AnDevCon SF 2014

Your Android App. On TV.

Whadaya Mean, “On TV”?

- What We Are Covering
  - Android apps displaying content on a TV, monitor, projector, etc.
- What We Are Not Covering
  - An Android phone balanced on top of a TV
  - TV commercials featuring Android apps
  - Sitcoms about balding Android presenters

But... But... But... Why?

- The Living Room
  - Entertainment apps, including games
  - Educational apps for children
- The Conference Room
  - Presentations
  - Collaborative workspaces
- Because the Marketing Department said so

Order of Battle

- Intro
- Mobile Direct Connections
- Using `Presentation`
- Direct-to-TV Devices
- Remote Playback Devices
- BREAK

Order of Battle

(Battles Sometimes Take a While...)

- Using `RemotePlaybackClient`
- Implementing `MediaRouteProvider`
- Design and Coding Considerations
- Business Opportunities
- Open Q&A

Axes, Not of Evil (Usually)

- Is There a Touchscreen?
  - Yes: Phones/tablets talking to a TV
  - No: Fire TV, Android TV
- Where is the Android Client App Running?
  - On the device connected to the TV?
  - On another device, directing what the TV-connected hardware should be doing?

Getting to a TV

- Mobile Direct Connection
  - Phones and tablets also pushing to a TV
- Direct to TV
  - Android devices that plug right into a TV
- Remote Playback Devices
  - Non-Android devices that plug right into a TV
  - Directed by Android (and some non-Android) devices

Mobile Direct Connections

Mobile Direct Connections

- The Players: Wired
  - HDMI
    - Some tablets, particularly via micro HDMI ports
  - MHL = Mobile High-Definition Link
    - Many phones, some tablets
  - SlimPort
    - Nexus devices, plus a smattering of others
  - Proprietary
    - Example: Samsung 30-pin to HDMI adapter

Mobile Direct Connections

- The Players: Wireless
  - Miracast
    - A.k.a., AllShare Cast
    - NOTE: Compressed protocol, adds latency
  - Intel WiDi
    - Extension of Miracast
  - WirelessHD
    - Fairly new, still mostly for non-mobile scenarios

Mobile Direct Connections

- Silicon Image
  - Chipset maker for MHL, WirelessHD, and kin
  - Sponsor of AnDevCon!
  - Booth in exhibit hall (Thursday and Friday)

The Gamma Quadrant

	Touchscreen	No Touchscreen
Client App Connects to TV	Mobile Direct Connections	
Client App Connects to Something Else		

What Is Shown

- Android 1.0 through 2.3
  - Generally nothing
  - Sporadic device-specific support (e.g., HTC DROID Incredible and composite output)
- Android 3.0 through 4.1
  - Mirroring
  - Some limited developer control (video players)
- Android 4.2+: Developer Control
  - At least for devices that shipped with 4.2+

Using Presentation

Using `Presentation`

- Subclass of `Dialog`, added in API Level 17
- Override `onCreate()`, call `setContentView()`
- Supplies `Context` suitable for use with designated `Display`
- When displayed using `show()`, appears on designated `Display`

Not All Screens Are Created Equal

- Key Differences
  - Size (720p, 1080p, etc.)
  - Density (`tvdpi`, `hdpi`, etc.)
- Net: Different `Context` for Different Display
  - Inflate layouts, load resources, etc. using proper `Context` to get the right ones for the targeted display

