	2s–3s
RTMP	The lowest theoretical delay under the quality route	Poor performance in high concurrency	1s–3s
HLS (m3u8)	Highly support in mobile browser	High latency	10s–30s

3. What are the popular VOD protocols?

Three VOD formats are commonly used: MP4, HLS and FLV.

- **MP4**: MP4 is a classic format that is well supported on both mobile devices and PC browsers (The default browsers of iOS and most Android devices support MP4. On PC it can be played in a FLASH widget). However, MP4 video files are formatted in a complicated manner, which makes it time-consuming to process the files. Furthermore, the complexity of index table can cause a slow load when a long MP4 file (e.g. half an hour) is played online.
- **HLS**: HLS is implemented by Apple Inc., and is well supported on mobile browsers. But on IE, the support for HLS depends on the secondary development of FLASH. (You're recommended to use Tencent Video Cloud's FLASH player). Unlike MP4 that has a slow indexing, HLS's compact m3u8 index structure allows a fast indexing, which makes it an ideal choice for VOD.
- **FLV**: Implemented by Adobe Systems, FLV is the most popular wrapper format on live broadcasting platforms. On PC, it's well supported by FLASH. However, on mobile devices, it is only supported by the Apps which implement their players (or use this player). Most mobile browsers don't support FLV. Tencent Video Cloud uses FLV for LVB recording.



VOD protocol	High support in mobile browser	Disadvantage
HLS (m3u8)	Highly support in mobile browser	Error rate and maintenance cost of multi-transport stream are higher than single file
MP4	Highly support in mobile browser	Complex format and poor fault tolerance, higher demand on player
FLV	Simple format and fewer problems, suitable for live transcription scenarios	Poor support in mobile browser，integrated SDK is required

4. What are the popular push protocols?

Although RTMP is not commonly used in live broadcasting, but it is dominant in push service (pushing data from **VJ** to **servers**). Domestic video cloud services use RTMP as the main push protocol. Tencent Video Cloud SDK's first module is VJ push, so the SDK is also called RTMP SDK.

5. Which features and protocols are supported by the Tencent RTMP SDK?

Tencent Video Cloud RTMP SDK supports Push, LVB and VOD.

- **Push**: Supports **RTMP publishing protocol**, as well as features such as hardware acceleration, beauty filter, bandwidth adaption and resolution adjustment.
- **LVB**: Supports **FLV (recommended)** and RTMP protocols, as well as instant broadcasting optimization, auto delay control and highly adaptive hardware-decoding.
- **VOD**: Supports **online** or **local** VOD services for **MP4\HLS\FLV** files. Note: The earlier versions of SDK only support FLV-based VOD.

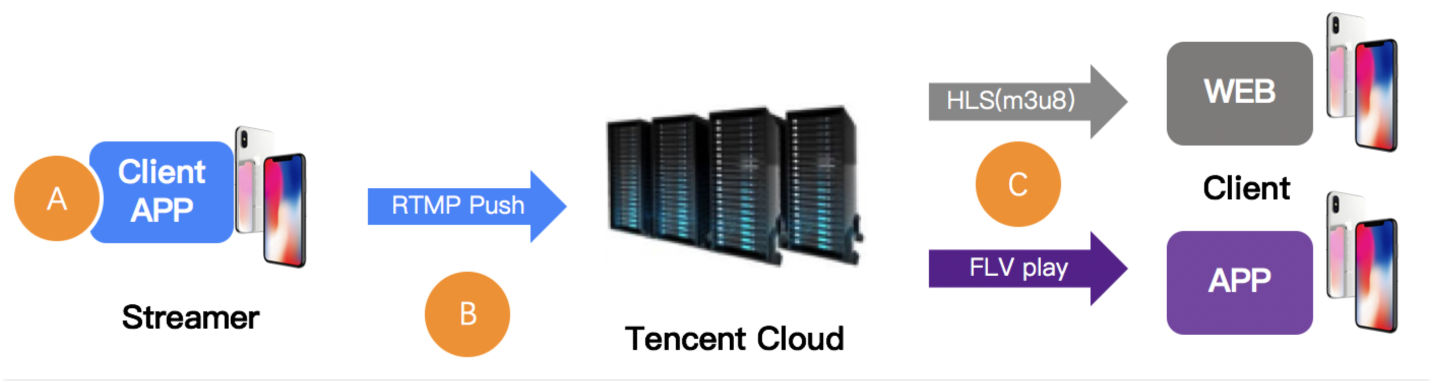
6. Is it mandatory to obtain the "License for Dissemination of Audio-Visual Programs through Information Network" for LVB?

Any applications that provide live broadcasting and other audio/video services over Internet are required to have the "License for Dissemination of Audio-Visual Programs through Information Network" issued by the relevant authority.

Video Stutter

Last updated：2018-07-24 16:19:42

1. What can cause a stutter during LVB?



Generally, there are three reasons for the stutter:

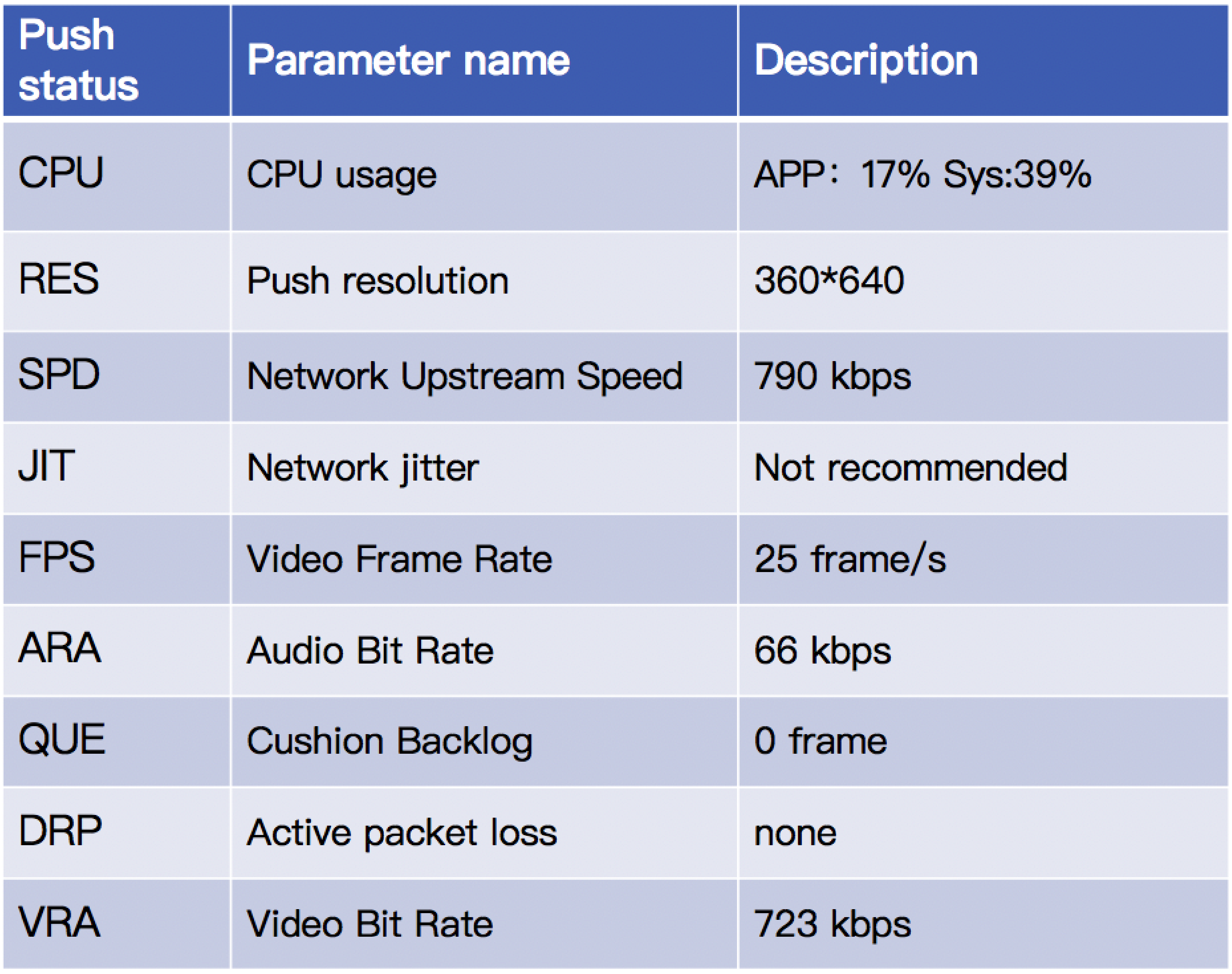
- **Reason 1: low frame rate**
  If the VJ uses a low-end phone, or there are CPU intensive applications running at the background, the frame rate of the video could be low. Typically, for an LVB to play smoothly, the frame rate of the video stream should be higher than 15 FPS. A frame rate lower than 10 FPS is **too low**, and can cause a stutter at **all the viewer ends**.
- **Reason 2: upstream clog**
  When pushing, VJs' phones generate audio and video data constantly. If the upstream bandwidth of a phone is too low, the generated audio and video data could clog the phone network and fails to be pushed, causing the stutter at **all the viewer ends**.
  Even though **domestic operators** offer broadband packages with a downstream bandwidth as fast as 10 Mbps, 20 Mbps or even 100 Mbps, the upstream bandwidth is highly limited. In many small cities, the upstream bandwidth is limited to 512 Kbps (i.e. a maximum of 64 KB data can be uploaded per second).
  Wi-Fi follows the IEEE 802.11 specification of carrier-sense multiple access and collision avoidance (CSMA/CA). To put it simply, a Wi-Fi hot spot can communicate with only one phone at one time, and other phones must verify or query if communication is possible before initiating a connection to a hot spot. Therefore, the more people using a Wi-Fi hot spot, the slower the connection is. Furthermore, Wi-Fi signal decays greatly when passing through walls or obstacles, and most of the families seldom take the Wi-Fi router position and the strength of Wi-Fi signal across rooms into consideration during the

design and decoration of their houses. Even the VJs themselves probably don't know how many walls are there between their routers and the rooms where they push video streams.

- **Reason 3: bad downstream connection**
  That is, the viewer's downstream bandwidth is insufficient or the network condition is unstable. For example, suppose the bitrate of an LVB stream is 1 Mbps (i.e. every second 1 M bits of data need to be downloaded). If the bandwidth at the viewer end is not fast enough, the viewer would experience serious stutter. Bad downstream connection only affects the viewers in the current network environment.

