MediaPlayer stutters at start of mp3 playback

Asked 13 years, 9 months ago Modified 3 years ago Viewed 30k times

Part of Mobile Development Collective

33

I've been having a problem playing an mp3 file stored in a raw resource: when the file first starts playing, it generates perhaps a quarter of a second of sound and then restarts. (I know that this is basically a duplicate of the problem described here, but the solution offered there hasn't worked for me.) I have tried several things and have made some progress on the problem, but it isn't totally fixed.

Here's how I'm setting up to play a file:

```
mPlayer.reset();
try {
    AssetFileDescriptor afd = getResources().openRawResourceFd(mAudioId);
    if (afd == null) {
        Toast.makeText(mOwner, "Could not load sound.",
                Toast.LENGTH_LONG).show();
        return;
    }
    mPlayer.setDataSource(afd.getFileDescriptor(),
            afd.getStartOffset(), afd.getLength());
    afd.close();
    mPlayer.prepare();
} catch (Exception e) {
    Log.d(LOG_TAG, "Could not load sound.", e);
    Toast.makeText(mOwner, "Could not load sound.", Toast.LENGTH_LONG)
            .show();
}
```

If I exit the activity (which calls `mPlayer.release()`) and come back to it (creating a new MediaPlayer), the stutter is usually (but not always) gone—*provided* I load the same sound file. I tried a couple of things that made no difference:

- Load the sound file as an asset instead of as a resource.
- Create the MediaPlayer using `MediaPlayer.create(getContext(), mAudioId)` and skip the calls to `setDataSource(...)` and `prepare()`.

Then I noticed that LogCat always shows this line at about the time that playback starts:

```
DEBUG/AudioSink(37): bufferCount (4) is too small and increased to 12
```

It got me wondering if the stuttering is due to the apparent rebuffering. This led me to try something else:

- After calling `prepare()`, call `mPlayer.start()` and immediately call `mPlayer.pause()`.

To my pleasant surprise, this had a big effect. A great deal of the stutter is gone, plus no sound (that I can hear) is actually played at that point in the process.

However, it still stutters from time to time when I call `mPlayer.start()` for real. Plus, this seems like a huge kludge. Is there any way to kill this problem completely and cleanly?

EDIT More info; not sure if related. If I call `pause()` during playback, seek to an earlier position, and call `start()` again, I hear a short bit (~1/4 sec) of additional sound from where it was paused before it starts playing at the new position. This seems to point to more buffering problems.

Also, the stuttering (and paused buffer) problems show up on emulators from 1.6 through 3.0.

MD android android-mediaplayer playback

Share

Improve this question

Follow

edited Dec 14, 2021 at 21:53

asked Mar 17, 2011 at 18:45

Ted Hopp

235k ● 48 ● 409 ● 530

For those with good memories, calling `start()` and `pause()` like this might remind you of the trick we used to have to do to force pre-loading of an `AudioClip` in a web page applet. :-) – Ted Hopp Mar 17, 2011 at 19:29

Perhaps it is to do with the encoding of the sound file, or possibly the length of it. Have you tried using your code with a different sound file? – Joseph Earl Mar 21, 2011 at 23:04

@Joseph - this is happening with four files, ranging from 56 seconds to just over 4 minutes in length. Each is a single stream of audio using MPEG Audio Layer 1/2/3 (mpga), in stereo, 44,100hz sample rate, 128kb/s bit rate. They were generated with GarageBand 5.1. I tried re-saving the files using Audacity 1.3. No change. The files play fine in media players on my computer. – Ted Hopp Mar 22, 2011 at 6:11

Do the MP3 files play in the default Android media player or other Android media player apps. If so then it seems you've ruled out the encoding/bit-rate of the file. If not then try some of the MP3s that come with Android. – Joseph Earl Mar 22, 2011 at 10:40

@Joseph - The files play fine when I put them on the sd card and play them with the Music app. – Ted Hopp Mar 22, 2011 at 22:45

2 Answers

Sorted by: Highest score (default)

69

+100

AFAIK the buffers that MediaPlayer creates internally are for storing decompressed samples, not for storing prefetched compressed data. I suspect your stuttering comes from I/O slowness as it loads more MP3 data for decompression.

I recently had to solve a similar problem with video playback. Thanks to `MediaPlayer` being unable to play an arbitrary `InputStream` (the API is strangely lame) the solution I came up with was to write a small in-process webserver for serving up local files (on the SD card) over HTTP. `MediaPlayer` then loads it via a URI of the form http://127.0.0.1:8888/videofilename.

