# Developing Streaming Media Applications Using the MMAPI: What Works, What Doesn't and What to Do About It


**Craig Robinson**

Director of Technology
RealNetworks
http://www.realnetworks.com/

**Brent Newman**

Lead SDE
RealNetworks


TS-5439

# Using the MMAPI

## Practical advice for developing streaming applications

You can develop streaming applications for mobile handsets using Java™ Platform, Micro Edition (Java ME) technology! In this session we will show you how, as well as point out some issues you need to watch out for. Our presentation concludes with code examples and a streaming demo.

# Agenda

What Can You Build?

Overview of Streaming Technologies

Practical Issues with Testing

Limitations of MMAPI Implementations

Helix JSR 135 Bindings

Code Examples

Demo

Q&A

# Agenda

**What Can You Build?**

Overview of Streaming Technologies

Practical Issues with Testing

Limitations of MMAPI Implementations

Helix JSR 135 Bindings

Code Examples

Demo

Q&A

# What Can You Build?

## Mobile media applications are hot right now!

- Mobile TV and VOD
  - Live television
  - On-demand shows
    - News, sports, movies
- Mobile music
  - Radio
  - On-demand tracks
- The ultimate mobile music experience
  - Jukebox in the sky
  - Any song, any time, any place

# Agenda

What Can You Build?

**Overview of Streaming Technologies**

Practical Issues with Testing

Limitations of MMAPI Implementations

Helix JSR 135 Bindings

Code Examples

Demo

Q&A

# Overview of Streaming Technologies

## Preliminaries: What is streaming?

- Just-in-time delivery of time-based media
  - Media is delivered from server to client
  - Media segments are delivered shortly before they must be presented
- In practice, some buffering exists
  - Pre-roll buffer
  - Network jitter buffering
  - Handover buffering (mobile)
  - 3GPP PSS defines a standard buffer model