2. Status Monitor

The RTMP SDK provides a status feedback mechanism, by which the RTMP SDK reports various status parameters every 1-2 seconds. You can register the **TXLivePushListener** listener to obtain these status parameters.



Push status	Parameter name	Description
CPU	CPU usage	APP：17% Sys:39%
RES	Push resolution	360*640
SPD	Network Upstream Speed	790 kbps
JIT	Network jitter	Not recommended
FPS	Video Frame Rate	25 frame/s
ARA	Audio Bit Rate	66 kbps
QUE	Cushion Backlog	0 frame
DRP	Active packet loss	none
VRA	Video Bit Rate	723 kbps

Push Status	Description
NET_STATUS_CPU_USAGE	CPU utilization of current process and overall CPU utilization of the machine
NET_STATUS_VIDEO_FPS	Current video frame rate, that is, the number of frames produced by video encoder per second
NET_STATUS_NET_SPEED	Current transmission speed (in Kbps)
NET_STATUS_VIDEO_BITRATE	The output bitrate of the current video encoder, i.e., the number of video data bits produced by the encoder per second (in Kbps)

Push Status	Description
NET_STATUS_AUDIO_BITRATE	The output bitrate of the current audio encoder, i.e., the number of audio data bits produced by the encoder per second (in Kbps)
NET_STATUS_CACHE_SIZE	Accumulated audio/video data size. A value ≥ 10 indicates the current upstream bandwidth is not enough to consume the audio/video data produced
NET_STATUS_CODEC_DROP_CNT	The number of global packet drops. To avoid a vicious accumulation of delays, the SDK actively drops packets when the accumulated data exceeds the threshold. A higher number of packet drops means a more severe network problem.
NET_STATUS_SERVER_IP	The IP address of the connected push server. It is typically the nearest one with few hops from the client.

3. Low Frame Rate

3.1 How to verify if the frame rate is too low

We can obtain the video frame rate of the current push from the **VIDEO_FPS** status data of TXLivePushListener. Typically, for an LVB to play smoothly, the frame rate of the video stream should be higher than 15 FPS. A frame rate lower than 10 FPS could cause an obvious stutter at the viewer end.

3.2 Solutions

- **3.2.1 Observe CPU_USAGE value**
  You can obtain the **CPU utilization for the current push SDK** and **overall CPU utilization for the system** from **CPU_USAGE** status data of TXLivePushListener. If the overall CPU utilization for the system exceeds 80%, video capture and encoding may be affected; if the CPU utilization reaches 100%, the VJ end may be terribly stuck and it is impossible for the viewers to have a smooth viewing experience.
- **3.2.2 Identify the high CPU consumers**
  On an LVB App, in addition to RTMP SDK, many other features such as on-screen comments, floating stars, and interactive text messages can consume some CPU resources. To test and evaluate the CPU utilization for just the push SDK, use this simple DEMO.
- **3.2.3 Choose a reasonable resolution**
  A higher resolution doesn't always come with better video quality: firstly, a high resolution needs a higher bit-rate to work; a definition with a low bitrate and high resolution is often inferior to that with a high bitrate and a low resolution. Secondly, a high resolution such as 1280 x 720 does not have an obvious advantage on a 5'' phone screen. Only when the LVB is played full-screen on a PC can the

resolution of 1280 x 720 make a significant difference from 960 x 540. However, a higher resolution can bring about a big increase of CPU utilization for SDK. Therefore, in most cases, it's recommended to simply set the video quality to **High Definition** with TXLivePush's setVideoQuality. A higher resolution isn't always the better.

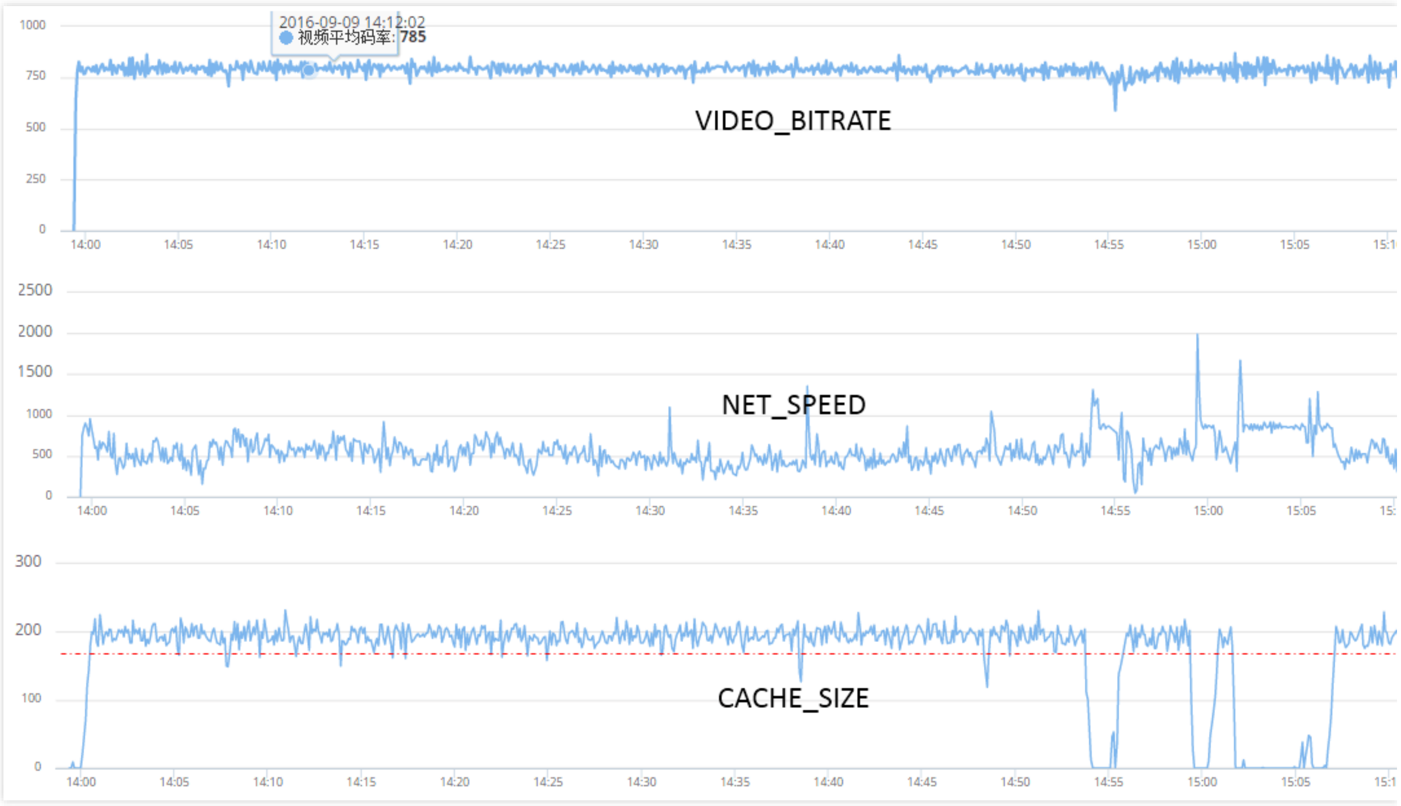
- **3.3.4 Use hardware acceleration if appropriate**
  Most smart phones support hardware encoding to lower the CPU utilization for video encoding. When CPU utilization is too high for an APP, you can enable hardware encoding to lower the CPU utilization. By default, the **High Definition** video quality option of TXLivePush's setVideoQuality uses software encoding (on some Android devices, hardware encoding doesn't function well due to the severe mosaics). You can use enableHWAcceleration of TXLivePushConfig to enable hardware encoding.

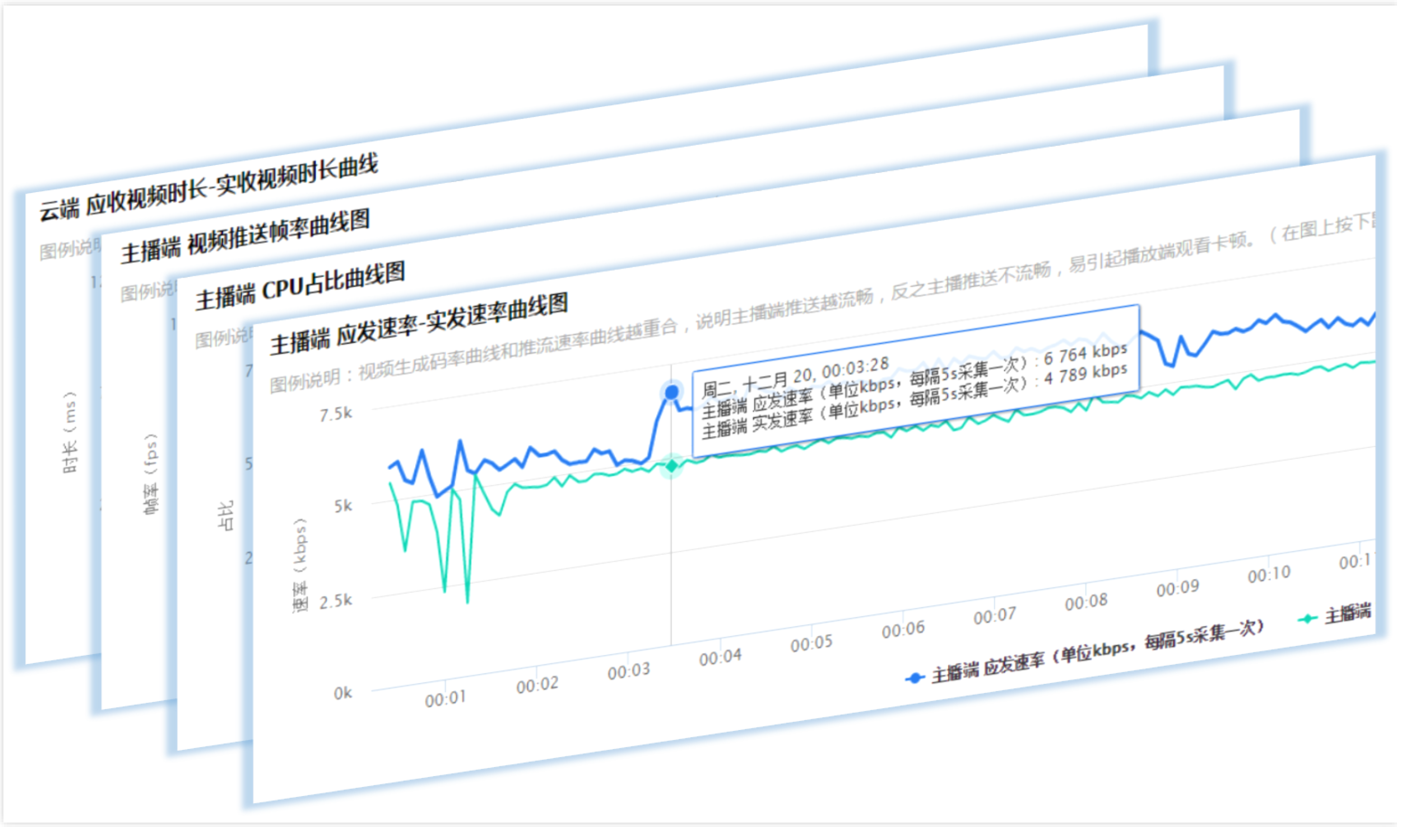
4. Upstream Clog

According to statistics, upstream clog at VJ end is responsible for over 80% of stutters among the video cloud's customers.