EDIT:

Below is the StreamProxy class I use to feed content into a MediaPlayer instance. The basic use is that you instantiate it, start() it, and set your media player going with something like `MediaPlayer.setDataSource("http://127.0.0.1:8888/localfilepath");`

I should note that it is rather experimental and probably not entirely bug-free. It was written to solve a similar problem to yours, namely that MediaPlayer cannot play a file that is also being downloaded. Streaming a file locally in this way works around that restriction (i.e. I have a thread downloading the file while the StreamProxy feeds it into mediaplayer).

```java
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.net.SocketTimeoutException;
import java.net.UnknownHostException;
import android.os.AsyncTask;
import android.os.Looper;
import android.util.Log;

public class StreamProxy implements Runnable {

    private static final int SERVER_PORT=8888;

    private Thread thread;
    private boolean isRunning;
    private ServerSocket socket;
    private int port;

    public StreamProxy() {
        // Create listening socket
        try {
          socket = new ServerSocket(SERVER_PORT, 0,
InetAddress.getByAddress(new byte[] {127,0,0,1}));
          socket.setSoTimeout(5000);
          port = socket.getLocalPort();
        } catch (UnknownHostException e) { // impossible
        } catch (IOException e) {
          Log.e(TAG, "IOException initializing server", e);
        }

    }

    public void start() {
        thread = new Thread(this);
        thread.start();
    }

    public void stop() {
        isRunning = false;
        thread.interrupt();
        try {
            thread.join(5000);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
    }

    @Override
      public void run() {
        Looper.prepare();
        isRunning = true;
        while (isRunning) {
          try {
            Socket client = socket.accept();
            if (client == null) {
              continue;
            }
            Log.d(TAG, "client connected");

            StreamToMediaPlayerTask task = new StreamToMediaPlayerTask(client);
            if (task.processRequest()) {
                task.execute();
            }

          } catch (SocketTimeoutException e) {
            // Do nothing
          } catch (IOException e) {
            Log.e(TAG, "Error connecting to client", e);
          }
        }
        Log.d(TAG, "Proxy interrupted. Shutting down.");
      }



    private class StreamToMediaPlayerTask extends AsyncTask<String, Void,
Integer> {

        String localPath;
        Socket client;
```

```java
        int cbSkip;

        public StreamToMediaPlayerTask(Socket client) {
            this.client = client;
        }

        public boolean processRequest() {
            // Read HTTP headers
            String headers = "";
            try {
              headers = Utils.readTextStreamAvailable(client.getInputStream());
            } catch (IOException e) {
              Log.e(TAG, "Error reading HTTP request header from stream:", e);
              return false;
            }

            // Get the important bits from the headers
            String[] headerLines = headers.split("\n");
            String urlLine = headerLines[0];
            if (!urlLine.startsWith("GET ")) {
                Log.e(TAG, "Only GET is supported");
                return false;
            }
            urlLine = urlLine.substring(4);
            int charPos = urlLine.indexOf(' ');
            if (charPos != -1) {
                urlLine = urlLine.substring(1, charPos);
            }
            localPath = urlLine;

            // See if there's a "Range:" header
            for (int i=0 ; i<headerLines.length ; i++) {
                String headerLine = headerLines[i];
                if (headerLine.startsWith("Range: bytes=")) {
                    headerLine = headerLine.substring(13);
                    charPos = headerLine.indexOf('-');
                    if (charPos>0) {
                        headerLine = headerLine.substring(0,charPos);
                    }
                    cbSkip = Integer.parseInt(headerLine);
                }
            }
            return true;
        }

        @Override
        protected Integer doInBackground(String... params) {

                        long fileSize = GET CONTENT LENGTH HERE;

            // Create HTTP header
            String headers = "HTTP/1.0 200 OK\r\n";
            headers += "Content-Type: " + MIME TYPE HERE + "\r\n";
            headers += "Content-Length: " + fileSize  + "\r\n";
            headers += "Connection: close\r\n";
            headers += "\r\n";

            // Begin with HTTP header
            int fc = 0;
            long cbToSend = fileSize - cbSkip;
            OutputStream output = null;
            byte[] buff = new byte[64 * 1024];
            try {
                output = new BufferedOutputStream(client.getOutputStream(),
32*1024);
                output.write(headers.getBytes());

                // Loop as long as there's stuff to send
                while (isRunning && cbToSend>0 && !client.isClosed()) {

                    // See if there's more to send
                    File file = new File(localPath);
                    fc++;
                    int cbSentThisBatch = 0;
                    if (file.exists()) {
                        FileInputStream input = new FileInputStream(file);
                        input.skip(cbSkip);
                        int cbToSendThisBatch = input.available();
                        while (cbToSendThisBatch > 0) {
                            int cbToRead = Math.min(cbToSendThisBatch,
buff.length);
                            int cbRead = input.read(buff, 0, cbToRead);
                            if (cbRead == -1) {
                                break;
                            }
                            cbToSendThisBatch -= cbRead;
                            cbToSend -= cbRead;
                            output.write(buff, 0, cbRead);
                            output.flush();
                            cbSkip += cbRead;
                            cbSentThisBatch += cbRead;
                        }
                        input.close();
                    }

                    // If we did nothing this batch, block for a second
                    if (cbSentThisBatch == 0) {
                        Log.d(TAG, "Blocking until more data appears");
                        Thread.sleep(1000);
                    }
                }
            }
            catch (SocketException socketException) {
                Log.e(TAG, "SocketException() thrown, proxy client has probably
closed. This can exit harmlessly");
            }
            catch (Exception e) {
                Log.e(TAG, "Exception thrown from streaming task:");
                Log.e(TAG, e.getClass().getName() + " : " +
e.getLocalizedMessage());
                e.printStackTrace();
            }

            // Cleanup
            try {
                if (output != null) {
                    output.close();
                }
                client.close();
            }
            catch (IOException e) {
                Log.e(TAG, "IOException while cleaning up streaming task:");
                Log.e(TAG, e.getClass().getName() + " : " +
e.getLocalizedMessage());
```

```
                e.printStackTrace();
            }

            return 1;
        }

    }
}
```

