



Get Your Content Onto Google TV

Christian Kurzke
Developer Advocate

Andrew Jeon
Platform Manager

Mark Lindner
Tech Lead



New Google TV Devices

- New Google TV devices with **ARM** processors
- Available in USA and **Internationally**



Sony



Vizio



LG



“Content” Is King

Streaming

- High Quality
- Securely

Integrated

- One Interface



“Content” Is King

Streaming

- High Quality
- Securely

Integrated

- One Interface



HTTP “Streaming”

Android

```
MediaPlayer mPlayer = new MediaPlayer();  
mPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);  
mPlayer.setDataSource(“http://music.foo.com/ForElise.mp3”);
```

Simultaneous download & playback == Progressive Download

“OK” for music. Today’s bandwidth is enough to download “perfect” audio quality.



Utilizing Network Bandwidth

Bit rate < Bandwidth

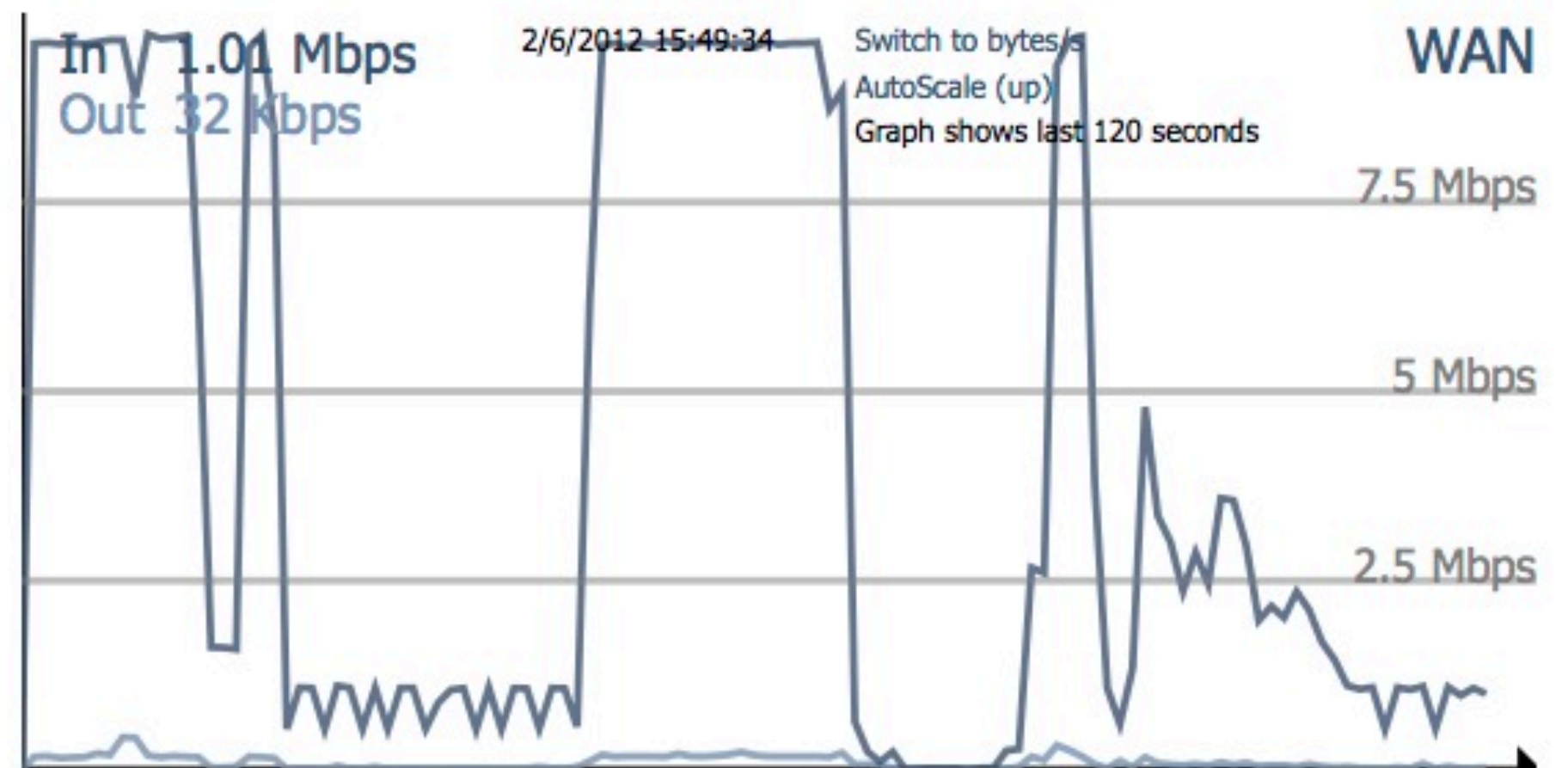
- **Not achieving best quality possible**

Bit rate > Bandwidth

- **Re-buffering**

Status: Traffic graph

Interface: WAN



Variable Bit Rate Encoding

VBR Benefits:

- Better **perceived** video quality than constant bit rate
- **Best use of bandwidth**

But:

- Variable bit rate **is not adaptive** to network capacity

Good choice for mobile, where primary goal is to **minimize overall bandwidth**.
Not ideal for TV where goal is to **maximize quality**.



Adaptive Bit Rate Streaming

Goal: **Bit rate** \approx **Bandwidth**

Typically consists of:

- A **set** of VBR video content files (synchronized to switch seamlessly)
- Descriptor file (XML) with meta information
- Metadata also typically has “seek” information (jump forward, backward in stream)

- The MediaPlayer **URL** points to the **descriptor** file

Various Standards: HLS, MPEG Dash, Smooth Streaming etc.



Android Adaptive Bitrate Streaming

Android

```
MediaPlayer mPlayer = new MediaPlayer();  
mPlayer.setDataSource("http://video.foo.com/MyStream.m3u8");
```

Android (mobile) supports:

- HTTP Live Streaming (HLS) and RTSP
- Widevine

Google TV has all this -

And now also....