4.1 Identify upstream clog

- **4.1.1: Relation between BITRATE and NET_SPEED**
  BITRATE (= VIDEO_BITRATE + AUDIO_BITRATE) refers to the number of audio/video data bits produced by the encoder for push per second; NET_SPEED refers to the number of data bits pushed actually per second. A long duration of BITRATE == NET_SPEED means a good push quality. However, a long duration of BITRATE >= NET_SPEED indicates a bad push quality.
- **4.1.2: CACHE_SIZE and DROP_CNT values**
  Once BITRATE >= NET_SPEED, the audio/video data produced by the encoder will build up on VJ's phone, with the severity indicated by the CACHE_SIZE value. When the CACHE_SIZE value exceeds the warning level, SDK will actively drop some audio/video data, thus triggering an increment of DROP_CNT. The figure below shows a typical upstream clog, with CACHE_SIZE remaining above the **red warning level**. This means that the upstream bandwidth doesn't meet the data transfer requirements (i.e. the upstream network is severely clogged).


2016-09-09 14:12:02
视频平均码率: 785
VIDEO_BITRATE
NET_SPEED
CACHE_SIZE
14:00
14:05
14:10
14:15
14:20
14:25
14:30
14:35
14:40
14:45
14:50
14:55
15:00
15:05


云端 应收视频时长-实收视频时长曲线
时长（ms）
主播端 视频推送帧率曲线图
帧率（fps）
主播端 CPU占比曲线图
占比
主播端 应发速率-实发速率曲线图
图例说明：视频生成码率曲线和推流速率曲线越重合，说明主播端推送越流畅，反之主播推送不流畅，易引起播放端观看卡顿。（在图上按下
周二, 十二月 20, 00:03:28
主播端 应发速率（单位kbps，每隔5s采集一次）: 6 764 kbps
主播端 实发速率（单位kbps，每隔5s采集一次）: 4 789 kbps
速率（kbps）
主播端 应发速率（单位kbps，每隔5s采集一次）
主播端

Note:
The figure similar to the above can be found in **LVB Console** -> **Quality Monitor**.

4.2 Solutions

- **4.2.1 Notify VJ of the bad network condition**
  In a scenario where video quality is important, it's the best practice to notify the VJ through appropriate UI interactions, such as **"The network condition is bad now. Please move closer to your router, and make sure the signal isn't blocked by any wall or obstacle."**
  For more information on how to do this, please see the **Event Handling** section in RTMP SDK's documentation about push. VJs generally are not aware of the upstream clog until receiving a notification from the App or a viewer. Therefore, it is recommended to remind the VJ about the network condition if the App receives multiple **PUSH_WARNING_NET_BUSY** events from RTMP SDK in a short time.
- **4.2.2 Proper encoding settings**
  The following shows the recommended encoding settings (suitable for beauty show LVB. For more information, please see How to Improve Video Quality?). You can set different video quality options using API setVideoQuality of TXLivePush.

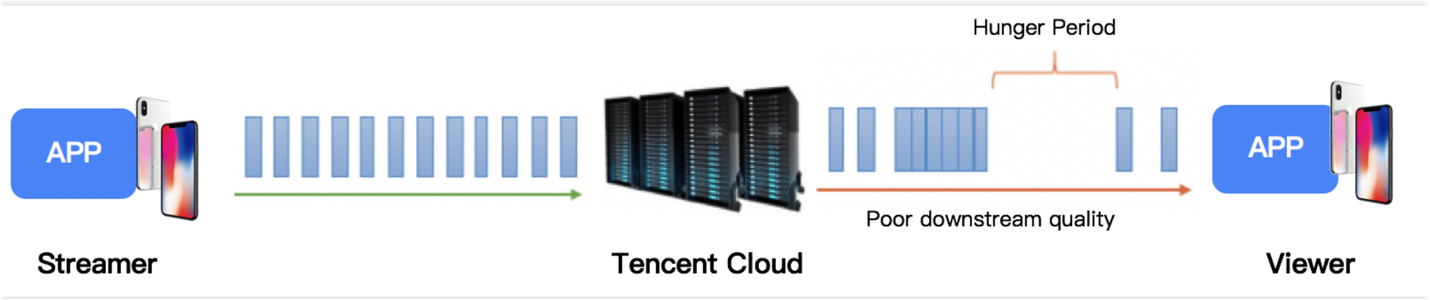
Option	Resolution	FPS	Bitrate	Scenario
Standard Definition	360*640	15	400-800 Kbps	Choose this option if you are more concerned about bandwidth cost. This option can bring a blurred video quality but reduce bandwidth cost by 60% compared to a high definition.
High Definition (recommended)	540*960	15	1200 Kbps	If you're more focused on video quality, select this option, which allows most mainstream mobile phones to present clear pictures.
Ultra High Definition	720*1280	15	1800 Kbps	This option is not recommended for scenarios where videos are mostly viewed in small screens. You can consider using this option if videos are viewed in large screens and the VJ has a great network.

- **4.2.3 Enable traffic control**
  Some clients may complain: "Any user can use our App. It's impossible to control their network conditions." If VJs' upstream bandwidth varies significantly, it's recommended to enable network

adaption by referring to the documentation for iOS and Android. However, the solution described in **4.2.1 Notify VJ of network condition** is the preferred one. After all, it's unrealistic to ensure the smoothness and high definition while achieving a high upstream bandwidth.

Field	Definition	Recommended Value
videoBitrateMin	The minimum video bitrate	400
videoBitrateMax	The maximum video bitrate	1,000 (the recommended value depends on the resolution)
enableAutoBitrate	Bit rate adaption	Enabled
autoAdjustStrategy	Bit rate adjustment strategy	AUTO_ADJUST_BITRATE_STRATEGY_2

5. Player Optimization



5.1 Stutter & lag

As shown in the figure above, instable downstream network or insufficient downstream bandwidth could cause **starvation periods** during playback. During these periods, the App doesn't receive enough audio and video data to play. To minimize the incidence of stutters at viewer end, you need to make the App cache as much video data as possible to survive the "starvation periods". However, caching too much audio and video data brings a new problem - **high delay**, a bad news for scenarios with a high requirement for interactions between VJ and viewers. Moreover, if the delay caused by stutter goes uncontrolled without any correction, it could **accumulate over time** (i.e. the longer the playback lasts, the higher the delay). Delay correction is a key indicator of an excellent player. Therefore, **delay and smoothness are the two ends of a balance**. Focusing too much on low delay will lead to slight network fluctuations that produce significant stutter at the player side. Conversely, overemphasis on smoothness will cause high delay. A typical case is the HLS (m3u8) protocol, which ensures a smooth playback experience by introducing a delay of 10-30 seconds.

5.2 Solutions

To allow you to get a better playback experience without the need to have much knowledge about traffic control and processing, Tencent Cloud RTMP SDK, being optimized over several versions, features a set of automatic adjustment technologies, based on which three excellent delay control schemes are introduced:

- **Auto**: Use this mode if you are unsure about what is your main scenario.

  > You can enter the Auto mode by turning on the setAutoAdjustCache switch in TXLivePlayConfig. In this mode, the player will automatically adjust delay based on current network conditions (by default, the player will automatically adjust delay within the range of 1s-5s. You can use setMinCacheTime and setMaxCacheTime to modify the default) to minimize the delay between VJ and viewers while ensuring sufficient smoothness, and thus to deliver a good interactive experience.

- **Speedy**: Suitable for **live shows** and other scenarios with a high requirement for delay.

  > Speedy mode (set by making **SetMinCacheTime = setMaxCacheTime = 1 second**) and Auto mode only differ in MaxCacheTime value (generally, MaxCacheTime is lower in Speedy mode and is higher in Auto mode). This flexibility can be largely attributed to the automatic control technology within the SDK, which automatically adjusts delay without causing stutter. MaxCacheTime is used to indicate the adjustment speed - the higher the MaxCacheTime value is, the more conservative the adjustment speed is, and therefore the lower the probability of stutter becomes.

- **Smooth**: Suitable for **Game LVB** and other HD (high bitrate) LVB scenarios.

  > You can enter the Smooth mode by turning off setAutoAdjustCache switch in the player. In this mode, the player uses a processing strategy similar to the caching strategy of the Adobe Flash kernel. When stutter occurs in a video, the video will go into the loading status until the cache is full; then it will go into the playing status until the next network fluctuation that can't be resisted. By default, the cache time is 5 seconds, which you can change using setCacheTime.