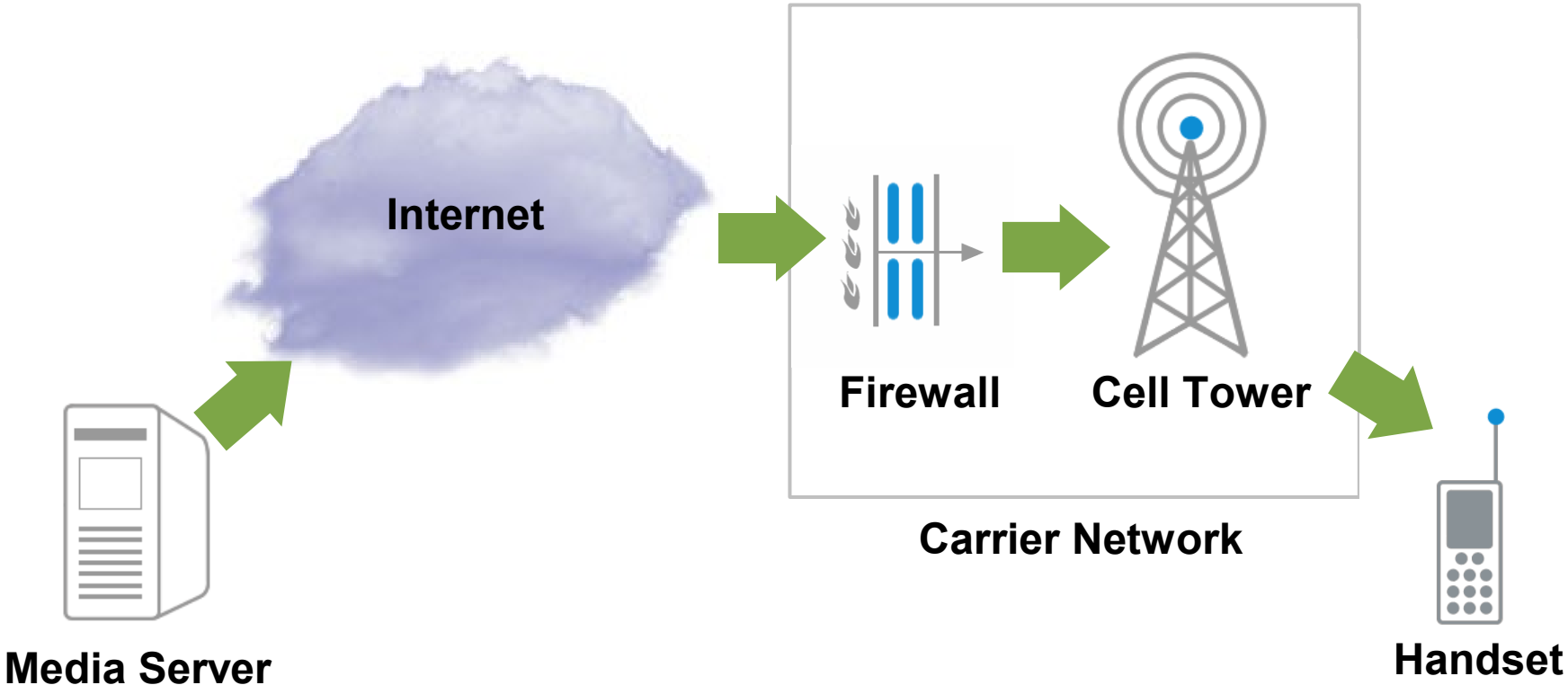
# Overview of Streaming Technologies

## 3GPP PSS

- 3rd Generation Partnership Project
  - Industry consortium that sets international mobile standards
- 3GPP packet switch streaming standards
  - “The” standard for mobile streaming
  - Consolidates and refines streaming and media standards defined by IETF, MPEG, W3C and others
- Relevant specifications include
  - TS 26.234: “Transparent end-to-end Packet-switched Streaming Service (PSS);Protocols and codecs”
  - TS 22.233: “Transparent End-to-End Packet-switched Streaming Service; Stage 1”
  - TS 26.233: “Transparent end-to-end packet switched streaming service (PSS); General description”

# Overview of Streaming Technologies

## Diagram: streaming to a mobile device



# Overview of Streaming Technologies

## Protocols used for mobile streaming with Java ME platform

- RTSP—Real Time Streaming Protocol
  - RTSP is a control protocol
    - Play, Pause, Stop
    - It is always used with a data transport protocol
  - RTP—Real-time Transport Protocol
    - Can be delivered via UDP or TCP
    - 3GPP specifies UDP transport for RTP

# Overview of Streaming Technologies

## Protocols used for mobile streaming with Java ME platform

- HTTP
  - Protocol underlying the Web, may also be used for streaming
  - TCP transport
  - Control and data on the same connection
  - Non-standard for streaming
  - Most implementations of MMAPI have problems with HTTP streaming
    - Entire clip is buffered prior to playback

# Overview of Streaming Technologies

## Audio Codecs

- RealAudio
  - Good quality at 48-64Kbps
  - Much better than mp3 at same bitrates
- AMR
  - Widely implemented voice codec
  - Standardized by 3GPP
- AAC/aacPlus
  - Great quality across range of mobile bitrates
  - Standardized by 3GPP

# Overview of Streaming Technologies

## Video Codecs

- RealVideo
  - Proprietary codec from Real
  - Low cost licensing
  - Great quality across a wide range of bitrates
- H.263
  - Widely implemented, but lower quality
  - Standardized in 3GPP PSS

# Overview of Streaming Technologies

## Video Codecs

- MPEG-4
  - Similar to H.263
  - Standardized in 3GPP PSS
- H.264 (MPEG-4 AVC)
  - High quality codec
  - Starting to see implementations accessible from Java ME platform
  - Standardized in 3GPP PSS

# Overview of Streaming Technologies

## Recommended encode bitrates for mobile streaming

| | Audio | Audio/Video |
|---|---|---|
| GPRS | 20kbps | 28kbps |
| CDMA 1x | 24kbps | 40kbps |
| EDGE | 32kbps | 54kbps |
| EV-DO | 48-64kbps | 100-128kbps |
| UMTS | 48-64kbps | 100-128kbps |

# Agenda

What Can You Build?