Using `Presentation`

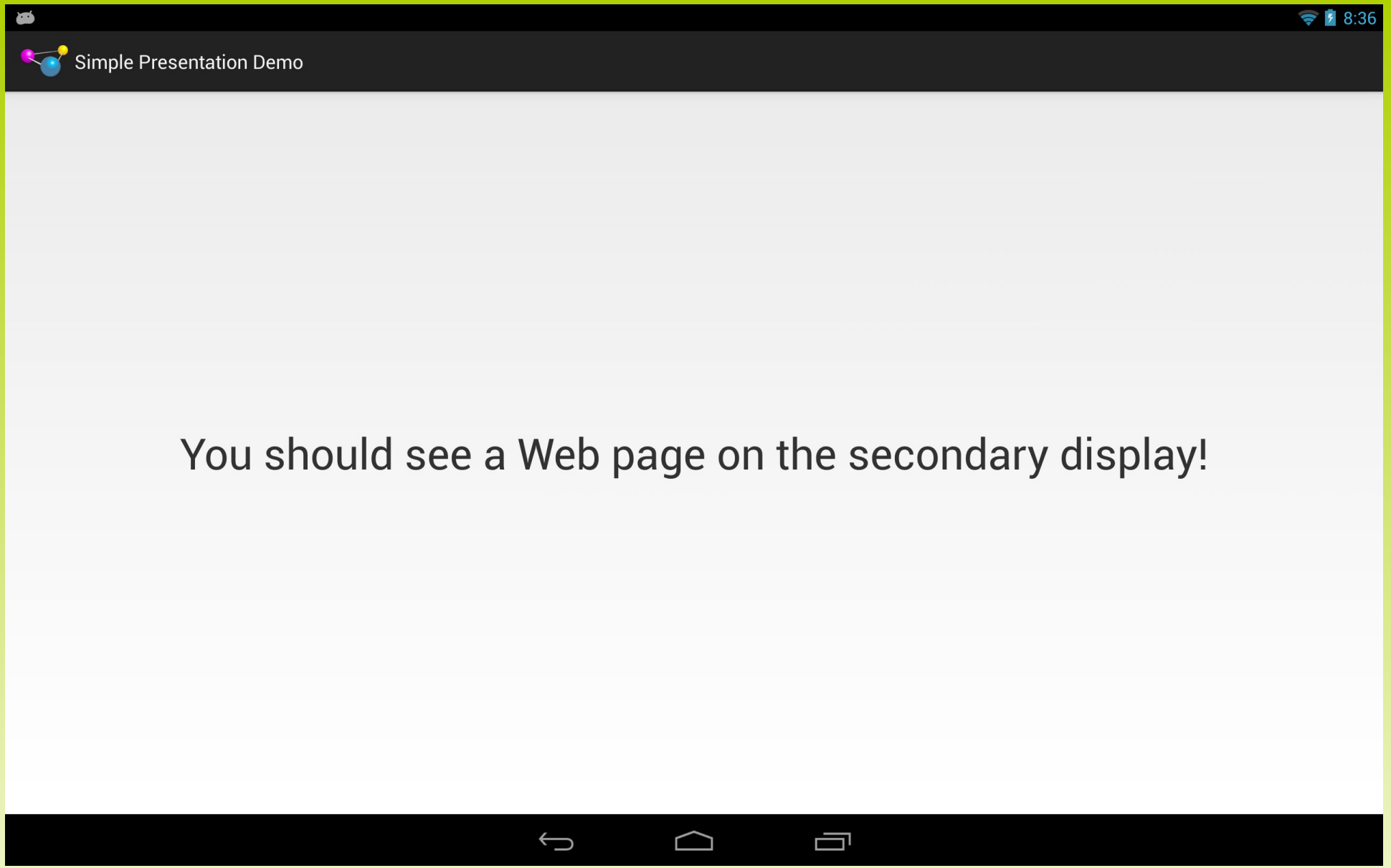
- CWAC-Presentation and PresentationHelper
  - Lets you know when a secondary display becomes available or goes away
  - Usage
    - Create instance
    - Supply a Context and a Listener
    - Forward `onPause()` / `onResume()`
    - Implement `showPreso()` / `clearPreso()`

