CS 528 Mobile and Ubiquitous Computing

Lecture 5a: Playing Sound and Video

Emmanuel Agu

Multimedia Networking: Basic Concepts

Multimedia networking: 3 application types

- Multimedia refers to audio and video

1. *streaming, stored* audio, video
   - *streaming:* transmit in batches, begin playout before downloading entire file
   - e.g., YouTube, Netflix, Hulu
   - Streaming Protocol used (e.g. Real Time Streaming Protocol (RTSP), HTTP streaming protocol (DASH))

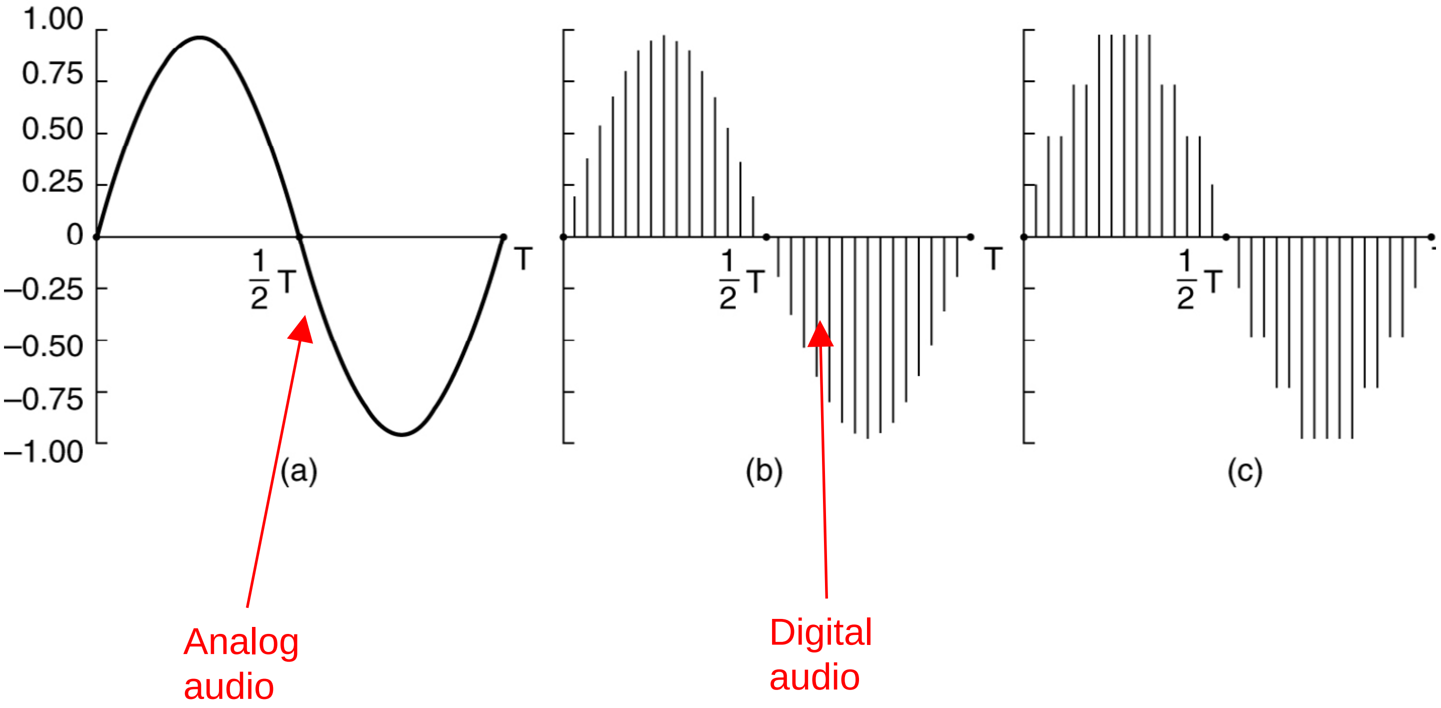
2. *streaming live* audio, video
   - e.g., live sporting event (futbol)

3. *conversational* voice/video over IP
   - Requires minimal delays due to interactive nature of human conversations
   - e.g., Skype, RTP/SIP protocols

Credit: Computer Networks (6th edition),
By Kurose and Ross

Digital Audio

- Sender converts audio from analog waveform to digital signal
- E.g PCM uses 8-bit samples 8000 times per sec
- Receiver converts digital signal back into audio waveform