Overview of Streaming Technologies

**Practical Issues with Testing**

Limitations of MMAPI Implementations

Helix JSR 135 Bindings

Code Examples

Demo

Q&A

# Practical Issues with Testing

## How to test mobile streaming applications

- RTSP
  - Server must be accessible to the handset (Internet)
  - Problems with carrier firewalls
    - Carrier must allow RTSP traffic (port 554)
    - Carrier must allow streaming of UDP traffic
- Debugging
  - Can sometimes sniff traffic on server side
  - Client side debugging is tricky
    - Standard mobile debugging techniques
- Need to have flat rate data plan
  - Streaming charges can really add up!

# Agenda

What Can You Build?

Overview of Streaming Technologies

Practical Issues with Testing

**Limitations of MMAPI Implementations**

Helix JSR 135 Bindings

Code Examples

Demo

Q&A

# Limitations of MMAPI Implementations

## Things to be aware of

- Codec support differs from handset to handset
- Implementations may incorrectly report what is supported
- Behaviour differences
  - Volume often not mapped to system volume
  - Player events not always reported
- HTTP streaming not widely supported
- No access to the data stream
  - Precludes things like decryption at Java technology layer

# Agenda

What Can You Build?

Overview of Streaming Technologies

Practical Issues with Testing

Limitations of MMAPI Implementations

**Helix JSR 135 Bindings**

Code Examples

Demo

Q&A

# Helix JSR 135 Bindings

## A Battle-tested implementation for JSR 135

- Open Source project on the HelixCommunity to adapt the Helix media engine to support JSR 135 interface
- Same media engine underlying the RealPlayer
- Supports 3GPP PSS and Real codecs and protocols
- Lightweight framework—optimized for mobile
- Can be used by JSR 135 implementers and handset manufacturers

# Agenda

What Can You Build?

Overview of Streaming Technologies

Practical Issues with Testing

Limitations of MMAPI Implementations

Helix JSR 135 Bindings

**Code Examples**

Demo

Q&A

# Code Examples

## What content MIME types does my handset support?

- To get all supported content types:

```
String[] contentTypes =
  Manager.getSupportedContentTypes(null);
```

```
Some Possible Results:
```

```
audio/w-wav                audio/amr
audio/midi                 video/mp4
audio/mid                  video/3gpp
audio/x-tone-seq           video/3gpp2
audio/mpeg                 video/x-pmd
audio/mp4                  audio/qcelp
audio/mp4a-latm            audio/vnd.qcelp
audio/3gpp                 audio/3gpp2
```

- Provide parameter to filter by protocol:

```
String[] contentTypes =
  Manager.getSupportedContentTypes("rtsp");
```

# Code Examples

## What media protocols does my handset support?

- To get all supported protocols

```
String[] protocols =
  Manager.getSupportedProtocols(null);
```

```
Some Possible Results:

  device
  http
  https
  rtsp
  file
  capture
```

- Provide parameter to filter by content type:

```
String[] protocols =
  Manager.getSupportedProtocols("audio/mp4a-latm");
```

# Code Examples

## What content types are streamable on my handset?

- To get all supported streamable contents:

```
String streamable =
  System.getProperty("streamable.contents");
```

```
Some Possible Results:
  audio/x-wav (emulator)
  audio/mp4a-latm (handset)
```

- Warning: This method is fairly unreliable; some devices return values here, others return null.