Announcing Support for Smooth Streaming Protocol

To make it easier for content owners to deliver streaming video content to Google TV:

- We have **now added** the capability to play back video using the Microsoft **Smooth Streaming** protocol
 - Developers can simply use URL points to ***.ism URL** with **Android MediaPlayer API** on **Google TV**

```
MediaPlayer mPlayer = new MediaPlayer();  
mPlayer.setDataSource("http://video.foo.com/test_video.ism");
```

- We are actively working on supporting **MPEG-DASH** streaming protocol in the future





**But what if you have a “custom”
Streaming Server?**

Supporting Custom Streaming Protocols

Streaming protocols are evolving, and many content owners are using proprietary protocols:

- Unique needs
- Legacy versions
- Future “Standard” Protocol versions

The problem:

Android **MediaPlayer.setDataSource()** only allows URI objects, Strings or FileDescriptors.
Does **not** allow generic **InputStream** object.



Introducing The Google TV Media Source API

We created a set of APIs which:

- Enable Android Developers to create own
 - **Streaming Protocol** implementation
 - **Media Container** parsers
- Decode and play back content using platform **hardware accelerated** codecs.

```
GtvMediaPlayer.setMediaSource(myMediaSource)
```

MyMediaSource can **extend** either **PullMediaSource** or **PushMediaSource** class



Implementing Your Custom Media Source

Android

```
public abstract class MediaSource {  
  
    protected int getNumberOfTracks() {  
        // Returns the number of tracks in the variable, mMediaInfo.  
    }  
  
    protected abstract void onStart() {  
        // Called when the NativePlayer is in the "preparing" state  
    }  
  
    protected abstract void onEnableTrack(int trackId, long startTimeUs) {  
        // Called when a track is enabled  
    }  
  
    protected abstract void onDisableTrack(int trackId) {  
        // Called when track is disabled. After this method is called  
    }  
  
    protected abstract void onHandleDiscontinuity (int trackId, int discontinuityType) {  
        // Called when a media track is skipping (e.g. format change, time jump)  
    }  
  
    protected abstract void onSeekTo(long timeUs) {  
        // Called when a media playback is seeked (ffwd, rwd).  
    }  
}
```



Implementing A Custom Pull Media Source

Implement all the methods in **PullMediaSource** and

- The following Methods

```
public class MyCustomPullMediaSource extends PullMediaSource {  
    ... implement all the "Media Source" methods  
  
    /* Called by NativePlayer request next unit of media stream. */  
    protected abstract AccessUnit onDequeueAccessUnit(int trackId) {  
  
        // in a "real world" implementation, re-use byte[] and MediaStreamChunk object!!  
  
        byte[] myData = ...          // read, parse and extract from custom data stream  
  
        AccessUnit myAccessUnit =  
            AccessUnit.createAccessUnit (myData, myDataLength, myMediaContext);  
  
        return (myAccessUnit);  
    }  
}
```

Android

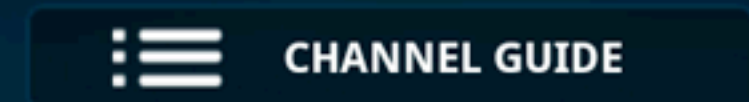
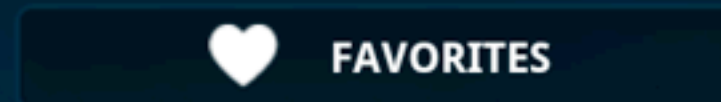




The Wutchuwant Countdown

fun./Janelle Monae

We Are Young



04:58:34



Gary Conners, Director, Advanced Product Technology, SiriusXM:

“With this capability, we have been able to quickly develop an app for Google TV that plays our proprietary audio streaming format [...] without requiring OEM-specific code.”

Additional GtvMediaPlayer Features

We have added more AV features to improve media playback experience on Google TV


Support multiple audio tracks

- Separate language tracks

Closed Captions and Subtitle support

- Support for standard **Timed Text Markup Language** (TTML)
- Provide a widget which developers can freely modify and use in applications to easily display **Closed Captions** and **Subtitles** based on TTML





You're **not** paying attention!
(*Ringing sound*)

“Content” Is King

✓ Streaming

- High Quality
- Securely

Integrated

- One Interface



Ensuring high quality delivery

For best results, you want to continuously **monitor**:

- Network bandwidth
- Playback **quality** (Frames per second, FPS and dropped frames)

This allows you to:

- Ensure **customer satisfaction**
 - Offer refunds or re-play
- Proactively detect (and mitigate) **connectivity problems**
 - “Sorry, your Internet connection does not support 1080p playback!”



We Introduce Quality of Service (QoS) APIs

Now you can measure:

- Frames per Second (**FPS**)
- Network **Bandwidth**
- Buffer size, **Buffer fill rate**
- Buffered media playback duration
- Audio Info
- Detailed errors from underlying system, Unexpected End Of Stream (EOS)

Use “**Analytics**” frameworks to aggregate Playback Quality statistics



Example using QoS API

- Implement a **MediaPlayer.OnInfoListener** class
- Register with **GtvVideoView.setOnInfoListener()**.

Android

```
private class InfoListener implements MediaPlayer.OnInfoListener {

    public boolean onInfo(MediaPlayer mediaPlayer, int what, int extra) {

        switch (what) {
            case GtvMediaPlayer.MEDIA_INFO_NETWORK_BANDWIDTH:
                Log.d(TAG, "Current Bandwidth: " + extra + " kbps"); break;

            case GtvMediaPlayer.MEDIA_INFO_FPS: {
                Log.d(TAG, "Current FPS: " + extra);
                // In real world, check instanceof()
                MediaPlayer gtvPlayer = (GtvMediaPlayer) mediaPlayer;
                // get current media playback info
                OnInfoMetadata mediaInfo= gtvPlayer.getOnInfoMetadata();
                int droppedFPS = mediaInfo.getInteger(OnInfoMetadata.MEDIA_INFO_META_DROPPED_FPS);

                Log.d(TAG, "Dropped FPS: " + droppedFPS);
            }
        }
    }
}
```

Developers first!

We will provide:

- An implementation of **Smooth Streaming protocol** with
 - Media Source API
 - PlayReady DRM handling in Java
- Multi-track audio handling
- TTML based closed caption handling
- Demonstration of QoS APIs to monitor streaming quality

We will release **sample code** how to use all those APIs as **Open Source**.



But Wait... There is More!