Audio Compression

- Audio CDs:
  - 44,100 samples/second
  - Uncompressed audio, requires 1.4Mbps to transmit real-time

- Audio compression reduces transmission bandwidth required
  - E.g. MP3 (MPEG audio layer 3) compresses audio down to 96 kbps

Video Encoding

- **Digital image:** array of <R,G,B> pixels
- **Video:** sequence of images
- **Redundancy:** Consecutive frames mostly same (1/30 secs apart)
- **Video coding (e.g. MPEG):** use redundancy *within* and *between* images to decrease # bits used to encode video
  - **Spatial** (within image)
  - **Temporal** (from 1 image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of times repeated* (N)

frame *i*

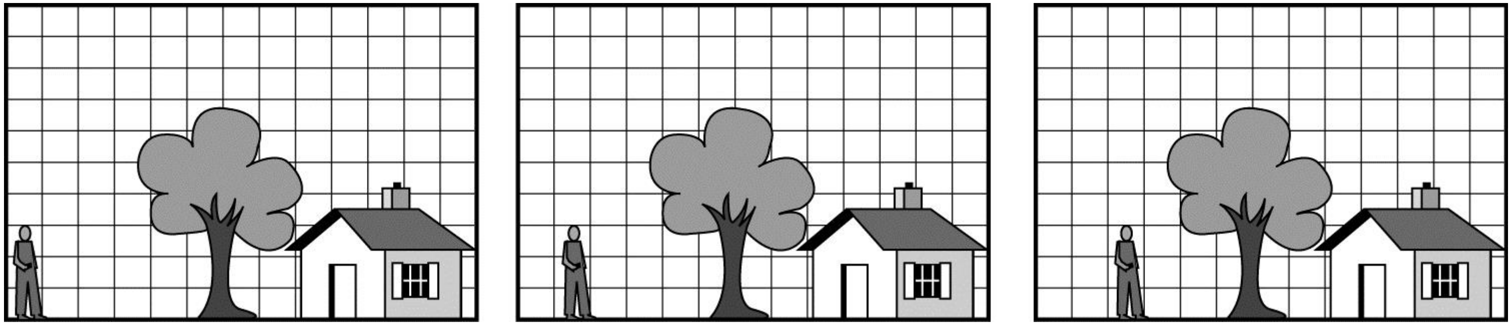
temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

Credit: Computer Networks (6th edition),
By Kurose and Ross

MPEG-2: Spatial and Temporal Coding Example

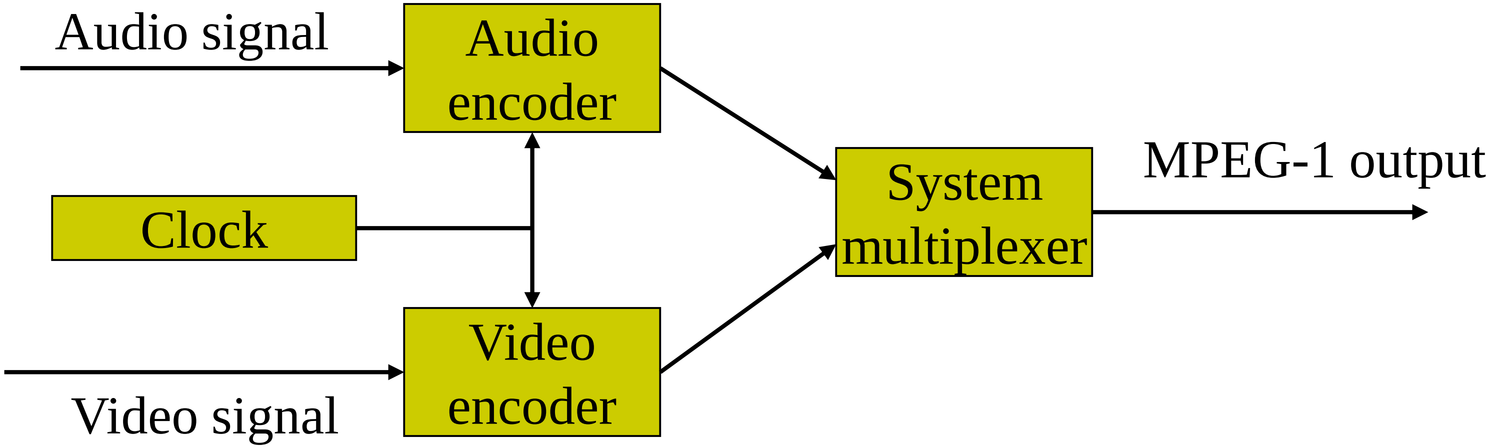
- MPEG-2 output consists of 3 kinds of frames:
  - **I (Intracoded)** frames:
    - JPEG-encoded still pictures (self-contained)
    - Acts as reference, if packets have errors/lost or stream fast forwarded
  - **P (Predictive)** frames:
    - Encodes difference between a block in this frame vs same block in previous frame
  - **B (Bi-directional)** frames:
    - Difference between a block in this frame vs same block in the last or next frame
    - Similar to P frames, but uses either previous or next frame as reference