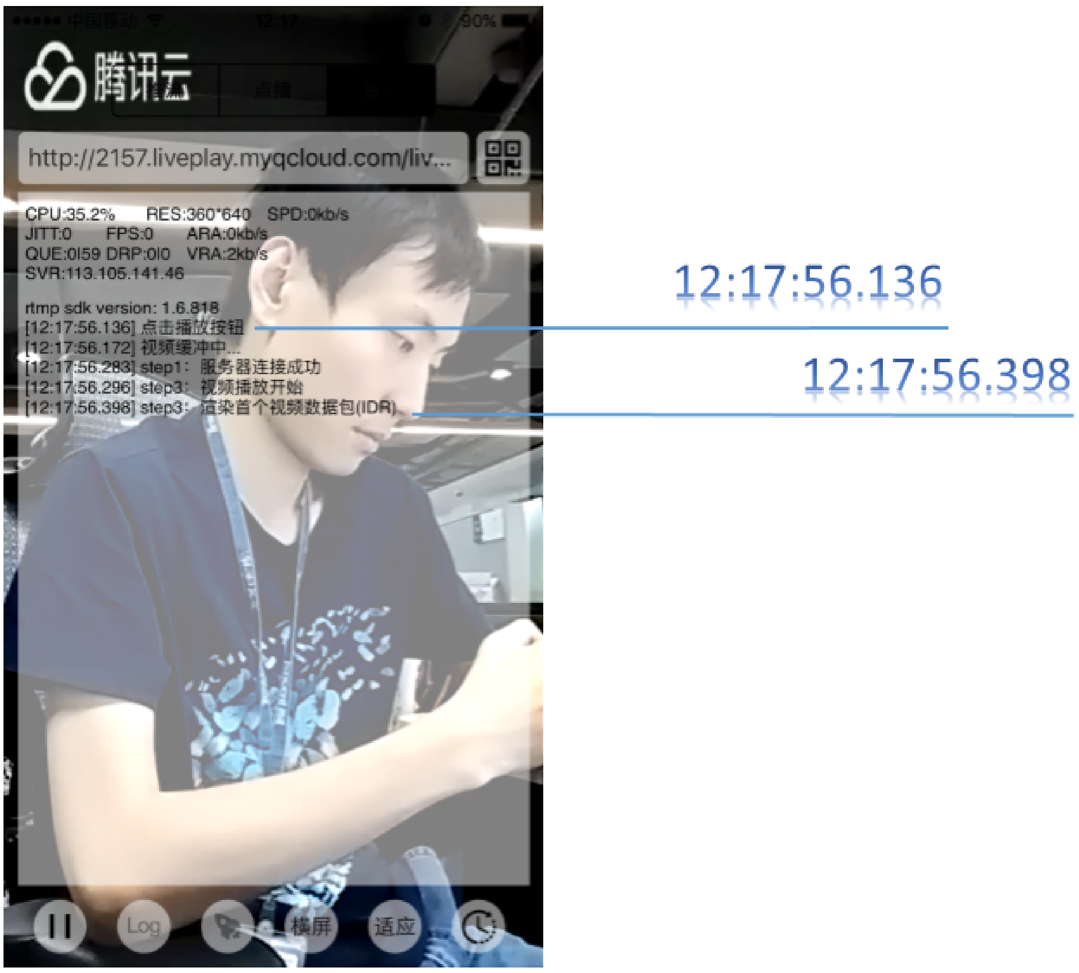
> This seemingly simple mode will be more reliable in scenarios with a low requirement for delay, because the mode in essence trades off delay slightly for a reduced stutter rate.

Instant Broadcasting

Last updated：2018-07-10 14:57:29

What is "Instant Broadcasting"?

Instant Broadcasting is to make the time length between the start of video playback and the moment the first frame is displayed as short as possible (in hundreds of milliseconds) to prevent viewers from waiting long.



```
CPU:35.2%     RES:360*640   SPD:0kb/s
JITT:0      FPS:0      ARA:0kb/s
QUE:0|59 DRP:0|0   VRA:2kb/s
SVR:113.105.141.46

rtmp sdk version: 1.6.818
[12:17:56.136] 点击播放按钮
[12:17:56.172] 视频缓冲中...
[12:17:56.283] step1: 服务器连接成功
[12:17:56.296] step3: 视频播放开始
[12:17:56.398] step3: 渲染首个视频数据包(IDR)
```

This depends on optimized cloud services and player. In an LVB, a combination of Tencent Cloud Audio/Video SDK and video cloud service allows you to open the first frame at a speed as high as about **200ms**, and even open it instantly if you have sufficient downstream bandwidth.

How to Achieve Instant Broadcasting?

Apps

Use Tencent Cloud Audio/Video SDK and the FLV playback protocol to achieve instant broadcasting:

- **HTTP + FLV playback protocol**
  The HTTP + FLV playback protocol is the most widely used protocol in the LVB industry. Thanks to its simple data format, it allows immediate access to Audio/Video data once the server is connected. In contrast, the RTMP protocol provides a slightly lower first frame rate than the FLV protocol due to the several negotiation handshakes required to establish a connection.
- **Tencent Cloud Audio/Video SDK**
  The way instant broadcasting works on cloud is quite simple. With a group of GOP pictures (containing at least a key frame for decoding) always cached on a server, the player can obtain a key frame (I frame) once it is connected to the server, and then proceed with decoding and playing. Such caching on cloud, however, has some side effects: the player is often suddenly stuffed with several seconds of audio/video data after it is connected to a server, which will cause a major delay on the player end. We call it "side effect of instant broadcasting".
  Besides instant broadcasting, a good player should also have excellent **delay correction** ability that automatically corrects player end delay to a proper level (such as less than 1 second) without affecting the viewing experience. Tencent Cloud Audio/Video SDK is a great way to achieve this, even allowing you to specify a delay correction mode for a player (iOS/Android).

PC browsers

The video playback kernels on PC browsers usually use the Flash control (Chrome also supports MSE, but it has no obvious advantage over Flash). Flash players adopt a rigid **forced buffering** strategy, providing little room for optimization of video loading speed, which is unlikely to be kept within 1 second. This fact can be found on many video websites and LVB platforms when you're using a PC browser.

Mobile browsers

- **iPhone**
  Safari works perfectly with HLS (m3u8) and even allows using iPhone's hardware decoding chip to facilitate video playback. Usually, you do not have to worry about the video loading speed as long as DNS caches are available, but this is limited to the iOS platform.
- **Android**
  Due to the severe fragmentation, the performance on the Android platform varies greatly with the system browsers that come along with the various OS versions and devices. The browsers within QQ and WeChat even have Tencent's X5 kernel installed, which would result in large performance variations.

Reduce Latency

Last updated：2018-07-10 15:02:22

Generally, the delay of RTMP push+FLV playback is about 2-3 seconds. An extra-long delay indicates an exception. In case of an extra-long delay, perform the troubleshooting by following steps below:

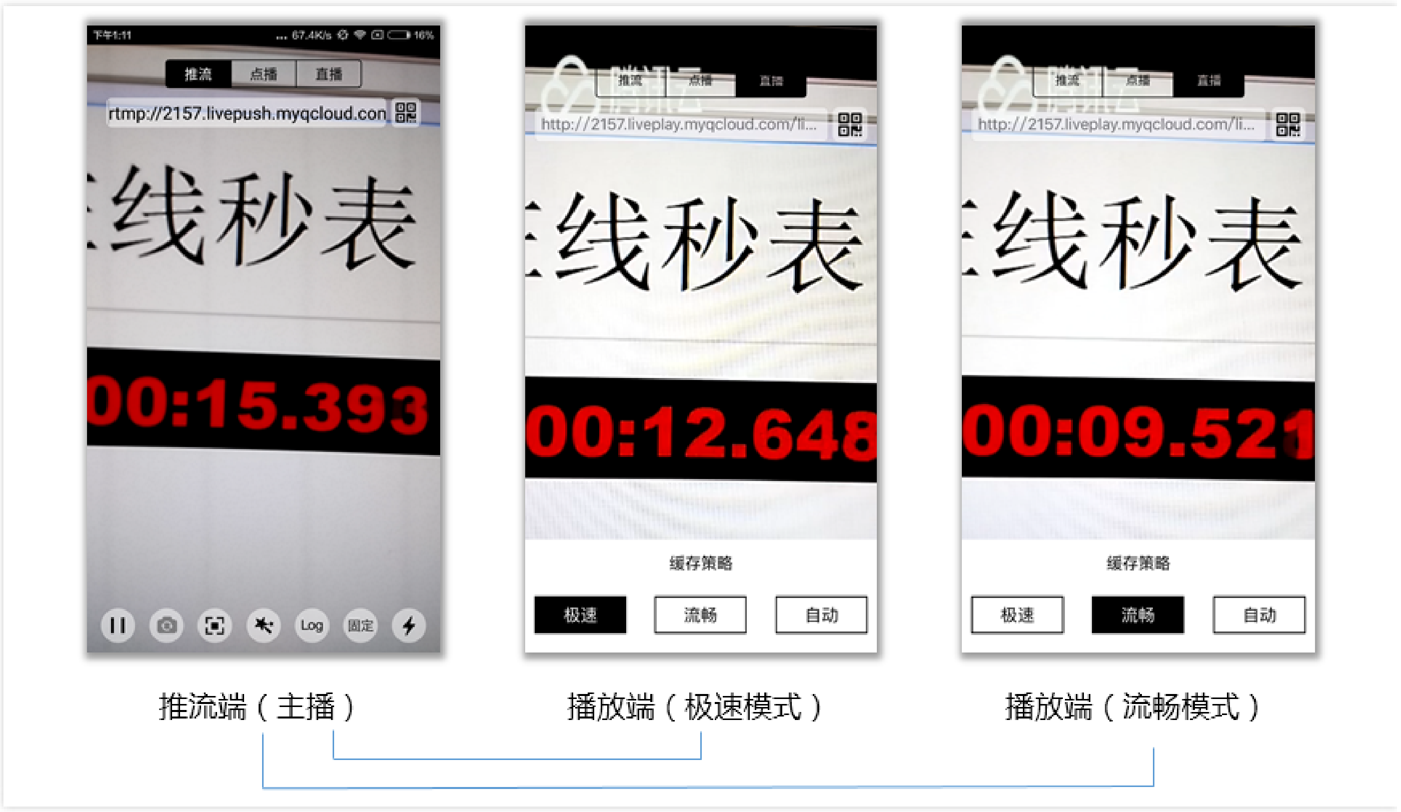
Step 1. Check the playback protocol

It is quite normal that many customers using HLS (m3u8) as the playback protocol experience a longer delay. Apple's HLS is a streaming protocol based on a total of 3-4 large-granular TS fragments, with each fragment featuring a time period of more than 5 seconds. Therefore, it is not unusual that the total delay may reach about 20-30 seconds.
To solve the problem, use the FLV protocol instead. Please note that only HLS (m3u8) playback protocol can be selected if you want to watch LVB on mobile browsers. Other LVB protocols are not supported on Apple's Safari browser.

Step 2. Check the player settings

Tencent Cloud RTMP SDK's player supports Speedy, Smooth and Auto modes. For more information about the settings, please see adjusting delay.

- **Speedy**: With a delay within 2-3 seconds in most scenarios, this mode is suitable for Beauty Show LVB.
- **Smooth**: With a delay within 5 seconds in most scenarios, this mode is suitable for scenarios that are insensitive to delay but have a high requirement for smoothness (such as Game LVB).