Google TV apps can now play back **High Quality Content** directly from **YouTube**

If you want to utilize **OUR** existing **encoding** and **streaming servers** inside **YOUR** app:

Use the **YouTube Android Player API** for:

- Playing videos and playlists
- Registering to be notified of playback events
- YouTube UI widgets (e.g., YouTubePlayerView, VideoThumbnailView)

Session: **YouTube Player API**, Thursday at 1:30 PM



“Content” Is King

- ✓ Streaming
 - ✓ • High Quality
 - Securely

Integrated

- One Interface



DRM - In a Nutshell

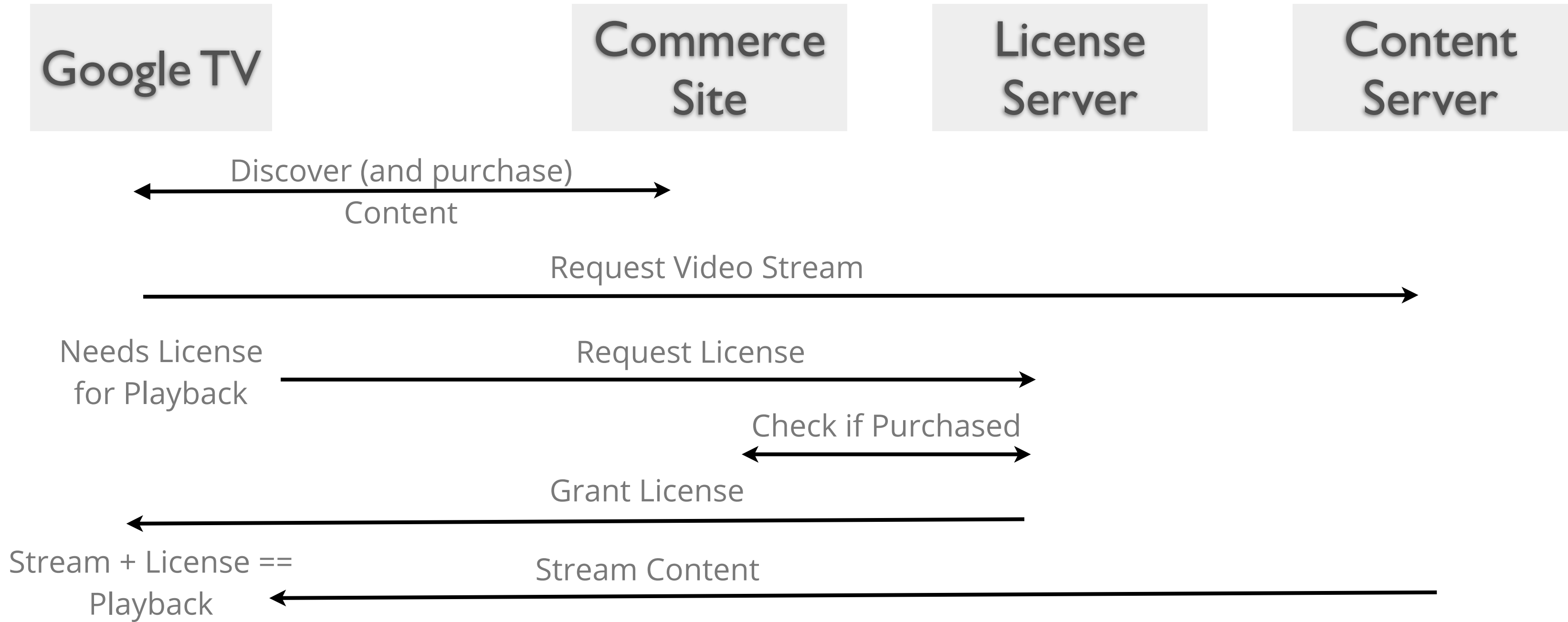
Digital Rights Management:

- License Management
- Transport Encryption
- Secure Decoding (Trusted Video Path)

HTTPS is not a DRM



Typical DRM Use Case



DRM - In Android

DRM Framework introduced in Android 3.0 (HoneyComb)

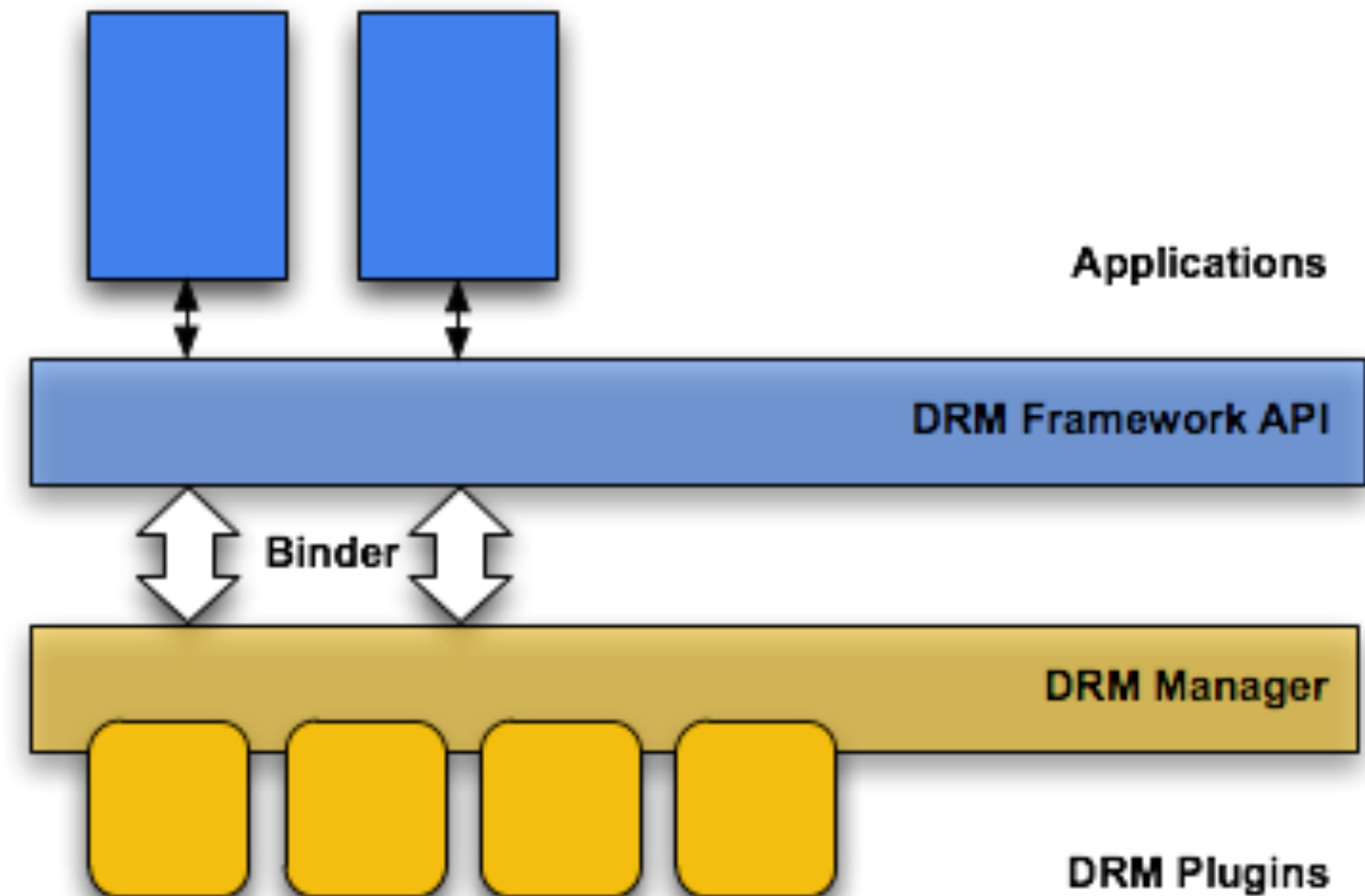
- Extensible
- Android Application interact with a “Native Code” **DRM Provider** implementation