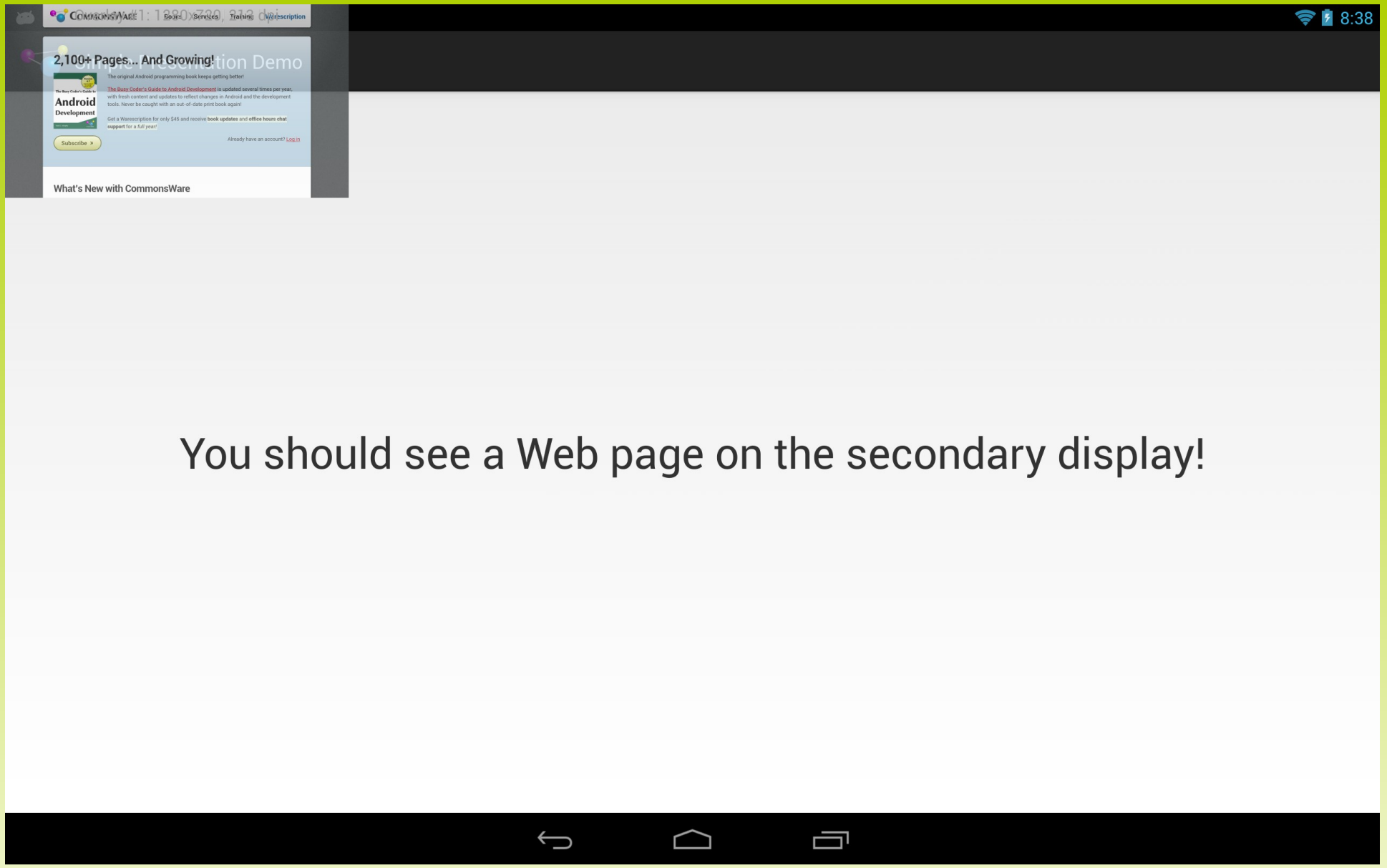
Fragments of a Presentation

- Good News!
  - `Presentation` extends `Dialog`, so `DialogFragment` works
  - `DialogFragment` supports either show-as-dialog or show-as-regular-fragment
- Bad News!
  - Since different `Contexts`, cannot use the same `DialogFragment` instance for each

Trying It Out

- Testing Options
  - Actual “secondary screen”
  - Simulated secondary display
    - Developer Options → Simulate secondary displays
    - Works well with hardware, less so with x86 emulator





Low-Level Exertion

- `DisplayManager`
  - System service (`DISPLAY_SERVICE`)
  - `getDisplays()` (all or those in a category, like `DISPLAY_CATEGORY_PRESENTATION`)
  - `registerDisplayListener()` to find out about changes in mix of displays
  - New to API Level 17

Low-Level Exertion

- MediaRouter
  - System service (`MEDIA_ROUTER_SERVICE`)
    - Better: MediaRouter from Android Support package
  - More general: find preferred “route” for audio or video
  - `getSelectedRoute()` to find the route to be used right now (`ROUTE_TYPE_LIVE_VIDEO`)
  - `addCallback()` to find out route changes
  - Added in API Level 16

Secondary Screen Strategies

- Secondary Screen: Not a Touchscreen
  - Whatever shows on secondary screen is driven and controlled by what is on the primary screen
- Example: Video Player
  - Second screen shows the video playback
  - Primary screen has play/pause, fast-forward, rewind, SeekBar, IMDB content, chat screen, etc.

Secondary Screen Strategies

- Must Have Secondary Screen
  - Presentation-specific app
- Optional, Dedicated UX
  - Game using second screen for primary output, device screen for control surface and secondary output
- Optional, Using Fragments
  - Push stuff to second screen or show “inline”

Secondary Screen Tactics

- Separate Fragments/Views
  - One for what is shown
  - One for controlling what is shown
- Mirroring Content
  - Same fragment/view, shown on both displays, controlled by primary screen

PresentationService

- Subclass of `Service`
- Tie to a `Display` via `DisplayManager` or `MediaRouter`
- `View` returned from `buildPresoView()` displayed on external display
- Key limitation: third-party library from some balding guy

Direct-to-TV Devices

Direct-to-TV Devices

- Android TV
  - Google's 3rd try for the living room
  - Reference hardware: Nexus Player
- Game Consoles
  - OUYA, etc.
- Android HDMI Sticks
  - eBay, Alibaba, etc.