# Code Examples

## RTSP streaming made easy

- Start the playback!

```
String mUrl = "rtsp://abc.org:554/content/test.3gp";
Player mPlayer = Manager.createPlayer(mUrl);

mPlayer.start(); // Blocks until player is realized
```

- Use another thread to avoid blocking the UI
- Implement PlayerListener interface
  - Allows you to get information about playback status
  - Can present feedback to the user

# Code Examples

## Sometimes you need to be creative!

- Many early implementations did not support streaming
- HTTP streaming often downloads entire file first
- Millions of these handsets are still in use!
- Creative solution: “chunked” playback
  - Use HTTP to read media
  - Break into individual, playable pieces
    - Play back-to-back
- Disadvantages
  - Can result in audible clicks or gaps between clips
  - Requires intimate knowledge of file format

# Code Examples

## ChunkPlayer overview

- Custom player implementation
- Receives events from MMAPI for each chunk
- Hides “chunking” behavior from rest of app
- Includes standard play(), stop(), etc.
- Manages time, download progress
- Must carefully manage memory, especially on older, limited handsets

# Code Examples

## playerUpdate()—manages next Chunk to be played

```
if (event == PlayerListener.END_OF_MEDIA) {
    mCurrentChunk = mChunker.getNextChunk();
    mCurrentChunk.addListener(this);

    if (mCurrentChunk != null) {
        try {
            mCurrentChunk.start();
            mLastClipStart = mMediaTime;
        }
        catch (Exception e) {
            // handle exception
        }
    }
    else { stop(); }
}
```

# Code Examples

## Chunker overview

- Reads file via HTTP
- Makes a copy of the original file header
- Breaks the audio data into playable “chunks”
- Copies new file header into chunk, adjusts for new size and duration
- Reports progress to ChunkPlayer
- Watches memory, reports problems
- Can be customized for any file format

# Code Examples

## Chunker initialization

```
public void open(String url, String contentType) {
  try {
    mHttpConnection =
     (HttpConnection)Connector.open(url);
    mInputStream = new
      DataInputStream(mHttpConnection.openInputStream());
    mTotalLength = (int)mHttpConnection.getLength();
    mDuration = 0;
    mContentType = contentType;
    mHeader = new byte[HEADER_SIZE];

    new Thread(this).start();
  }
```

# Code Examples

## Chunker—main run() method

```
if ((readFully(mHeader, HEADER_SIZE, 0) == true) &&
    (isValidHeader(mHeader) == true) &&
    (decodeFormat(mHeader) == true)) {
    int newChunkSize = getNextChunkSize();
    while (!mDone) {
        if (newChunkSize != 0) {
            processNextChunk(newChunkSize);
        }
        else {
            try {
                Thread.sleep(1000);
            } catch (Exception ie) { mDone = true; }
        }
        newChunkSize = getNextChunkSize();
```

# Code Examples

## Creating a Chunk

```
public void processNextChunk(int size) {
    Chunk chunk = new Chunk(mType);
    chunk.allocateBuffer(HEADER_SIZE + size);
    // Copy header information into buffer
    System.arraycopy(mHeader, 0, chunk.getData(),
                     0, HEADER_SIZE);
    // Set the total size
    setUINT32(chunk.getData(),
              size + HEADER_SIZE - 8, 4);
    chunk.setDuration((8 * (long)size * 1000000) /
                      ((long)mAverageBPS));
```

# DEMO

Streaming with the Java ME Platform

# Summary

- Mobile media applications are hot!
- You can build streaming apps with the Java ME platform
- Different handsets support different codecs
- Testing mobile streaming apps can be tricky
- Some MMAPI implementations are better than others
- Helix JSR 135 bindings provide a solution for VM vendors and handset manufacturers
- Some problems with current implementations can be overcome with creative strategies

# For More Information

- JSR 135
  - http://www.jcp.org/en/jsr/detail?id=135
- 3GPP
  - http://www.3gpp.org/
- Helix JSR 135 bindings at HelixCommunity
  - http://helix-client.helixcommunity.org/2005/devdocs/jsr135adaptation
- Java ME Platform Media Services Profile
  - Document created by Real as a guide for handset manufacturers and JSR 135 implementers (see us for details)

# Q&A

Craig Robinson, RealNetworks

Brent Newman, RealNetworks

# Developing Streaming Media Applications Using the MMAPI: What Works, What Doesn't and What to Do About It


**Craig Robinson**

Director of Technology
RealNetworks
http://www.realnetworks.com/

**Brent Newman**

Lead SDE
RealNetworks


TS-5439