Android DRM Framework in package: [android.drm](#)

Android has built-in support for Widevine DRM.

But: Custom DRM Plugins

- Require Native Code
- Difficult to develop



Introducing PlayReady DRM for Google TV

ARM based Google TV devices will include support for **Microsoft PlayReady DRM**:

- Implemented as a **plug-in** to Android DRM Framework
- Accessed via standard **Android DRM** (Java) **APIs**
- Supports basic **license acquisition** and **management**
 - **Extensible** to adapt to custom license servers and protocols.
- Playback using hardware "**Trusted Video Path**" (TVP)
- Integrated with Smooth Streaming protocol



Google TV - Keeping Your Content Safe

Google TV DRM Components:

- Android **DRM Framework**
 - Managing playback rights of content
 - **Widevine** and **PlayReady** support now built into the platform
- **Trusted Video Path** (TVP)
 - Keeps decrypted video data securely in hardware secure “sandbox”
 - Protects streamed media securely
- **HDMI Content Protection**
 - Protecting Video Content all the way to the Television display



“Content” Is King

- ✓ Streaming
 - ✓ • High Quality
 - ✓ • Securely

Integrated

- One Interface



The "User Interface" To Your Entertainment



How Viewers Discover Content

Users **Search** for content in System Search

Users **Browse** for content in the TV & Movies app



How Google TV Finds Content

Content available on Google TV is an aggregate of:

- Backend:
 - Search engine indexed data: Video Site Map XML files
- Client side:
 - Media Devices (physical)
 - Media Devices (**virtual**)



Media Device Interactions





office



The **Office** (2005-Present)

TV Series | 170 episodes available



Office Space (1999)

Movie | R | 1 hr 29 min | Available from 3 sources



See all TV, Movie, and Video results

for '**office**'



See web search results from Google.com

for '**office**'



The **Office** (2002)

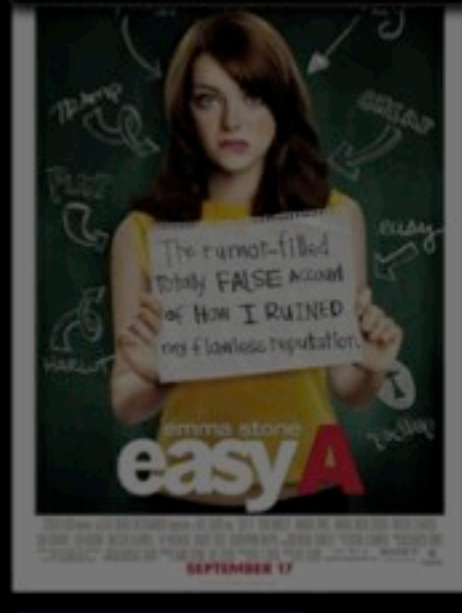
TV Series | 12 episodes available

Favorite Channels

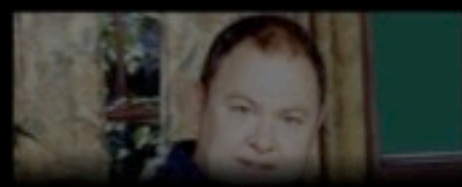


Movies on TV

Countdown to the Closing Bell
 Aired May 11, 2008. Stock market updates.
 Press 1-5 to rate 19 mins left, 12:00 PM-1:00 PM FBN
 Press menu for options



Comedy






DVR Recorded

Episodes

Details

 browse

Showing all 88 episodes from Fringe

Free and available now

Episode 21

May 4, 2012

Brave New World, Part 1 of 2 **HD**

Walter is forced to revisit his painful past when a fringe event causes people to spontaneously combust; the team faces off with David Robert Jones.

BUY

Episode 20

Apr 27, 2012

Worlds Apart **HD**

Both teams fight for the same cause; shocking developments related to the Cortexiphan children arise.

BUY

Episode 19

Apr 20, 2012

Letters of Transit **HD**

The Observers and the team engage in a battle in the year 2036.

FREE

Episode 18

Apr 13, 2012

The Consultant **HD**

Walter goes to the alternate universe to help investigate an event that has ties to both worlds.

BUY



Your "device"
HERE



Showing all 88 episodes from Fringe

Free and available now

Episode 21 Brave New World, Part 1 of 2 HD

Letters of Transit



Watch instantly

FREE



View details

BUY

HD



View details

BUY



Rent now

\$2.99

Episode 18
Apr 13, 2012

The Consultant HD

Walter goes to the alternate universe to help investigate an event that has ties to both worlds.

BUY

browse

One Way Pairing

A (legacy) device that is controlled via an **Infra-Red** (IR) Blaster.

- When tighter integration is not available

Device cannot send information to GoogleTV.



