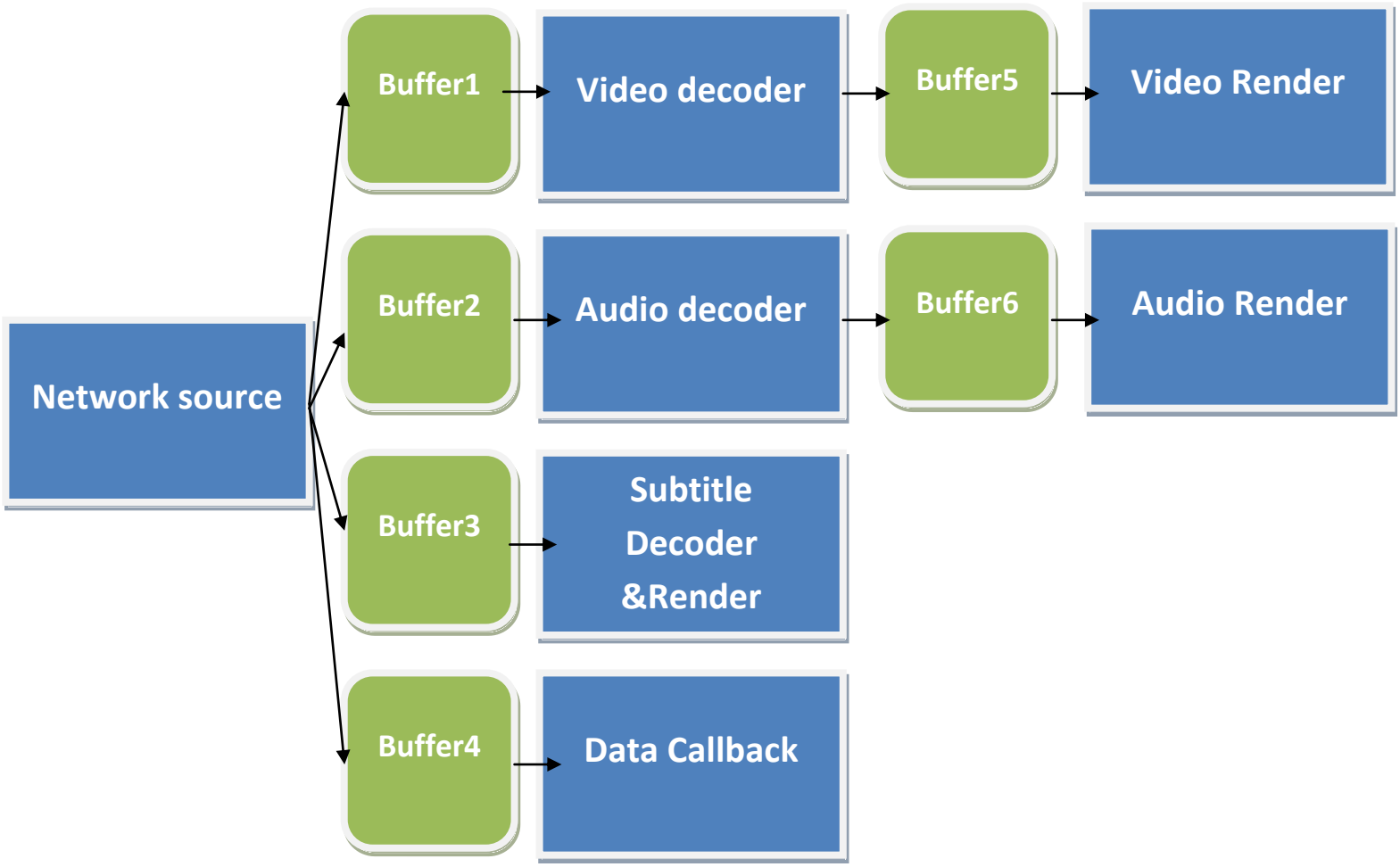
VXG Mobile
Video Player
SDK for Android

(Buffering)

VXG Inc.,

Feb 28, 2017

1. **Pipeline and buffers model**



2. Buffer description

Buffer1 – Video buffer, data format is Video Elementary data.

Buffer size is 8 MB for HD resolution and less and 32 MB for UHD resolution.

Buffer2 - Audio buffer, data format is Audio Elementary data.

Buffer size is 4 MB.

Buffer3 – Subtitle buffer, data format depends on subtitle type.

Buffer size is 2 MB.

Buffer4 – Data buffer, data format depends on data.

Buffer size is 1 MB.

Buffer5 – Video render buffer, index of decoded frame in case of h/w decoder and video frame in case s/w decoder.

Buffer size for s/w decoder is 16 MB for up to HD resolution, UHD – 96MB).

Buffer size for h/w decoder is 1 MB.

Buffer6 – Audio render buffer, decoded sample, format PCM.

Buffer size is 4 MB.

Buffer sizes can be configured for custom platforms.

3. Buffering setting

1) setConnectionBufferingType/getConnectionBufferingType

Set buffering mode
0 – Buffering on start of streaming. The buffer size is estimated in milliseconds (difference between PTS on video and audio channel)
1 – Buffering on start of streaming and during the playback if data in buffer is less than threshold. The buffer size is estimated in bytes for all channels.

Default value is 0.

2) setConnectionBufferingTime/getConnectionBufferingTime

Set buffering size in milliseconds.

Default value is 1000 (1 seconds).

3) setConnectionBufferingSize/getConnectionBufferingSize

Set buffering size in bytes.