Step 3. Disable watermarking in background

Tencent Cloud supports watermarking in background to cater for the customers who cannot use Tencent Cloud RTMP SDK's pusher (supporting watermarking for live streaming) but have a need for watermarking. However, this solution will introduce an extra three second delay. If you are using the Tencent Cloud RTMP SDK to push streams, disable the watermarking in background and then perform watermarking on the App of VJ.

Step 4. Third-party pusher

The desired effect can be ensured only when Tencent Cloud integrated solution is used. Many third-party pushers deal with insufficient upstream bandwidth through unbounded buffer. If you're using a third-party pusher, you're recommended to use Tencent Cloud RTMP SDK's push Demo to make a comparison to eliminate the possibility that the third-party pusher causes an extra-long delay due to the encoding buffer.

Step 5. Check OBS settings

Many customers who use OBS for push report a long delay at viewer end. It is recommended to configure parameters as described in push on PC. Be sure to set the key frame interval to 1 or 2.

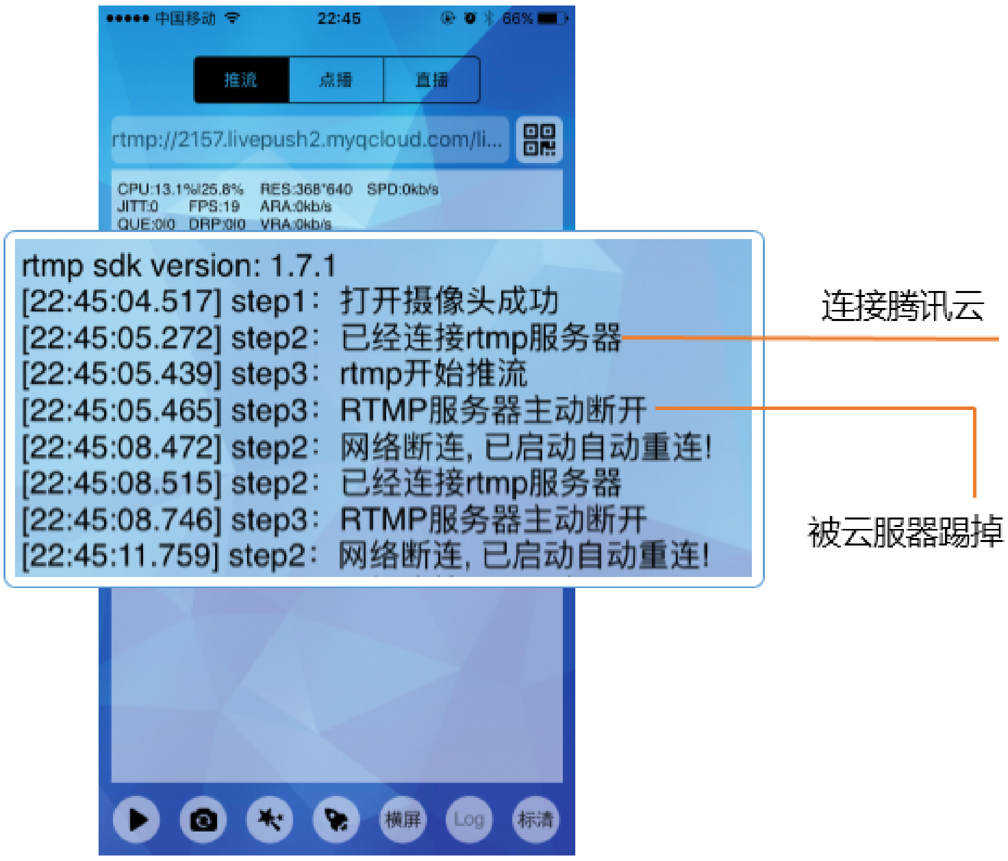
Pushing Failure

Last updated：2018-07-24 16:21:41

Tencent Cloud's customer complaints about unsuccessful push are mainly caused by the following three reasons:

Incorrect txSecret

Tencent Cloud requires adding Hotlink protection to all push URLs to ensure security. Miscalculated hotlink protection or expired push URLs will be **rejected** by Tencent Cloud. In this case, RTMP SDK will throw a **PUSH_WARNING_SERVER_DISCONNECT** event, and RTMP SDK DEMO behaves as follows:



```
CPU:13.1%|25.8%   RES:368*640   SPD:0kb/s
JITT:0       FPS:19     ARA:0kb/s
QUE:0|0     DRP:0|0    VRA:0kb/s
```

```
rtmp sdk version: 1.7.1
[22:45:04.517] step1： 打开摄像头成功
[22:45:05.272] step2： 已经连接rtmp服务器
[22:45:05.439] step3： rtmp开始推流
[22:45:05.465] step3： RTMP服务器主动断开
[22:45:08.472] step2： 网络断连，已启动自动重连!
[22:45:08.515] step2： 已经连接rtmp服务器
[22:45:08.746] step3： RTMP服务器主动断开
[22:45:11.759] step2： 网络断连，已启动自动重连!
```

Please see How to Get the Push URL to learn how to get a reliable push URL.

Expired txTime

Some customers who worry about their LVB traffic being hacked would set a shorter txTime, such as 5 minutes after the current time. In fact, with txSercet signature, you don't need to set such a short validity period. Furthermore, with a too-short validity period, in case of a network interruption during LVB, the VJ

would not be able to resume push due to the expiration of push URL.
It's recommended to set the txTime to a value that is 12 or 24 hours after the current time, making the validity period longer than a normal LVB duration.

Push URL is in use already

A push URL can only be used by one pusher, and any other client attempting to push will be rejected by Tencent Cloud. In this case, RTMP SDK throws a **PUSH_WARNING_SERVER_DISCONNECT** event.

Unable to connect to CVM

The default port number used by RTMP push is **1935**. If the firewall of the network for your test doesn't open the port 1935 to the Internet, you may be unable to connect to the CVM. In this case, you can verify whether the problem is caused by this reason by changing network (for example, use 4G instead).

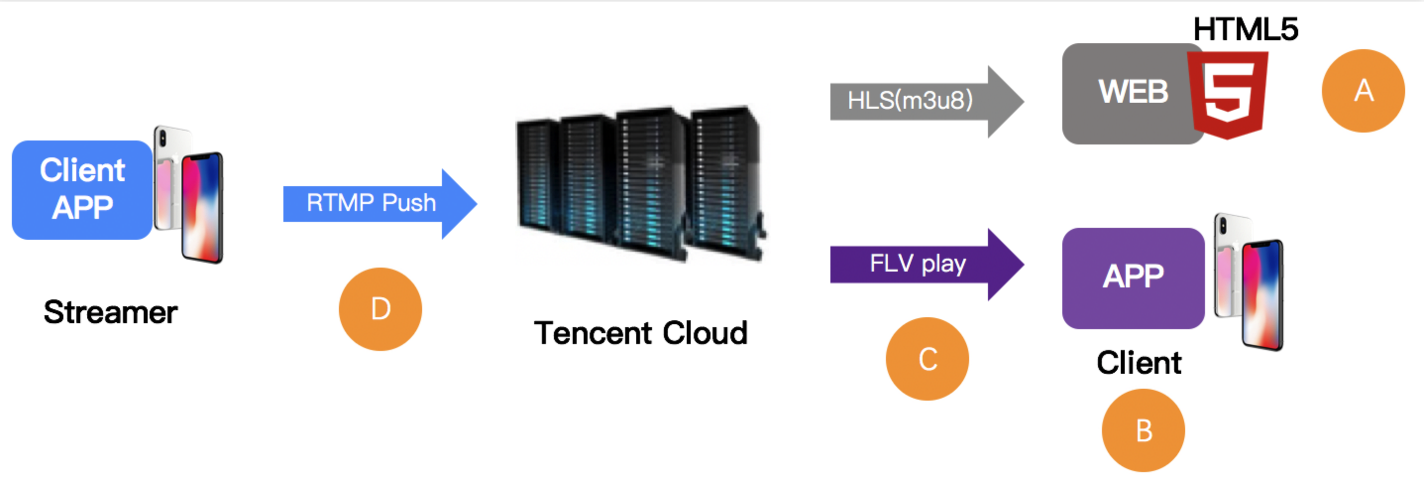
Push URL for Mini LVB

The push URL for Mini LVB can be obtained through debugging. You can search for the keyword **startPush** in the global search, then set a debugging breakpoint, where the RTMP SDK is called by Mini LVB. The parameter startPush is the push URL.

Watching LVB Failure

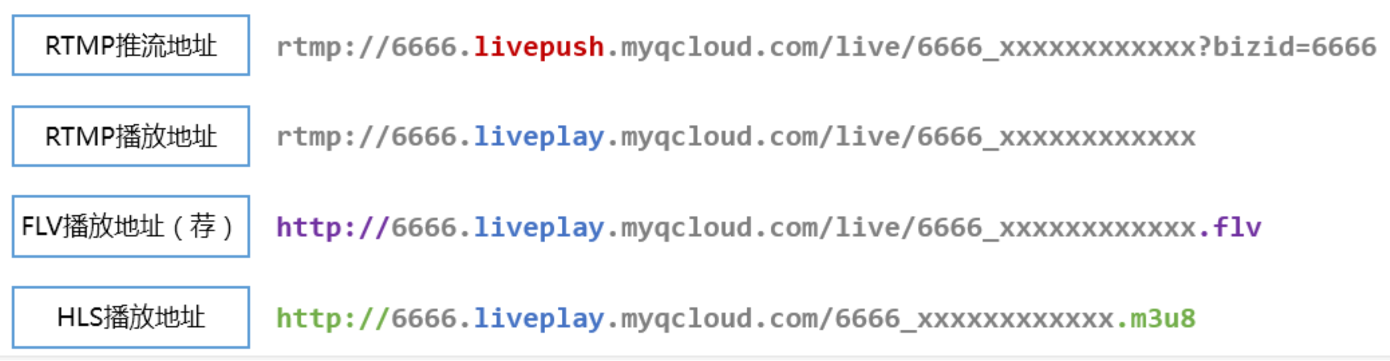
Last updated：2018-07-24 16:24:15

If you're unable to watch the LVB and have no idea what goes wrong with it, you can identify the cause of the problem in a short time by following the steps below:



Step 1. Check the playback URL

First of all, check whether the playback URL is correct. An incorrect URL is the most likely cause of most problems. Tencent Cloud's LVB URLs include push URL and playback URL. You need to first verify whether **the push URL is accidentally used as the playback URL**.