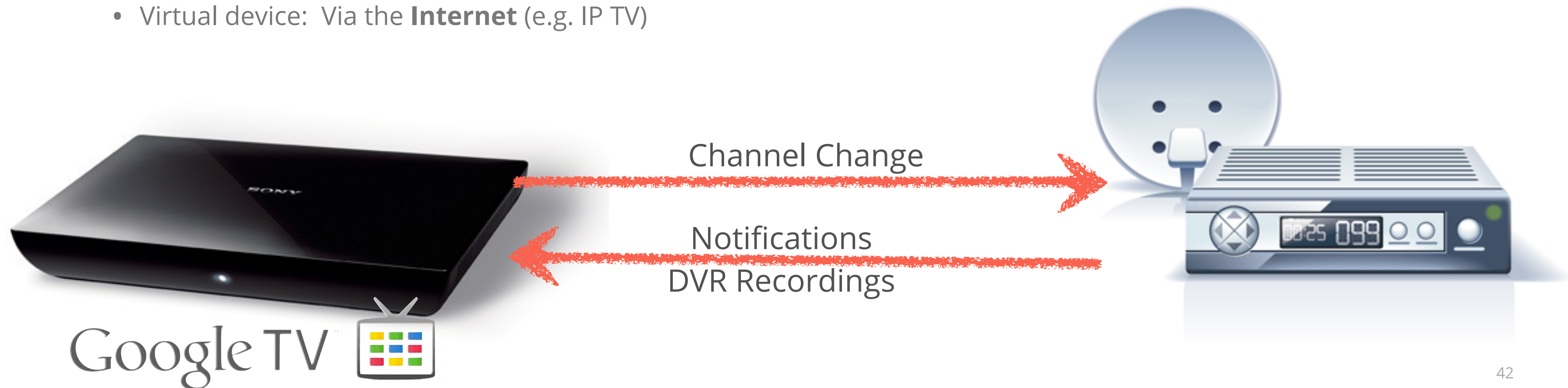
Channel Change
e.g: tune to:
"tv://channel/CNN")
or perform action
"Menu", "Channel Up"



Two Way Pairing

When a published communication protocol exists for controlling the device.

- Bidirectional communication
- Device can **send information** and **events** to GoogleTV
 - Channel change events
 - Closed caption text
- Frequently uses **TCP/IP**
 - Physical device: **Locally** connected
 - Virtual device: Via the **Internet** (e.g. IP TV)



“Content” Is King

- ✓ Streaming
 - ✓ • High Quality
 - ✓ • Securely

Integrated ✓

- One Interface



Integrating New Media Devices



Google TV 



???



Building Your Own Media Device

A Media Device is:

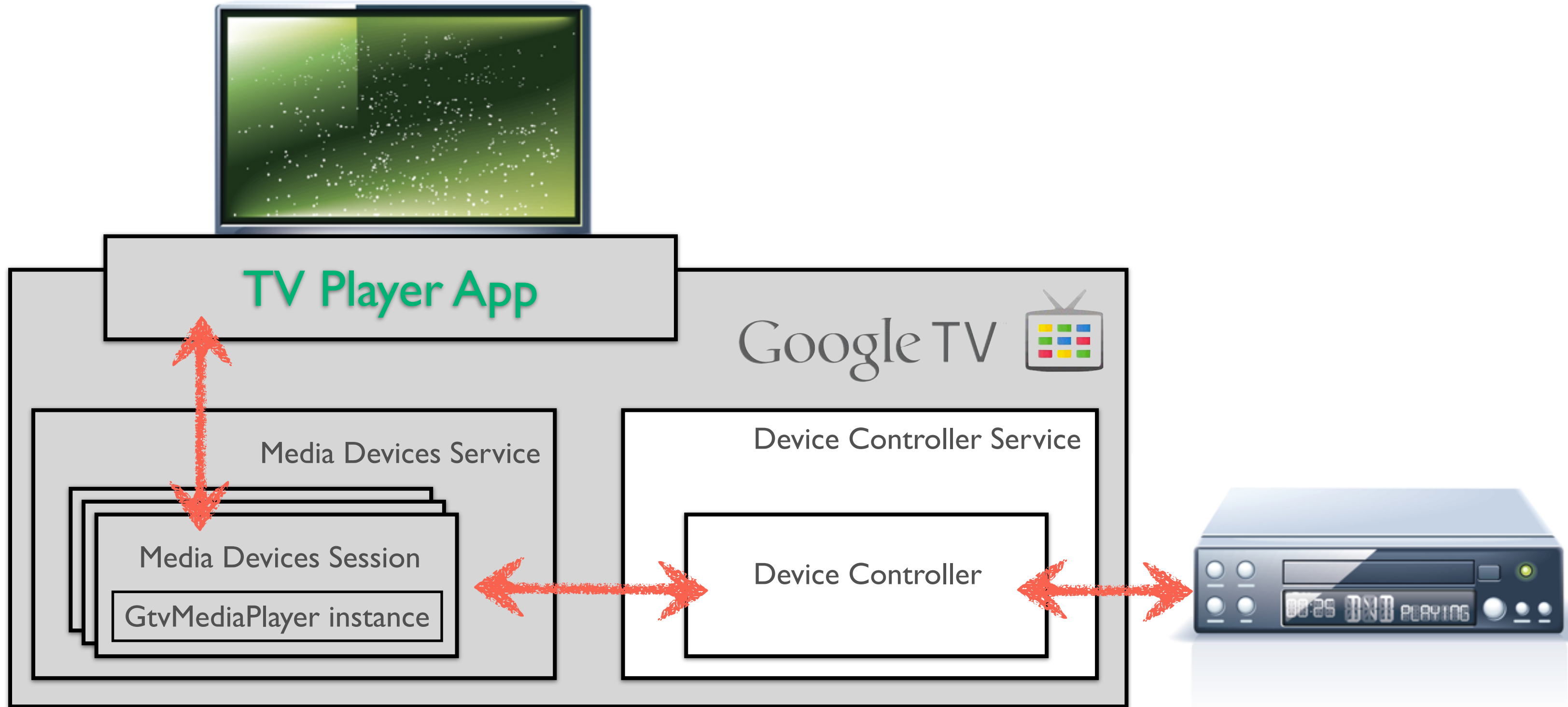
- Similar to a “Device **Driver**”
- **Packaged** as an **.apk**
- **Installed** from the **Google Play** store

Media Devices consist of the following software components:

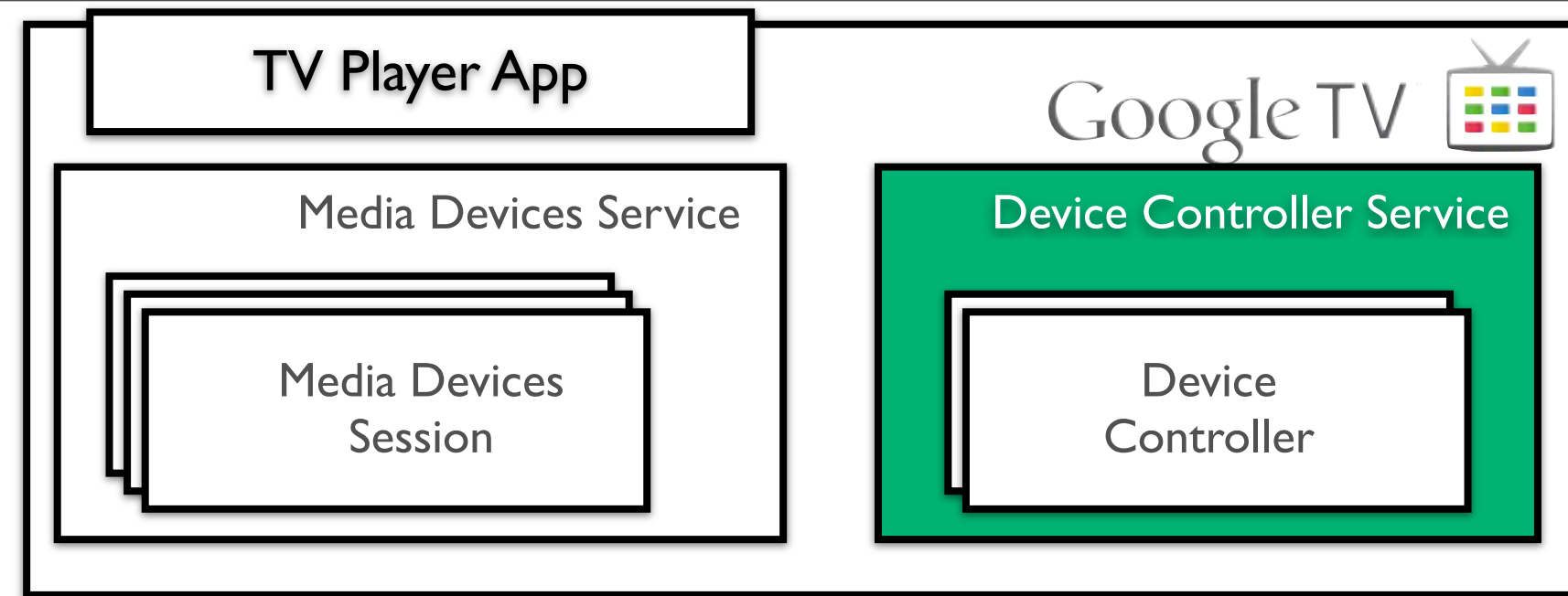
- **Media Device Controller Service** (Android Service)
- **Setup Activity**
- **Settings Activity**



Media Devices Framework Overview



Device Controller Service



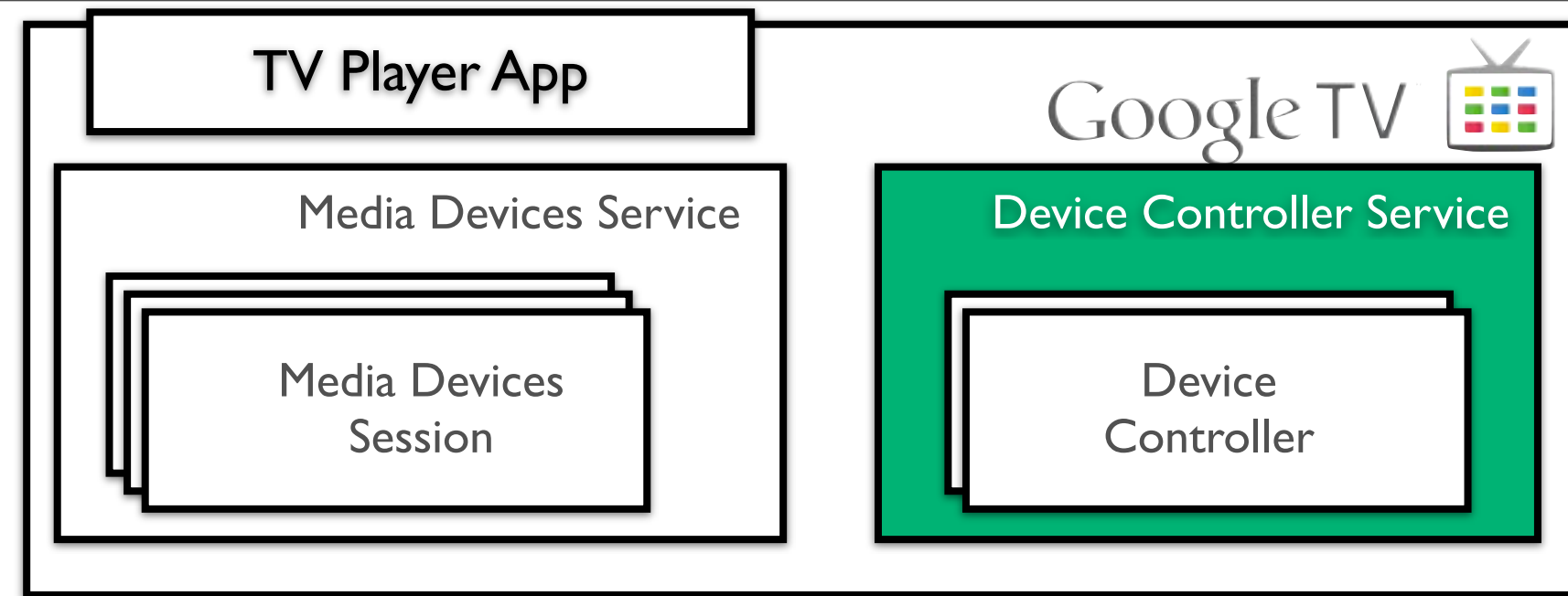
- Implemented as **Android service** (runs in the background)
- Interfaces to one or more media devices
- **Notifies the system:**
 - When devices go online / offline
 - Events like channel changes etc.
- **Reports** the devices **information:**
 - Channel lineups
 - Channel numbers / Call Signs
 - Names, logo icons



Implementing a Device Controller Service

(Step 1/2)