Direct-to-TV Devices

- Amazon Fire TV / Fire TV Stick
  - Also serve as Miracast connector, for mobile direct connection approach
    - Supposedly: cannot get it to work
  - Differences
    - Stick is smaller, akin to Chromecast
    - Stick is a bit less powerful

Direct-to-TV Devices

- “Reach More Customers by Bringing Your App to the Living Room with Amazon Fire TV Devices”
  - Mike Hines, Amazon
  - Wednesday, 1:45pm - 2:15pm

The Delta Quadrant

	Touchscreen	No Touchscreen
Client App Connects to TV		Direct-to-TV Devices
Client App Connects to Something Else		

All Systems Normal

- Direct-to-TV Development: Standard Stuff
  - Activities, fragments, widgets, containers
  - Ten-foot design considerations play a role
    - E.g., no action bar
  - Big key: do not assume a touchscreen!
    - Design/coding impacts on navigation, etc.
    - Manifest: `<uses-feature>` to opt into non-touchscreen devices

I Iz A Tee Vee?

- Detecting Direct-to-TV
  - Means
    - `PackageManager.FEATURE_TELEVISION` and `PackageManager.FEATURE_LEANBACK`
    - `-television` resource set qualifier
  - Uses
    - Hide portions of a tablet-y UI (e.g., action bar)
    - Enable/disable activities via boolean resource and `android:enabled`

Remote Playback Devices

Remote Playback Devices

- Chromecast
  - Also “mobile direct connection” via mirroring support
- Android TV (via Cast SDK)
  - ADT-1 does not seem to support open remote playback protocol
- Matchstick
  - In-progress Kickstarter for Firefox OS-based Chromecast competitor

Remote Playback Devices

- Roku
  - Also Miracast capable... if it actually worked
- Smart TV
  - The “smart” is the stuff running on the TV itself, perhaps controlled by a mobile device
- Direct-to-TV Devices
  - When the app that is running is controlled by a mobile device

The Alpha Quadrant

	Touchscreen	No Touchscreen
Client App Connects to TV		
Client App Connects to Something Else	Remote Playback Devices	

Remote Playback APIs

- RemotePlaybackClient
  - Part of Android and Android Support package
    - Open source FTW!
  - Extensible: anyone can write a MediaRouteProvider to support their own hardware
    - Would be installed by user on their mobile device to enable communications to the desired remote playback device
  - Supports Chromecast out of the box... if it actually worked

Remote Playback APIs

- Cast SDK
  - Proprietary API, part of Play Services
  - Three tiers of “receivers”
    - Default = what RemotePlaybackClient uses
    - Styled = you provide CSS to tailor a bit what is shown on-screen when the client “casts” something
    - Custom = you provide full HTML5 Web app that is controlled by the client
  - Chromecast, Android TV

Remote Playback APIs

- “Bringing Android Content to the Big Screen with Google Cast”
  - Kevin Nilson, Google
  - Wednesday, 4:00pm - 5:15pm

Remote Playback APIs

- Connect SDK
  - Open source, sponsored by LG
  - Wrapper around other APIs
    - Documented and... not so documented
  - Supports Chromecast, Roku, Fire TV, Apple TV, and various LG smart TVs
  - Mostly for remote media playback, with limited support for Chromecast-style “custom receivers”

Remote Playback APIs

- Matchstick Fling SDK
  - Open source (OpenFlint???)
  - Same basic concept as Cast SDK, with slight change in verb
  - Unclear if this will be promoted as any sort of open specification for use in other areas

COFFEE

MediaRouter and RemotePlaybackClient

Enter the MediaRouter

- Added in API Level 16
- Manages media routes
  - Live audio (e.g., Bluetooth speakers)
    - Added API Level 16
  - Live video (e.g., HDMI, MHL, Miracast)
    - Added API Level 17
  - Remote playback (e.g., Chromecast)
    - Added... well, this gets complicated

Your MediaRouter Choices

- android.media.MediaRouter
  - Built into Android
  - Supports live audio and live video
- android.support.v7.media.MediaRouter
  - Ships with Android Support package, in mediarouter-v7 library project
  - Supports live audio, live video, and remote playback
  - Hint: you probably want this one

Interacting with a `MediaRouter`

- Get an instance (e.g., `getInstance()`)
- Build a `MediaRouteSelector`
  - Identifies types of routes you are interested in
- Add `Callback` via `addCallback()` to find out about changes in route availability
  - Tied to the `MediaRouteSelector`
  - Key method: `onRouteSelected()`

MediaRouteActionProvider

- A.k.a., “the cast button”
  - Technically, there's a `MediaRouteButton` that's the “cast” button, but we'll ignore that for now...
- Roles
  - Tapped, brings up dialog to choose a media route
    - Of categories of relevance to app (e.g., remote playback)
  - Shows highlight to indicate when app is connected to a media route or not

OK, Which `MediaRouteActionProvider`?