3 consecutive frames

MPEG Generations

- Different generations of MPEG: MPEG 1, 2, 4, etc
- MPEG-1: audio and video streams encoded separately, uses same clock for synchronization purposes



- Sample MPEG rates:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

Playing Audio and Video in Android

MediaPlayer

http://developer.android.com/guide/topics/media/mediaplayer.html

- Classes used to play sound and video in Android
  - **MediaPlayer:** Plays sound and video
  - **AudioManager:** plays only audio

- MediaPlayer can fetch, decode and play audio or video from:
  - Audio/video files stored in app's resource folders (e.g. **res/raw/** folder)
  - External URLs (over the Internet)

- Any Android app can use MediaPlayer APIs to integrate video/audio playback functionality

MediaPlayer

http://developer.android.com/guide/topics/media/mediaplayer.html

- MediaPlayer supports:
  - **Streaming network protocols:** RTSP, HTTP streaming
  - **Media Formats:**
    - Audio (MP3, AAC, MIDI, etc),
    - Image (JPEG, GIF, PNG, BMP, etc)
    - Video (MPEG-4, H.263, H.264, H.265 AVC, etc)

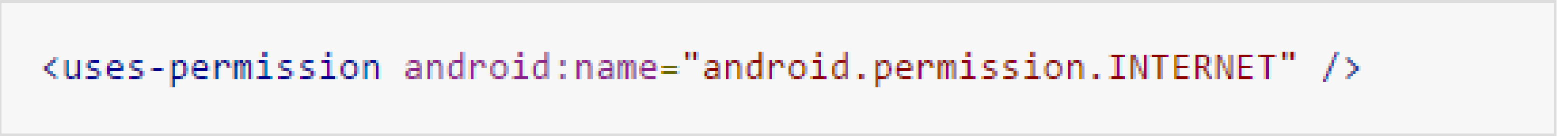
- 4 major functions of a Media Player
  - **User interface**, user interaction
  - Handle **Transmission errors**: retransmissions, interleaving
  - **Decompress** audio
  - **Eliminate jitter:** Playback buffer (Pre-download 10-15 secs of music)

Using Media Player:

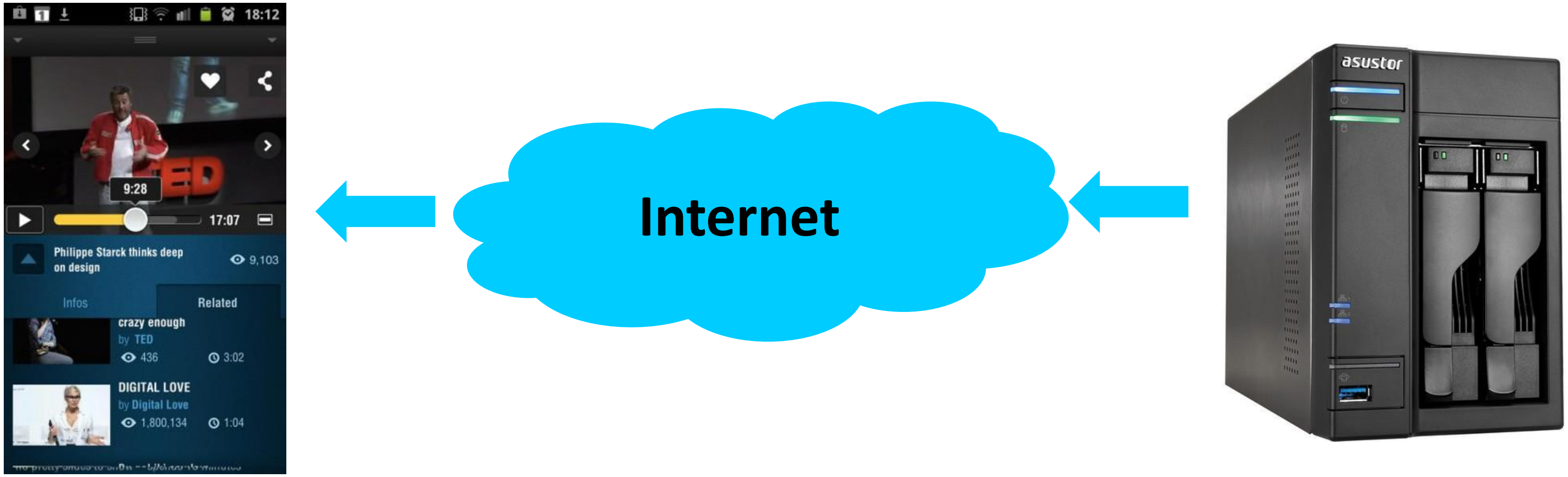
http://developer.android.com/guide/topics/media/mediaplayer.html

Step 1: Request Permission in AndroidManifest or Place video/audio files in res/raw

- If streaming video/audio over Internet (network-based content), request network access permission in AndroidManifest.xml:



```
<uses-permission android:name="android.permission.INTERNET" />
```

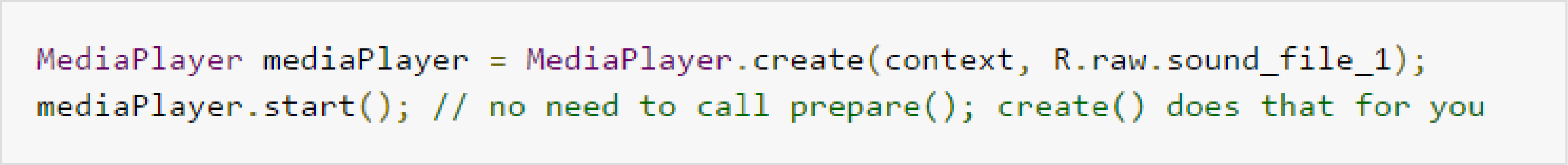


- If playing back local file stored on user’s smartphone, put video/audio files in **res/raw** folder

Using MediaPlayer

Step 2: Create MediaPlayer Object, Start Player

- To play audio file saved in app’s **res/raw/** directory



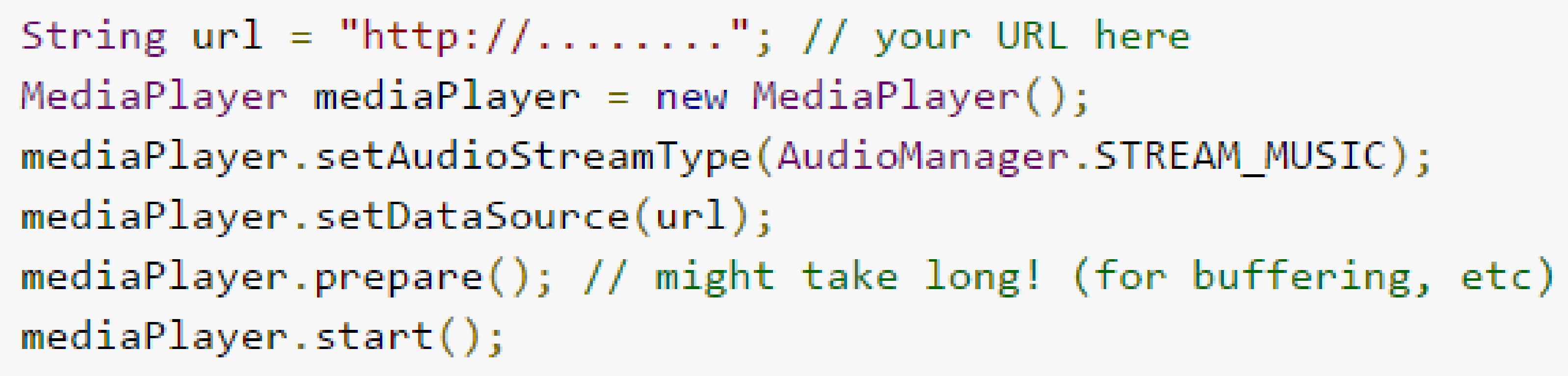
```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

- **Note:** Audio file opened by create (e.g. sound_file_1.mpg) must be encoded in one of supported media formats

Using MediaPlayer

Step 2: Create MediaPlayer Object, Start Player

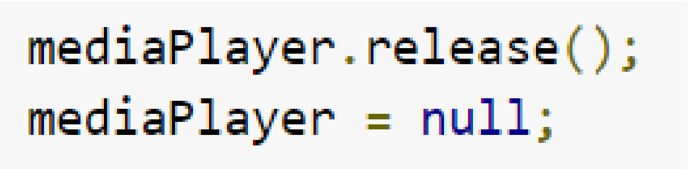
- To play audio from remote URL via HTTP streaming over the Internet