Share

Improve this answer

Follow

edited Mar 25, 2011 at 21:24

answered Mar 25, 2011 at 11:57

Reuben Scratton
38.7k ● 9 ● 80 ● 86

Interesting. Are you suggesting that I decode the mp3 files into something uncompressed (like PCM/WAVE) and feed that to MediaPlayer? – Ted Hopp Mar 25, 2011 at 16:42

1 Not at all, I'm suggesting that you preload some or all of the compressed data into memory. I think your problem is that when you simply give MediaPlayer a FileDescriptor, it doesn't buffer up enough data to feed into the MP3 decompressor... you suffer a buffer underrun, hence the stuttering. – Reuben Scratton Mar 25, 2011 at 17:06

1 Ah. That would explain why the trick of calling start() and then pause() right at the beginning would have an effect. I'm theorizing that it wasn't fully effective because the call to pause() came too soon, so the buffer didn't have a chance to fill up. I replaced the start/pause sequence with a call to seekTo(500) (and fiddled a bit with onSeekComplete) and the stutter now seems to have gone away! – Ted Hopp Mar 25, 2011 at 19:09

Glad you found a way out. If you later find out you need the advanced buffering solution (i.e. HTTP streaming the file into MediaPlayer within the process) then let me know and I'll send you the code. – Reuben Scratton Mar 25, 2011 at 19:53

2 Can't do that because I don't use this hacky technique any more. These days I avoid MediaPlayer completely because it is *stuffed* with bugs. I replaced it with a port of FFmpeg with a powerful custom Java wrapper that, amongst other things, allows you to play arbitrary InputStreams. – Reuben Scratton Apr 4, 2013 at 20:44

3

Would using `prepareAsync` and responding to `setOnPreparedListener` suit you better? Depending on your activity workflow, when the `MediaPlayer` is first initialized you could set the preparation listener and then call `mPlayer.prepareAsync()` later once you're actually loading the resource, then start playback there. I use something similar, albeit for a network-based streaming resource:

```
MediaPlayer m_player;
private ProgressDialog m_progressDialog = null;

...

try {
    if (m_player != null) {
    m_player.reset();
    } else {
        m_player = new MediaPlayer();
    }

    m_progressDialog = ProgressDialog
        .show(this,
            getString(R.string.progress_dialog_please_wait),
            getString(R.string.progress_dialog_buffering),
            true);

    m_player.setOnPreparedListener(this);
    m_player.setAudioStreamType(AudioManager.STREAM_MUSIC);
    m_player.setDataSource(someSource);
    m_player.prepareAsync();
} catch (Exception ex) {
}

...

public void onPrepared(MediaPlayer mp) {
    if (m_progressDialog != null && m_progressDialog.isShowing()) {
      m_progressDialog.dismiss();
    }

    m_player.start();
}
```

There's obviously more to a complete solution (error-handling, etc.) but I think this should work as a good example to start from that you can pull the streaming out of.

Share Improve this answer Follow

answered Mar 20, 2011 at 16:14

Matt Bishop
1,007 ● 7 ● 19

I'll try this and report the effects. I don't hold much hope for it, because the problem is not in the time to prepare; it seems to be the processing that's done after `start()` is called. But then, I'll be the first to admit that I don't fully understand how MediaPlayer works internally. – Ted Hopp Mar 20, 2011 at 19:04

3 I've now tried this variation and it had no effect on the problem. – Ted Hopp Mar 21, 2011 at 1:34

Could you share some details on the file being played? What's the bitrate? What encoder was used to build the MP3 container? I realize it's a workaround but perhaps you're simply expecting too much and the device has trouble pushing the data out, thus a lower bitrate might solve your issue. Also, is this in the emulator? Does the issue occur on different devices? – Matt Bishop Mar 22, 2011 at 0:04

@Matt - Four files are involved. They were created by GarageBand 5.1 with a sample rate of 44,100hz. Three are at a bit rate of 128kbs and are roughly a minute long each. The fourth, at about 4 minutes, is at 86kbs. The files play smoothly after the initial stutter (actually, more like a restart), so I wonder how bit rate could be the issue. – Ted Hopp Mar 22, 2011 at 6:04

I've attempted to find the source for Android's MP3 playback and have been unsuccessful. Some encoders don't always create the container in the most efficient way and perhaps the

MP3 playback module for Android gets hung up for a moment. Perhaps you can try exporting the sound files as pure PCM and then using another encoder like LAME to see if that makes any playback differences. You could also pad the audio tracks with half a second of silence so that the stutter isn't heard. I realize these are workarounds but MediaPlayer seems unchangeable at this point; the buffer can't be adjusted. – Matt Bishop Mar 22, 2011 at 13:34