RTMP推流地址	`rtmp://6666.livepush.myqcloud.com/live/6666_xxxxxxxxxxx?bizid=6666`
RTMP播放地址	`rtmp://6666.liveplay.myqcloud.com/live/6666_xxxxxxxxxxx`
FLV播放地址（荐）	`http://6666.liveplay.myqcloud.com/live/6666_xxxxxxxxxxx.flv`
HLS播放地址	`http://6666.liveplay.myqcloud.com/6666_xxxxxxxxxxx.m3u8`

Playback URL for Mini LVB:
The playback URL for Mini LVB can be obtained through debugging. You can search for the

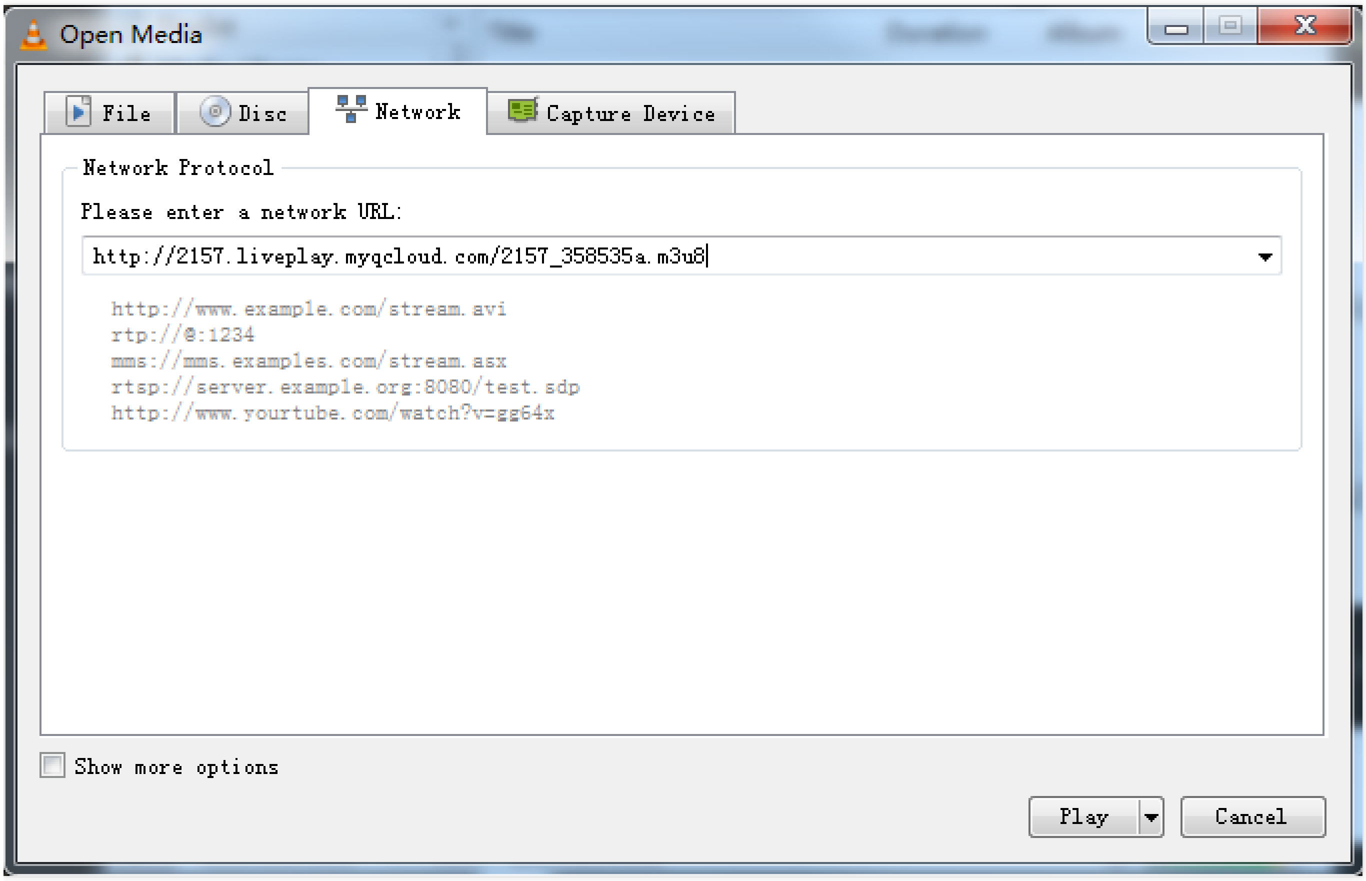
keyword **startPlay** in the global search, then set a debugging breakpoint, where the RTMP SDK is called by Mini LVB. The parameter startPlay is the playback URL.

Step 2. Check the video stream

A correct playback URL does not always mean a normal playback. Next, you need to check whether the video stream is normal:

- In **LVB**, the LVB URL becomes unavailable once the VJ stops the push.
- In **VOD**, if the video files have been removed, watching videos is also impossible.

A frequently used solution is making a check using VLC, an open-source player on PC that supports many protocols.



Please enter a network URL:

http://2157.liveplay.myqcloud.com/2157_358535a.m3u8

```
http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x
```

Step 3. Check the player

If there's no problem with the video stream, then you need to check whether the player is normal on a case-by-case basis:

3.1 Web browser (A)

- **Format**: Mobile browsers only support playback URLs in **HLS (m3u8) and MP4** formats.
- **HLS (m3u8)**: Tencent Cloud HLS protocol is based on "Lazy Start". In short, Tencent Cloud only starts the transcoding for HLS format when a viewer requests a playback URL in an HLS format. The purpose is to prevent waste of resources. But it also creates a problem: **The playback URL in an HLS format cannot be played until 30 seconds after the first user in the world initiates a request**.
- **Tencent Cloud Web player:** Supports playback URLs based on multiple protocols, and adopts the optimal playback policy based on the current platform (PC/Android/iOS). The internal selective retry logic can also deal with the Lazy Start of HLS (m3u8).

3.2 RTMP SDK (B)

If RTMP SDK DEMO works normally for playback, it's recommended to check whether the interfacing logic is incorrect by referring to the RTMP SDK playback document (iOS & Android).

Step 4. Check for firewall blocking (C)

It is common that the corporate network environments of many customers restrict video playback through firewalls that detect whether the resources requested by HTTP are streaming media resources (After all, no boss wants his employees to watch videos during working hours). The fact that you can watch the LVB normally over 4G network but cannot watch it over your company's Wi-Fi network indicates your company has imposed restrictions on the network policies. In this case, contact the administrator for a special treatment of your IP.

Step 5. Check the pusher (D)

If the LVB URL does not work and there is no possibility of firewall blocking described in Step 4, it is likely that the push is unsuccessful. Go to Why the Push is Unsuccessful for a further troubleshooting.

Increase Definition

Last updated：2018-07-10 14:56:58

Scenario 1: Beauty LVB

Step 1: Update SDK to the latest version

The beauty filter effects are optimized with each release of new version of SDK. For example:

- In 1.9.1, the beauty filter engine was updated, and many improvements were made to the foreground focus, algorithm, exposure and performance.
- In 1.9.2, the noise reduction effect was optimized to greatly reduce the noises at night and improve the clarity of person images in videos.
- In 2.0.0, more filters were provided for iOS to improve the visual effect of yellowish skin tone.
- ... ...

Step 2: Configure image quality

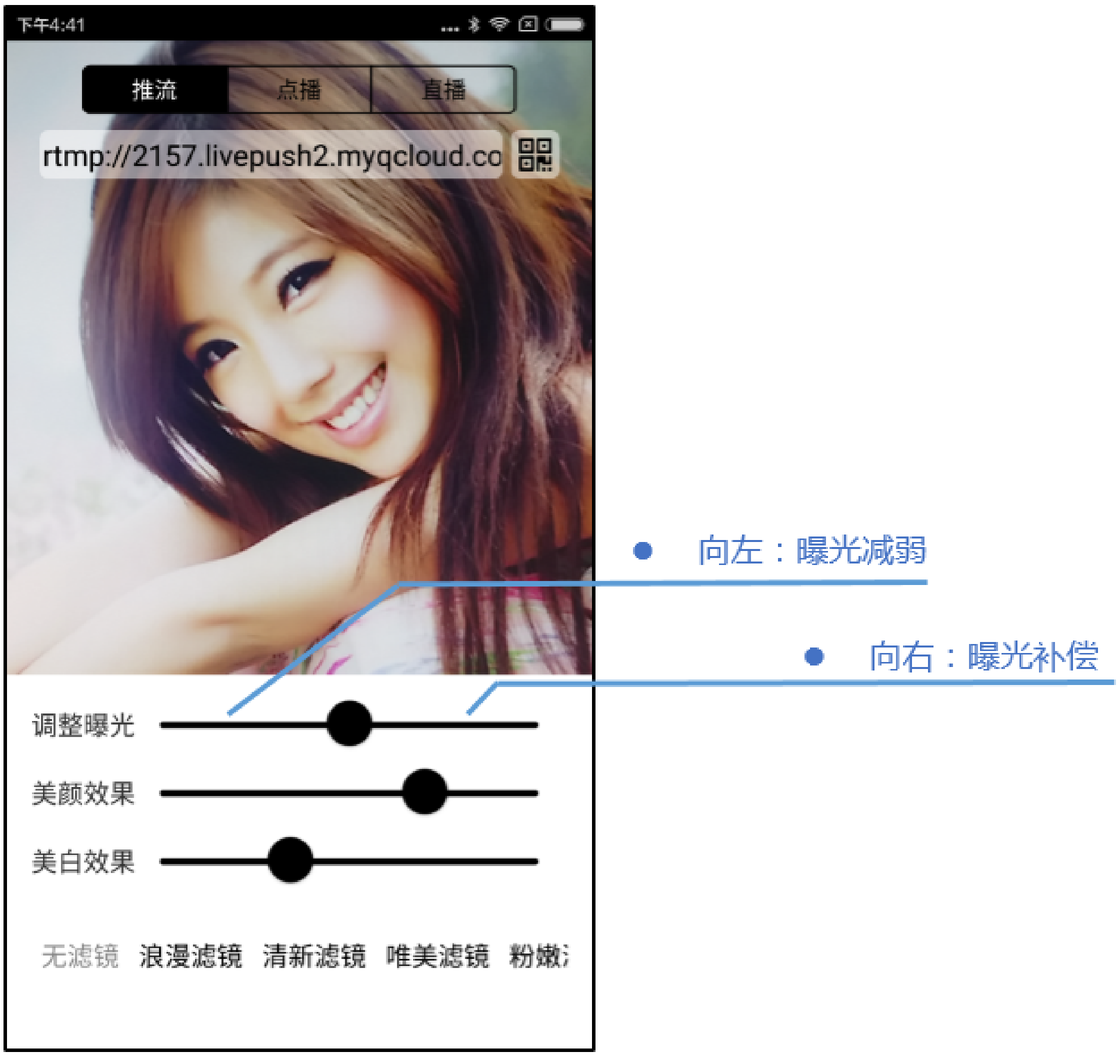
The video image quality displayed to VJs is different from that displayed to viewers:

- **VJ vs Viewer**:
  The video images seen by a VJ are produced from the captured videos that are directly rendered on the phone screen and thus have the best clarity. The images need to go through **Video Encoding** > **Network Transmission** > **Video Decoding** before being rendered on the phone screen of viewers. The video encoding can affect the image quality, so the images displayed to viewers are not as clear as the ones displayed to the VJ. Inappropriately configured parameters can greatly reduce the image quality. A typical example is "high resolution plus low bitrate", which can cause blurred display and serious mosaics.
- **setVideoQuality**
  In 1.9.1, the function setVideoQuality was added in TXLivePusher with a range of levels available. You can have the best image quality in beauty LVB by simply choosing **HD** mode. For more information, please see iOS Platform and Android Platform.