```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Releasing the MediaPlayer

- MediaPlayer can consume valuable system resources
- When done, call **release( )** to free up system resources
- In **onStop( )** or **onDestroy( )** methods, call



```
mediaPlayer.release();
mediaPlayer = null;
```

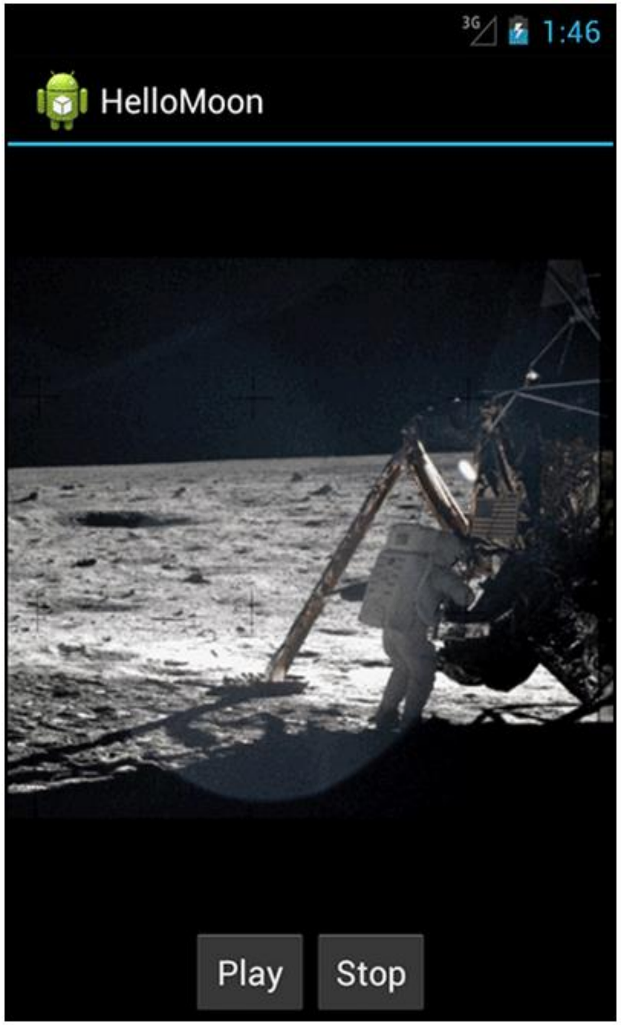
- **MediaPlayer in a Service:** Can play media (e.g. music) in background while app is not running
  - Start MediaPlayer as service