- Option #1: `android.media`
  - Works with native `MediaRouter`, native action bar
  - Not a great choice
- Option #2: `android.support.v7.media`
  - Works with Android Support's `MediaRouter`, `AppCompat` action bar
- Option #3: CWAC-MediaRouter cross-port
  - Works with Android Support's `MediaRouter`, native action bar

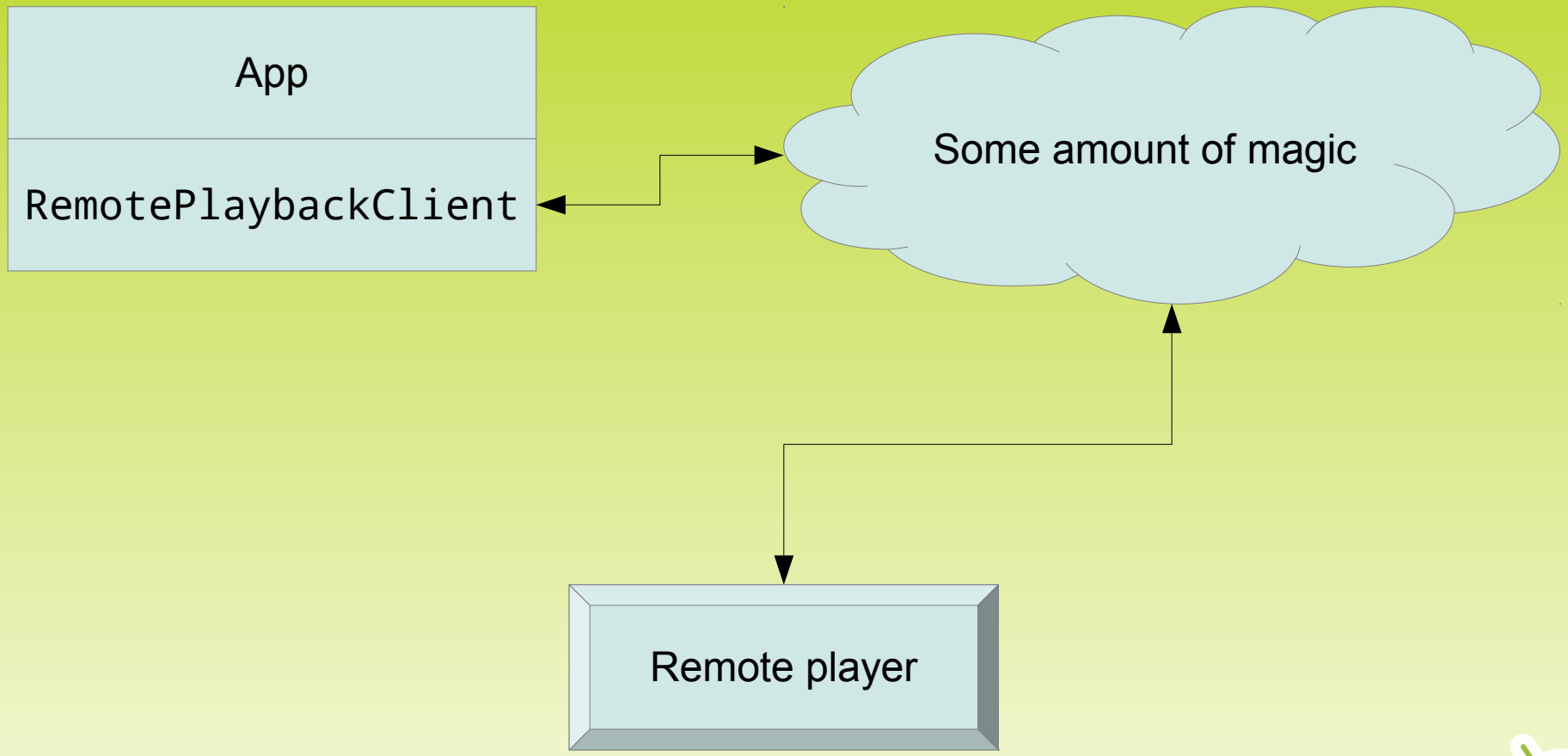
Wiring Up the Provider

- Add to menu resource
  - Use right `MediaRouteActionProvider` class given your chosen action bar implementation!
- Call `setRouteSelector()` on the `MediaRouteActionProvider`
  - Controls which routes it pays attention to
  - Usually the same `MediaRouteSelector` that you used with `addCallback()`

RemotePlaybackClient

- Client API for working with remote players
- Basic Mechanics
  - Get an instance
  - Tie to a media route
  - Call methods like `play()`, `pause()`, `resume()`, ...
    - E.g., `play()` takes URL to play back on remote player