Subclass **AbstractDeviceControllerService**



- Register the device(s) and their channel lineup(s) with the system:

```
public final class MyDeviceService extends AbstractDeviceControllerService {
    public void onCreate() {
        super.onCreate();
        Device myDevice = buildDevice();
        setChannelUpdateInterval(myDevice.getId(), CHANNEL_UPDATE_INTERVAL_MS);
        addDevice(myDevice);
    }

    protected final AbstractDeviceController buildController(final Device device) {
        return new MyDeviceController(this, getSettings(), device.getId());
    }
}
```

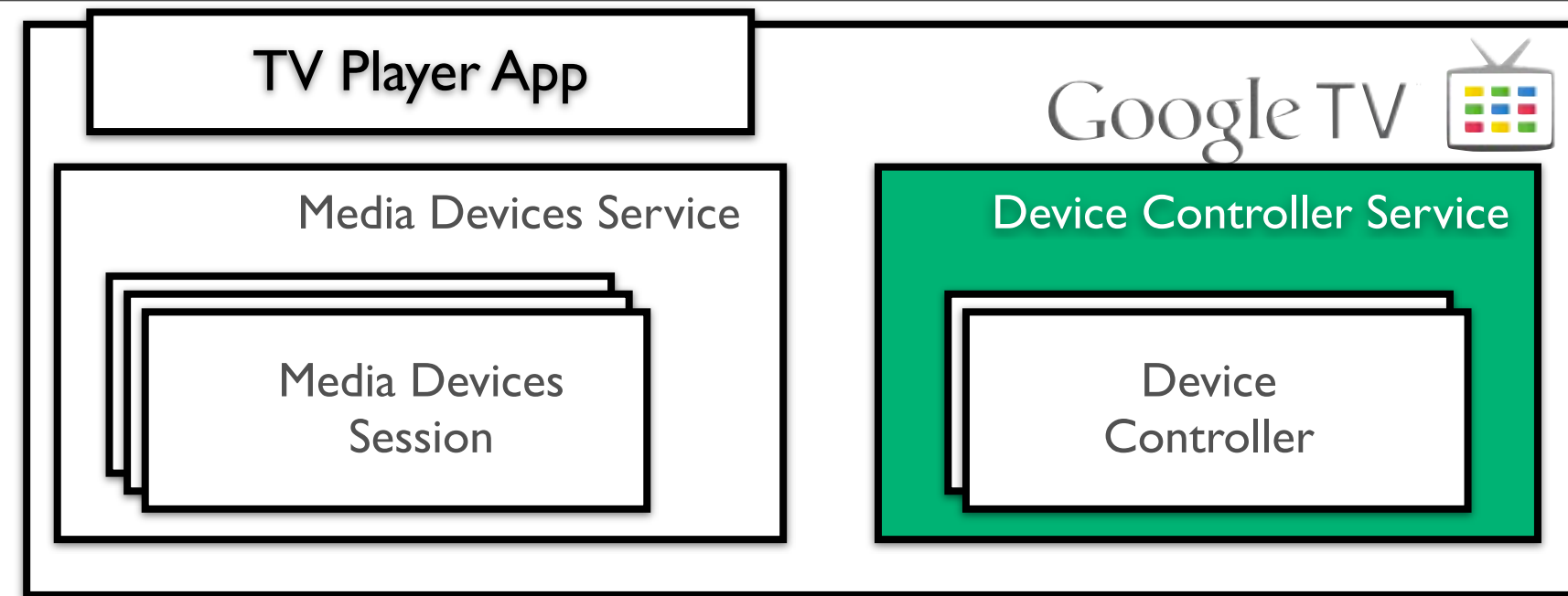
Android



Implementing a Device Controller Service

(Step 2/2)

Subclass **AbstractDeviceControllerService**



- Define and implement device features

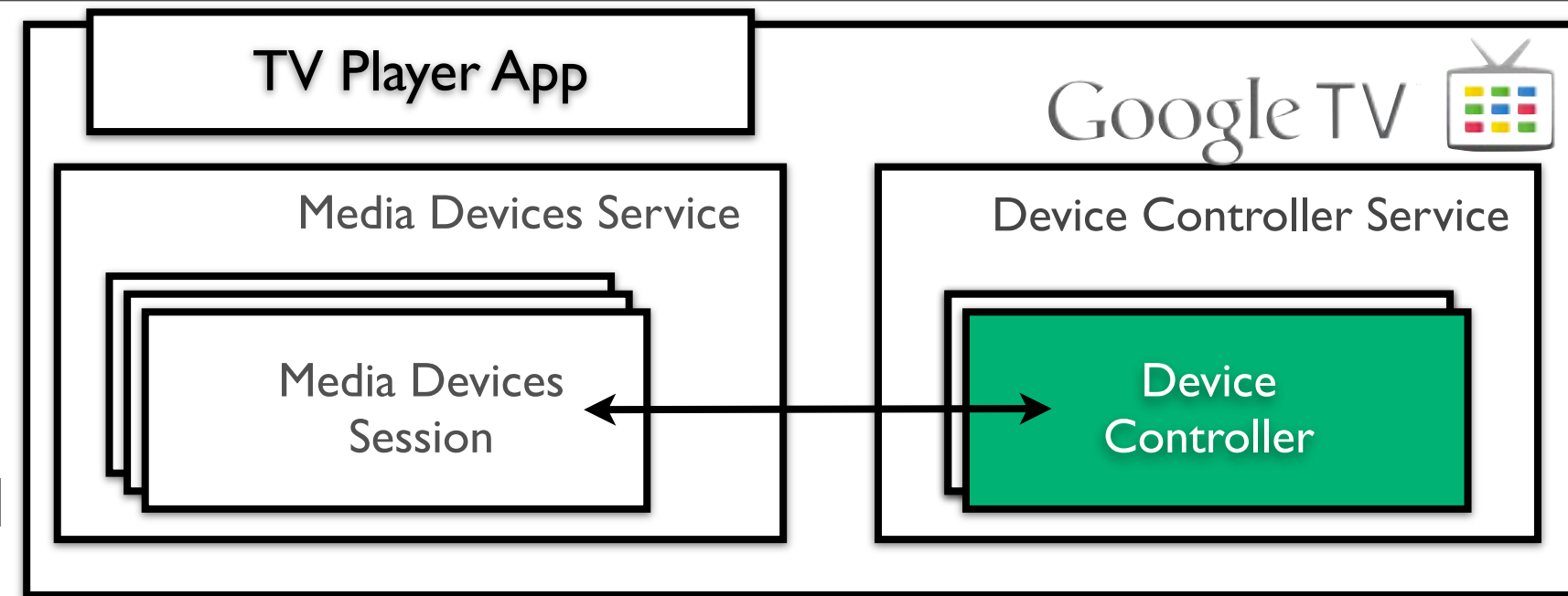
```
private Device buildDevice() {  
    String deviceId = getSettings().generateUniqueId("mydevice");  
    return new Device.Builder(getPackageName(), deviceId)  
        .setLabel(getString(R.string.my_device_label))  
        .setCapability(Capability.CAN_DISCONNECT, true)  
        .setCapability(Capability.HAS_CHANNEL_LINEUP, true)  
        .setCapability(Capability.LOCK_CHANNEL_LINEUP, true)  
        .build();  
}  
protected void checkForChannelUpdates(final String deviceId) {  
    ...  
}  
}
```