Step 3: Add manual exposure on Android

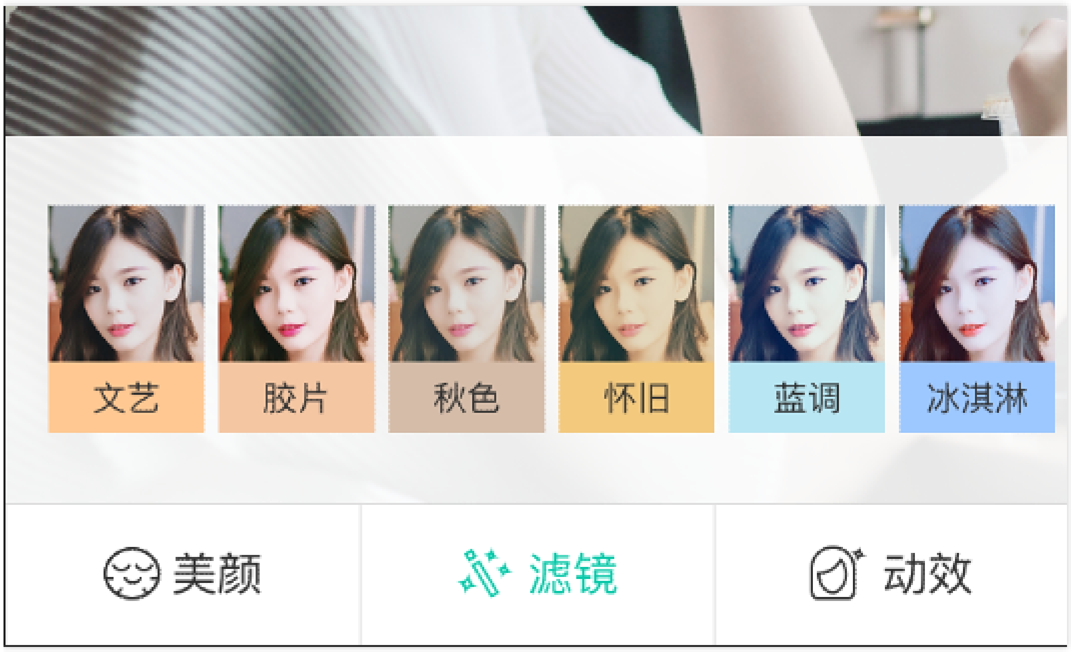
The same beauty filter algorithm may yield different effects on different Android phones. This is because the difference between various devices in terms of exposure performance leads to different visual effects. On iOS devices, auto-exposure is adopted. But Android devices are significantly different from iOS ones, and some low-end Android phones have an unsatisfactory auto-exposure effect. Therefore, it is recommended to provide a slider on the page for VJs to adjust exposure value manually as needed.

The API TXLivePush::**setExposureCompensation** in the Android RTMP SDK can be used to adjust exposure, with the parameter value being a float value between -1 and 1: 0 - no adjustment; -1 - minimum exposure; 1 - maximum exposure.



Step 4: Add color filters

Filters are also important because different color filters create different atmosphere. A VJ can choose a filter to match his/her clothes or room light to create a better visual effect.



Color filters have been supported as of RTMP SDK 1.9.1. The setFilter added in TXLivePusher can be used to set filter effect. Eight sets of color filter materials are made available in the Demo. You can use them on a royalty-free basis.

Step 5: Serious mosaics on Android

Some customers find that the images pushed by the Android RTMP SDK have serious mosaics, especially for dynamic images. This is a common problem created by Android hardware encoding, and you can solve it in two ways:

- **For a lower battery drain**
  If you care more about the App's battery drain, increase the bitrate of the pushed streams or use **HD** in setVideoQuality (if you set a low bitrate, the Android's hardware encoding module ensures a consistent bitrate by greatly reducing the image quality).
- **For a lower bandwidth cost**
  If you care more about the bandwidth cost, increasing bitrate may not be a good solution. Instead, you can solve this problem by disabling hardware acceleration. For more information, please see setHardwareAcceleration.

Step 6: Disable network adaption

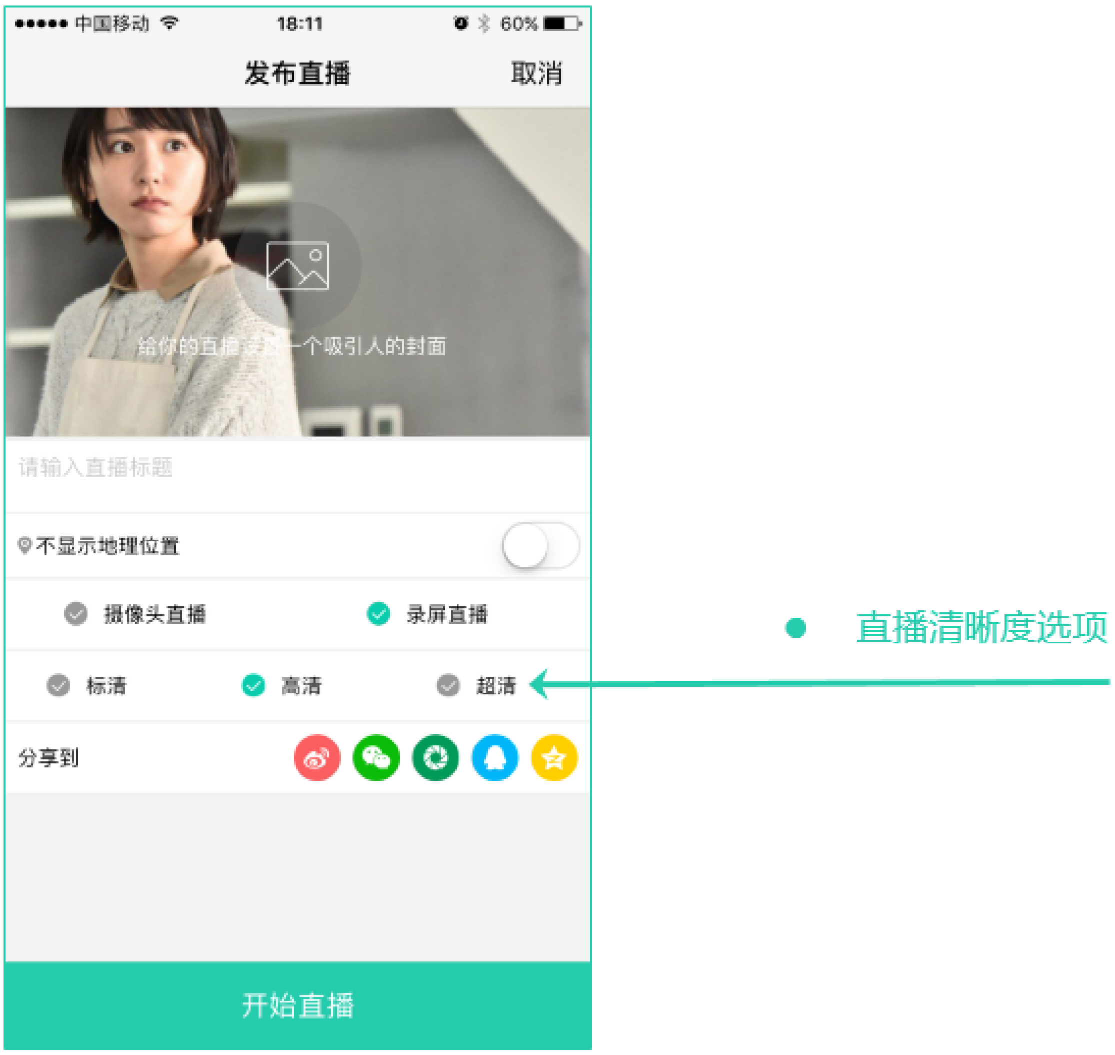
The **AutoAdjustBitrate** in **TXLivePushConfig** is used to enable/disable network adaption. If it is enabled, the image quality is reduced to ensure smoothness in case of a poor network of VJ. But this feature is **not suitable** for beauty shows. Network adaption is suitable for game LVB scenarios, where viewers attach more importance to smoothness than image quality. If the VJ's network becomes unstable during a battle, it's ok to have a degraded image quality but stutters are totally unacceptable, so it is necessary to trade

off image quality for smoothness (frame rate). However, image quality is more important in beauty show scenarios. Many customers report that the image quality varies greatly with different rooms. This is likely caused by the enabling of network adaption.
We recommend that you disable network adaption and deal with network fluctuations by using system alerts to solve the problem more fundamentally.

Scenario 2: Game LVB

Option 1: Simple approach

Provide three definition options on the LVB starting page - SD, HD and UHD - **for VJs to select from**. A VJ in game LVB can find out which option is suitable for the game he/she is playing. The configurations for the three options are as follows:



Option	Resolution	FPS	Bitrate

Option	Resolution	FPS	Bitrate
SD	VIDEO_RESOLUTION_TYPE_360_640	20	800 kbps
HD	VIDEO_RESOLUTION_TYPE_540_960	20	1000 kbps
UHD	VIDEO_RESOLUTION_TYPE_720_1280	20	1800 kbps

Note:
The minimum FPS in game LVB scenarios is 20. Serious stutters can occur at viewer end in case of an FPS less than 20.