The RemotePlaybackClient Flow



Using `RemotePlaybackClient`

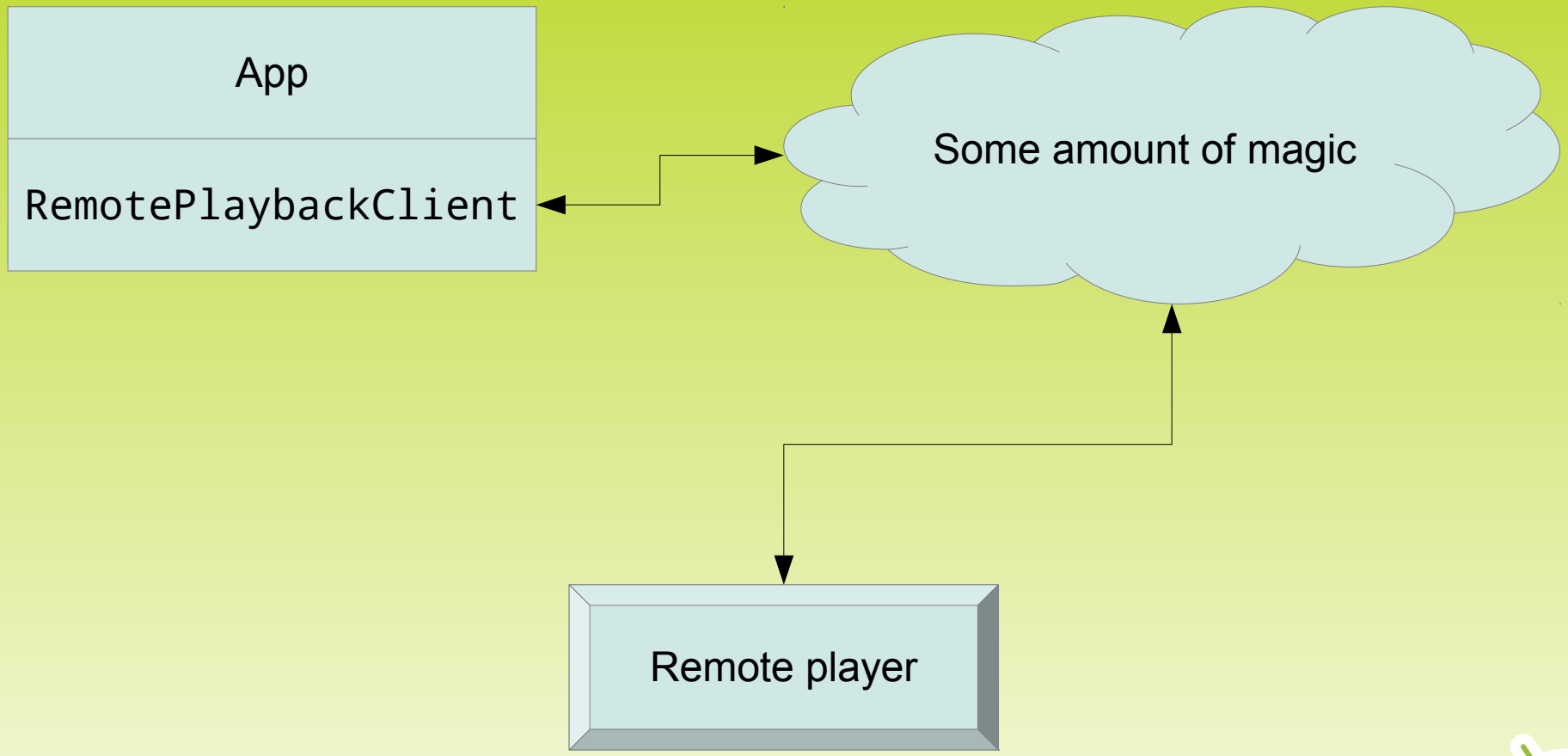
- Create instance
- Connect to route
- Call control methods
  - `play()`, `pause()`, etc.
  - Callbacks to find out success or failure... in theory
- Disconnect and release