Android



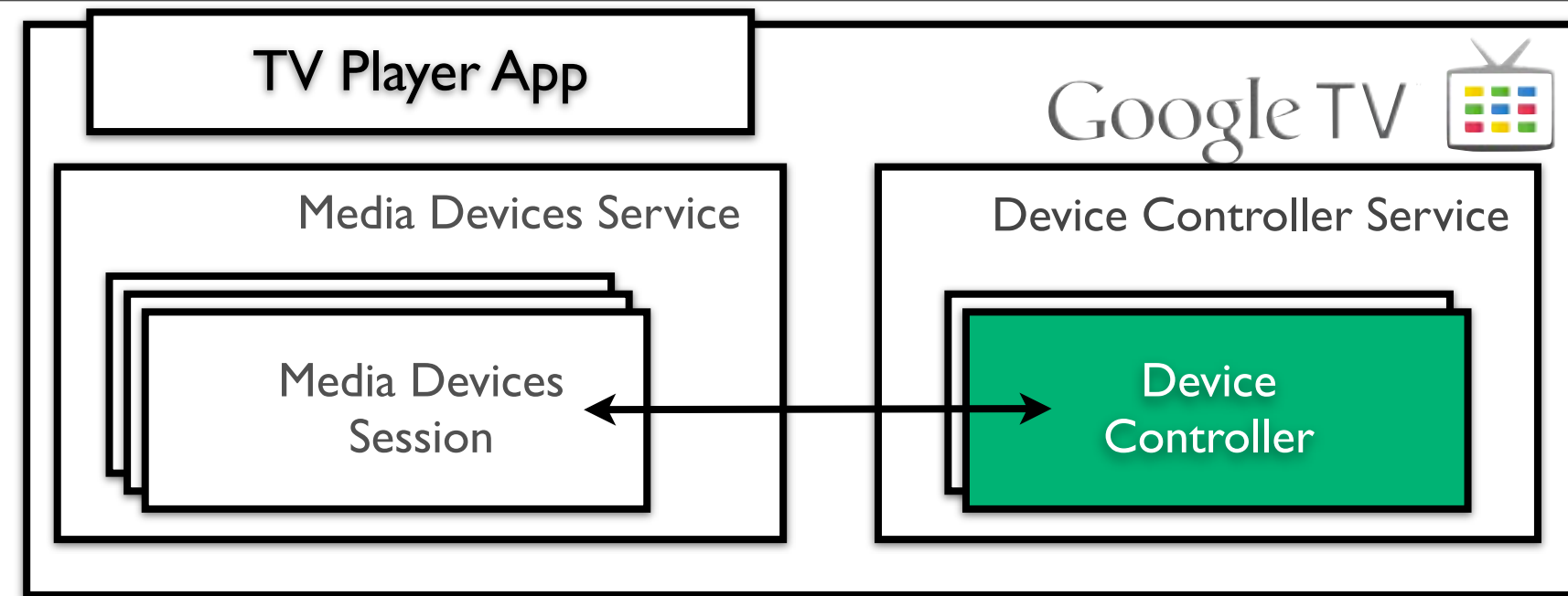
Device Controller

- Communicates to **device** using its **protocol**
- Handles user keypress **events**,
 - Channel Up, Fast Forward, Guide
- Tunes device to requested **URIs** ("tv://...") which represent a channel, a VOD program or a DVR recording, etc.
- Controls the Media Player in the associated Device Session including start, stop playback and can change the players URI
 - e.g., "**hdmi://...**" URIs for passthrough devices
 - "**http://...**" URIs for "virtual" devices streaming video over the Internet



Implementing a Device Controller

Subclass **AbstractDeviceController**



```
public final class MyDeviceController extends AbstractDeviceController {  
    ...  
    public void performAction(ActionEvent event) {  
        switch (event.getAction()) {  
            case CHANNEL_UP:  
                myNextChannel(); break;           // Implement necessary "channel up" change  
            ...  
        }  
    }  
    public void tuneToChannel(ChannelNumber channel) {  
        Uri videoUri = ...;                       // Determine Video URI for the requested channel  
        notifyLocationChanged(videoUri, EventSource.USER); // inform the Session Media Player  
    }  
}
```

Android



Setup Activity / Pairing

Set up DISH Network

Enter code for DISH Network receiver

Use your berlin controller to enter the code shown in the video screen to the right.

Back

Next

Confirmation Code:
90210



You should see your TV signal playing above.



Settings Activity

DISH Network settings

Global

Fast Forward
Swap skip forward and fast forward keys

Rewind
Swap skip back and rewind keys

DVR Information

IP Address
172.17.21.181

Status
Connected

Recordings and FVOD
143 items. Last updated: 7 minutes ago

Channels



"Content" Is King

✓ **Streaming**

✓ • High Quality

✓ • Securely

Integrated ✓

• One Interface ✓



Bringing Content To Google TV:

- **Supports** many different **Streaming** Protocols
 - HTTP Live Streaming
 - Smooth Streaming
 - Optionally - allows to implement your own
- **Compatible** with Industry **Standard DRM** Solutions
 - Widevine
 - Smooth Streaming and PlayReady
 - Trusted Video Path
- **Integrates** with devices in your living room (and in the cloud)
 - Media Devices API



Learn More About Google TV

<http://developers.google.com/tv/>

+Google TV Developers



<Thank You!>

+Google TV Developers

+Christian Kurzke

+ Andrew Jeon

+Mark Lindner