Default value is 0.

4. Buffering mode

There are 2 modes for media data buffering:

1. Buffering on start of streaming. The size of the buffer is in milliseconds (default mode)

connectionBufferingType = 0;

connectionBufferingTime is used ; connectionBufferingSize is NOT used

For example :

connectionBufferingTime = 1000;

It means that data will be accumulated on start of streaming, the size of audio and video buffering is 1 second if PTS are correct.

2. Buffering on start of streaming and buffering during playback if video or audio buffers underflow.

connectionBufferingType = 1;

connectionBufferingSize is used ; connectionBufferingTime is NOT used

For example :

connectionBufferingType = 1;

connectionBufferingSize = 1000;

It means that 1 Mbytes of all received data (video, audio and meta data) will be accumulated on start of streaming.

The data will be buffered again during streaming if video or audio buffers become empty. Video and audio decoders and renders are paused during buffering.

5. Buffering status

getInternalBuffersState is used to get last buffer status
Return fullness of inner buffers in pipeline. Detail information are provided for every buffer in video and audio pipeline.

Definition
public BuffersState getInternalBuffersState ()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **getInternalBuffersState()** returns buffers fullness and size with detail information about every buffer, otherwise null is returned. All errors are provided in callback status

Remarks
int getBufferSizeBetweenSourceAndVideoDecoder() :
Size of buffer between Source and Video decoder (Buffer1)

int getBufferFilledSizeBetweenSourceAndVideoDecoder() :
Fullness of buffer between Source and Video decoder (Buffer1)

int getBufferFramesSourceAndVideoDecoder():
Return number of frames between network source and video decoder.

int getBufferSizeBetweenSourceAndAudioDecoder():
Size of buffer between Source and Audio decoder (Buffer2)

int getBufferFilledSizeBetweenSourceAndAudioDecoder():
Fullness of buffer between Source and Audio decoder (Buffer2)

int getBufferSizeBetweenVideoDecoderAndVideoRenderer():
Size of buffer between Video decoder and Render (Buffer5)

int getBufferFilledSizeBetweenVideoDecoderAndVideoRenderer() :
Fullness of buffer between Video decoder and Render decoder (Buffer5)

int getBufferFramesBetweenVideoDecoderAndVideoRenderer():
Return number of frames between video decoder and video render.

int getBufferSizeBetweenAudioDecoderAndAudioRenderer():
Size of buffer between Audio decoder and Render (Buffer6)

int getBufferFilledSizeBetweenAudioDecoderAndAudioRenderer():

Fullness of buffer between Audio decoder and Render decoder (Buffer6)

int getBufferVideoLatency() : return latency by PTS between last received video frame by network source and last rendered video frame by video render.

int getBufferAudioLatency() : return latency by PTS between last received audio sample by network source and last rendered audio sample by audio render.

Example :

```
BuffersState buf_state;
buf_state = player.getInternalBuffersState();
if (buf_state.getBufferSizeBetweenAudioDecoderAndAudioRenderer() != 0 &&
buf_state.getBufferSizeBetweenAudioDecoderAndAudioRenderer() != 0)
Log.e("TEST", "buf_level: Source: " +
buf_state.getBufferFilledSizeBetweenSourceAndAudioDecoder()*100/buf_state.getBuff
erSizeBetweenSourceAndAudioDecoder() + " Render: " +
buf_state.getBufferFilledSizeBetweenAudioDecoderAndAudioRenderer()*100/buf_stat
e.getBufferSizeBetweenAudioDecoderAndAudioRenderer());
```

6. Buffering control

The simplest method to control buffering amd latency is : change playback speed and check buffering values. Below sample function shows how control inner buffer using buffer state and playback speed. Function checks number frames and increase or decrease the playback speed.

```
private void ControlByNumberVideoFrames(MediaPlayer.BuffersState state) {

    if (state != null && mUpperMaxFrames > 0 && mUpperNormalFrames > 0) {
    // Get Number video frames in inner buffers
    int video_frames = state.getBufferFramesSourceAndVideoDecoder() +
    state.getBufferFramesBetweenVideoDecoderAndVideoRenderer() ;
        if (mVideoFrames == 0)
            mVideoFrames  = video_frames;
        else
            //  normalize frames to avoid fast switching in case if network is not stable
            mVideoFrames  = (mVideoFrames + video_frames)/2;

        if (mVideoFrames != 0)
        {
            if (mVideoFrames  >=  mUpperMaxFrames &&
              mNormalSpeed  ==  mSpeed)
            {
                // Set over speed to reduce number of buffered frames and latency
```

```
                    mSpeed = mOverSpeed;
                    mPlayer.setFFRate(mSpeed);
                }
                else if ( mVideoFrames <= mUpperNormalFrames  &&
                    mOverSpeed == mSpeed )
                {
                // Set normal speed in case if number of frames archives  the defined threshold
                    mSpeed = mNormalSpeed;
                    mPlayer.setFFRate(mSpeed);
                }
                else if ( mVideoFrames <= mLowerMinFrames &&
                    mNormalSpeed == mSpeed)
                {
                // Set low speed to accumulate the buffer
                    mSpeed = mDownSpeed;
                    mPlayer.setFFRate(mSpeed);
                }
                else if ( mVideoFrames >= mLowerNormalFrames &&
                    mDownSpeed == mSpeed)
                {
                    mSpeed = mNormalSpeed;
                    mPlayer.setFFRate(mSpeed);
                }
                }
        }
}
```