Other RemotePlaybackClient Capabilities

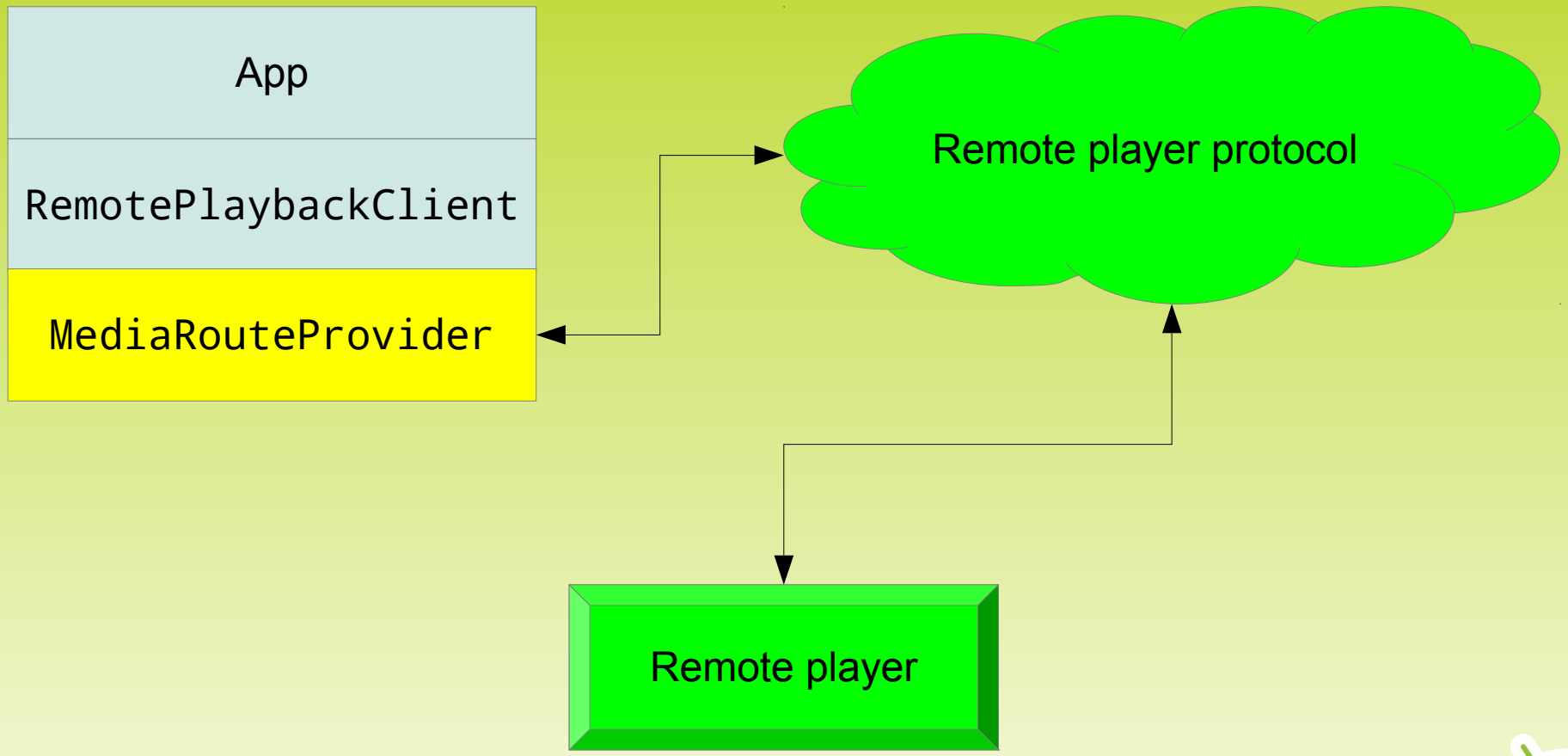
- Volume control
- Session management
  - Playback queues, managed by different users
- `RemoteControlClient`
  - Controls on the device's lockscreen
  - Obsolete as of Android 5.0 – use media control notification on the lockscreen instead

Implementing MediaRouteProvider

The RemotePlaybackClient Flow



The RemotePlaybackClient Flow



MediaRouteProvider

- Bridge Between Client and Player
  - Converts media actions like play and pause into whatever it takes to tell the player to do those actions
- Plugin
  - Distribute `MediaRouteProvider` to client devices as part of an app
  - Chromecast/Android TV `MediaRouteProvider` distributed via Google

`MediaRouteProvider` Examples

- “Mock” implementation for automated testing
- Browser Playback
  - Web page opens WebSocket, shows QR code with IP information
  - Device scans QR code to configure `MediaRouteProvider`
  - Cast to the Web page, which plays back media

MediaRouteProvider Examples

- Alternative Ecosystems
  - Write media player app for Fire TV, OUYA, etc.
  - Write `MediaRouteProvider` for use on client device to talk to your app running on TV-connected device for remote control
- Alternative Hardware
  - Embedded media player, exposing protocol over Android-capable wireless (e.g., Bluetooth)
  - `MediaRouteProvider` converts commands to requests over this wireless protocol

Who Uses Your Provider?

- Your Client App
  - Ensures that if your `MediaRouteProvider` is installed, the client device has something to "cast" content to it
- Any Other `RemotePlaybackClient`-using App
  - Ideally, some decent subset of those for developers aiming at Chromecast
  - Will help once more remote players have market presence

Implementing `MediaRouteProvider`

- Create a `MediaRouteProvider`
- Register it for specific routes
  - Via specially-crafted `IntentFilters`
- Supply a `RouteController` to handle actions
  - Details are up to you
- Create a `MediaRouteProviderService` to publish `MediaRouteProvider`, add to manifest

Other `MediaRouteProvider` Capabilities

- Volume control
- Session management
  - Playback queues, managed by different users

Design & Coding Considerations for TV

Elements of 10-Foot UI

- Overscan
  - Legacy of analog TV signals and tube-based sets
  - Net: entire screen resolution may not be accessible
  - Net net: keep important stuff away from edges!
    - So if it gets cropped, no biggie
    - ~10% padding on each side
    - Background can extend to edge