Option 2: Professional approach

Configure different resolutions and bitrates for different games. For example:

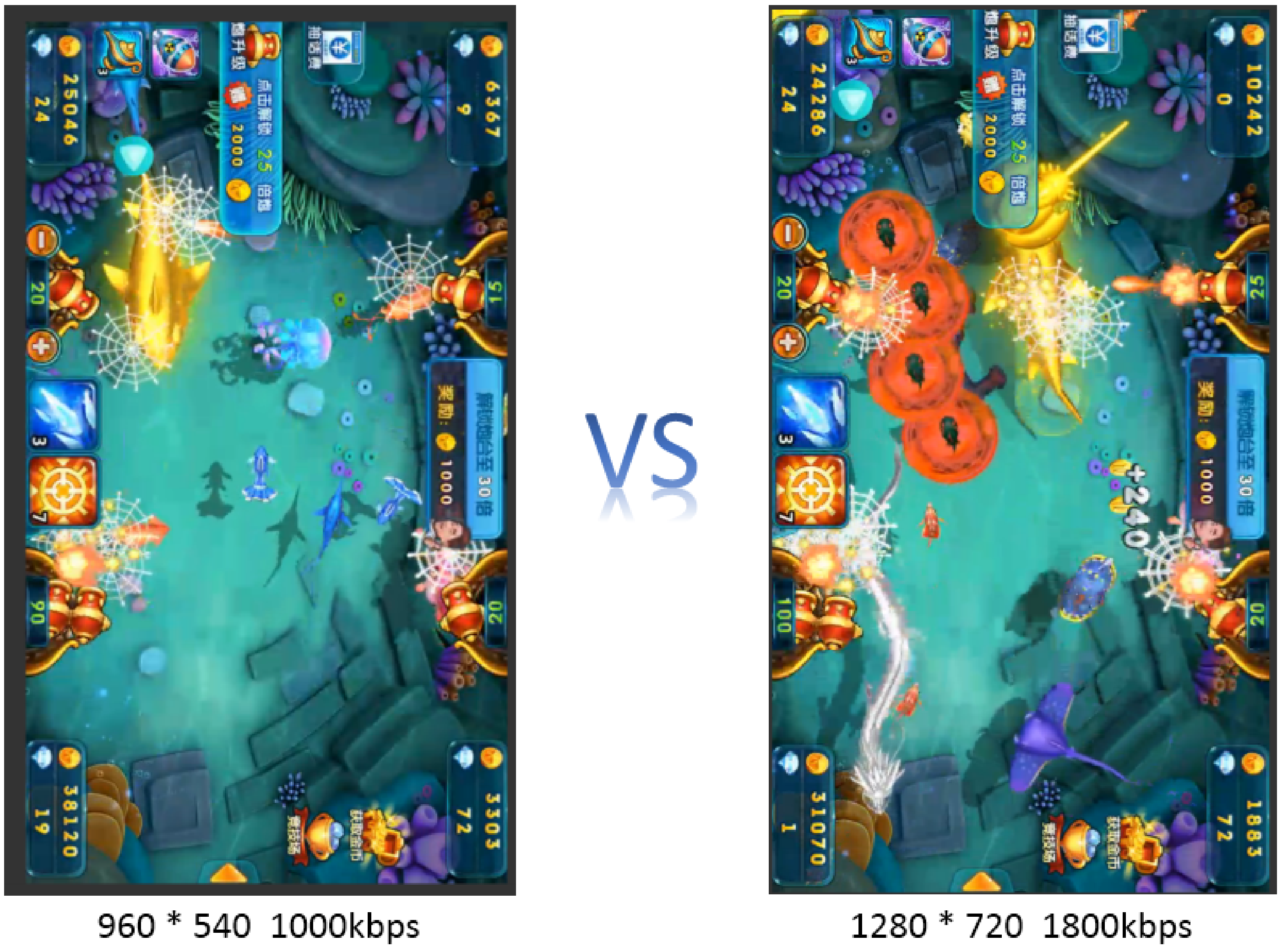
- **Clash Royale** - For such a game that features less dynamic images, the combination of a resolution of **960 * 540** and a bitrate between 800 kbps and 1000 kbps can produce a good effect.
- **Fishlord** - For such a game that features more dynamic images, the combination of a resolution of **960 * 540** and a higher bitrate between 1200 kbps and 1500 kbps is recommended.
- **Temple Run** - For such a game that features highly dynamic images, it is recommended to choose a resolution of **640 * 360** and a very high bitrate, for example, 2000 kbps, to avoid serious mosaics.

Tips on Audio/Video

1: A resolution of 720p does not necessarily mean a higher clarity

For a given bitrate, for example, 800 kbps, **a higher resolution will make it harder for an encoder to deliver a good image quality**. The encoder can support sufficient pixels only by decreasing color elements or introducing mosaics. For the same movie file sized at 2 GB, a 1080p resolution may render less clear images than a 720p resolution.
If the viewers watch videos on small phone screens, they won't see much difference between **960 * 540 1000 kbps** and **1280 * 720 1800 kbps**. For instance, the two images below are captured from screencap

LVB on iOS using the airplay technology:



Note:
You will see the difference if the images are displayed in full-screen mode on a 32-inch LCD screen.

2: Keep the FPS within 24

For a given bitrate, for example, 800 kbps, a higher FPS makes it necessary for the encoder to increase the compression ratio for each frame, which means reducing the image quality to support enough frames. If the video source is camera, then 24 FPS is the maximum frame rate for naked eyes. Therefore, an FPS of 20 is already enough to offer a good user experience. Some 3D game players may ask: "Does a higher frame rate, such as 60 or 120 FPS, mean a higher smoothness? "
It depends on the scenarios: In a game scenario, a higher **rendering frame rate** is recommended to make the motion effects rendered with 3D models more similar to the motion trajectories in the real world. But a high frame rate is not needed for capturing. For example, what are captured by a phone camera are the

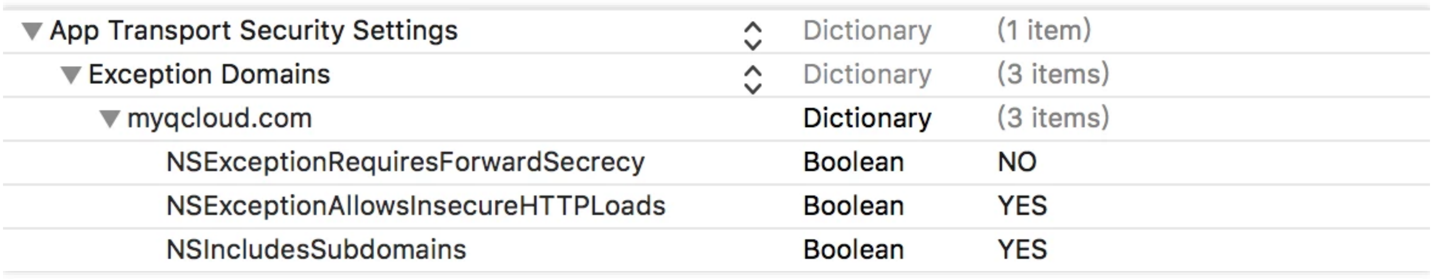
objects in reality, which are in motion continuously and are not simulated through refreshes of images, so 20 FPS is enough.
For game LVB, an FPS of 24 is ideal, but you also need to consider such factors as system encoding cost, phone temperature, and CPU utilization.

Compatibility with Apple ATS

Last updated：2018-07-11 10:11:37

Apple Inc. announced in WWDC 2016 that by default all new Apps submitted as of January 1, 2017 are not be allowed to use `NSAllowsArbitraryLoads=YES` to bypass ATS restrictions. Tencent Cloud will officially support HTTPS as of December 12. By then, you only need to use the new version of SDK (APIs remain unchanged) and change the video URL prefix from `http://` to `https://` . The new SDK can be automatically adapted to the change.

Please note that compared with HTTP, HTTPS provides a higher security (which is not absolutely necessary for videos), but leads to a lower connection speed and a higher CPU utilization. If HTTP is still required for your App under Apple's new policy, you need to modify Info.plist by adding `myqcloud.com` to `NSExceptionDomains` , as shown below.



▼ App Transport Security Settings	Dictionary	(1 item)
▼ Exception Domains	Dictionary	(3 items)
▼ myqcloud.com	Dictionary	(3 items)
NSExceptionRequiresForwardSecrecy	Boolean	NO
NSExceptionAllowsInsecureHTTPLoads	Boolean	YES
NSIncludesSubdomains	Boolean	YES

Disabling ATS for specific domain names can be approved by Apple's audit team, but you may need to specify that `myqcloud.com` is a domain name for video playback.

Delete IDs of Channel Hosting Mode & LVB Code Access Mode

Last updated：2018-07-11 10:12:58

1. API for Flow Deletion in Channel Mode

The API is not provided to create flow in LVB Code mode, so there is no API for flow deletion in this mode.

The API is provided to create and delete flow in channel mode. API for flow deletion:
https://cloud.tencent.com/document/product/267/4722

2. Deleting Channel

In channel mode, you can delete a channel on the console, while in the LVB code mode, you are not allowed to delete a channel. The LVB backend automatically deletes LVB code flows that are not pushed in last 7 days. The LVB code id with ongoing flows being pushed will not be deleted.

3. Channel Number Limit

In the LVB code mode, there is no limit on the number of created channels.

In channel mode, a maximum of 50 channels can be created by default. If more channels are needed, please apply for us to dissolve the limit on the backend.

In addition, the monthly concurrent flow pushing increases as the flow pushing increases, thus leading to more channel cost.

Difference Between Stream Interruption and Stream Suspension

Last updated：2018-07-11 10:11:12

1. Stream Interruption

If a stream that is being broadcasted is interrupted, the push stops and viewers are unable to watch the LVB. When a stream interruption occurs, a push can be initiated at the VJ end to resume the LVB.

2. Stream Suspension

If a stream that is being broadcasted is suspended, the push stops and viewers are unable to watch the LVB. When a stream suspension occurs, no push can be initiated at the VJ end during a certain period of time. The end time of suspension can be set on the console and the suspension period cannot be greater than one month.
For example, if stream suspension is enabled for the Stream test123 at 2018-01-01 12:00, and the end time of suspension is set to 2018-01-05-12:00, the VJ for this stream cannot push stream and viewers cannot watch the LVB during this period.