Elements of 10-Foot UI

- Navigation for Remote Controls
  - No touchscreen, so no random access of widgets
  - Focus modes important!
    - May work out of the box, may require tweaking via `android:nextFocusDown`, etc.
    - Bonus: your app is more accessible as a result

Elements of 10-Foot UI

- Navigation for Remote Controls
  - Navigation typically orthogonal to scrolling
  - No navigation “after” scrollable region
  - Use item selection events to automatically change UI
    - Example: showing details of some item
    - Item click events are for taking positive action, such as initiating playback

Elements of 10-Foot UI

- Fonts
  - Larger with medium weight
    - Configurable by user, if possible
  - Simpler fonts (sans serif, compared to serif)
  - Use fewer words
  - Use more line spacing

Elements of 10-Foot UI

- Colors
  - Avoid pure white (#FFFFFF), use light grey (#F1F1F1 or #EBEBEB) to avoid ghosting
  - Use muted colors
  - Use cooler colors (green, blue, violet)
    - Less bleeding than warmer colors (red, orange)

Leanback

- Uses of the Term
  - Intent filter category for Android TV launcher
  - “Feature” for Android TV-style devices (for use with <uses-feature>)
  - Android Support library (`leanback-v17`)
  - What one does on a sofa with respect to the TV
  - What one does in an office chair in a conference room
    - Until you lean too far back and topple over, embarrasing yourself in front of your boss and threatening your career

Leanback Support Library

- Standard Android TV UI Elements
  - BrowseFragment and related classes
    - Two-dimensional browsing of content
  - SearchFragment
    - Searching for content
  - PlaybackOverlayFragment
    - Popup controller for media playback
  - Etc.

Leanback Support Library

- Using BrowseFragment
  - Extend, add to an activity
  - Call Key Setters
    - `setTitle()`
    - `setHeadersState()`
  - Configure ObjectAdapters
    - Establishes two-dimensional roster of headers, content within a header
    - Uses Presenters to render model data
    - Attach to BrowseFragment via `setAdapter()`

Leanback Support Library

- Using BrowseFragment
  - Call `setOnItemClickListener()`
    - Launch details or content from there
  - Other Configuration
    - Colors
    - Badges
    - Search options
- Compatible with anything!
  - Not tied to Android TV
  - Only needs API Level 17+

Coding Key: Intermittency

- In Most Cases, TV Is Not Always There
  - User is not at home
  - User is not at the office, and the app was set up to work with some office group meeting rig
  - User is at home, but the TV is being used by somebody else, and the user is too tired to go to some other room and set up the TV in there, because the user had a really long day at work trying to get the office group meeting rig to work...

Coding Key: Intermittency

- Strategy #1: TV is Purely Optional
  - App is usable on mobile device with touchscreen
  - TV simply adds more output possibilities
- Strategy #2: TV is Essential
  - Direct-to-TV devices
  - App is pointless without secondary display
    - Example: app for delivering a conference presentation

Keeping Code Consolidated

- Mobile Direct Connection
  - Think fragments, `ViewGroups`, etc. that can be conditionally shown on the touchscreen or on an external display
- Remote Playback Devices, Media Playback
  - Offer local media playback (e.g., `VideoView`) as fallback

Keeping Code Consolidated

- Remote Playback Devices, Custom Receivers
  - Use the HTML/CSS/JS of custom receiver in a WebView for when remote playback device is unavailable?
  - Use some native replacement?

Ideas for Business Opportunties

Media Players

- Support Background Operation with PresentationService
- Support Direct-to-TV Devices
  - Local use
  - Directed by MediaRouteProvider and RemotePlaybackClient

The Conference Room

- Mobile Direct Connection
  - Group visuals on TV, controller on device(s)
- Direct-to-TV
  - Group visuals on TV
  - Accept input from mobile devices
    - Dedicated protocol
    - `RemotePlaybackClient`
  - Also possible with remote playback devices, but more challenging to implement

Games

- Mobile Direct Connection
  - Game on TV, controller and secondary info on mobile
- Direct-to-TV
  - Game on TV, using game controllers
- Remote Playback Devices
  - Good for casual games and perhaps more
  - Basically a Web game with mobile device controller

Dedicated Hardware

- Matchstick-style Streaming Players
  - Off-the-shelf hardware and Android
  - Focusing more on what software you can provide
- Edutainment Consoles
  - Off-the-shelf hardware and Android
  - Focusing more on software and controller hardware, with eye towards children
- Kickstarter, etc. as avenues for initial customers

Summary

The Roster of Quadrants

	Touchscreen	No Touchscreen
Client App Connects to TV	Mobile Direct Connections	Direct-to-TV Devices
Client App Connects to Something Else	Remote Playback Devices	The Undiscovered Country