Playing Audio File using MediaPlayer Example from Android Nerd Ranch 1st edition

MediaPlayer Example to Playback Audio

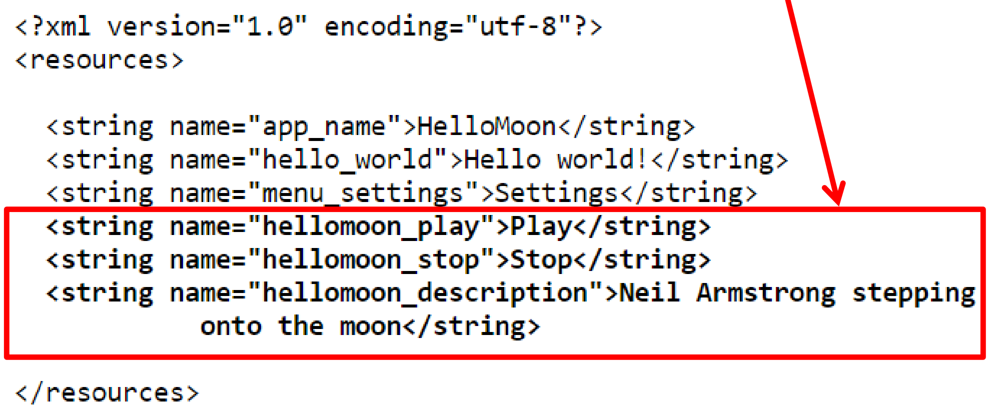
from Android Nerd Ranch (1st edition) Ch. 13

- **HelloMoon app** that uses **MediaPlayer** to play audio file



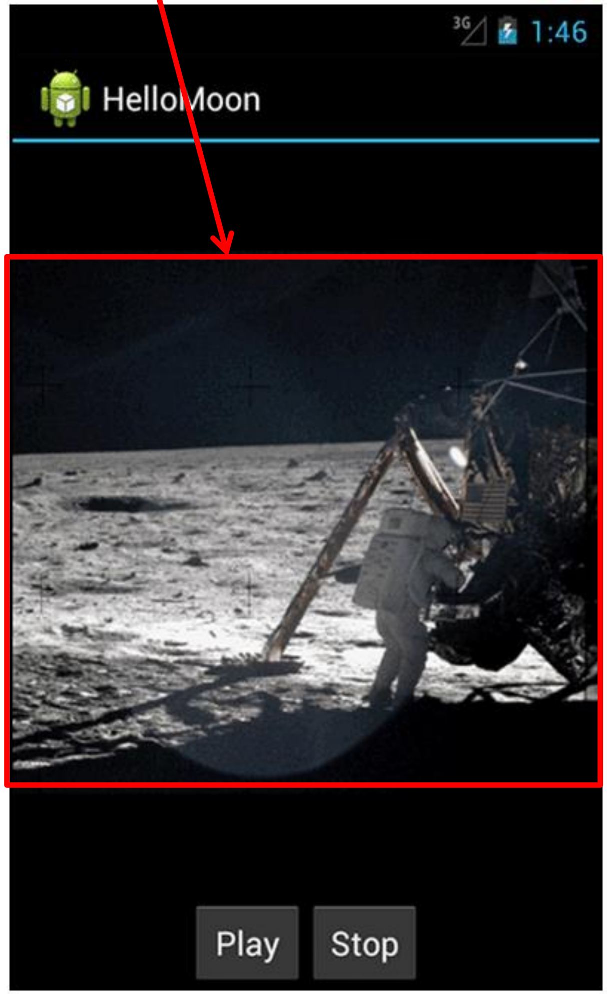
HelloMoon App

- Put image **armstrong_on_moon.jpg** in **res/drawable/** folders
- Place audio file to be played back (**one_small_step.wav**) in **res/raw** folder
- Create **strings.xml** file for app



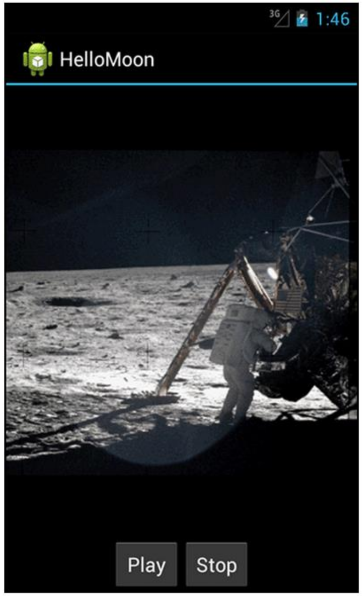
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

  <string name="app_name">HelloMoon</string>
  <string name="hello_world">Hello world!</string>
  <string name="menu_settings">Settings</string>
  <string name="hellomoon_play">Play</string>
  <string name="hellomoon_stop">Stop</string>
  <string name="hellomoon_description">Neil Armstrong stepping
            onto the moon</string>

</resources>
```

armstrong_on_moon.jpg

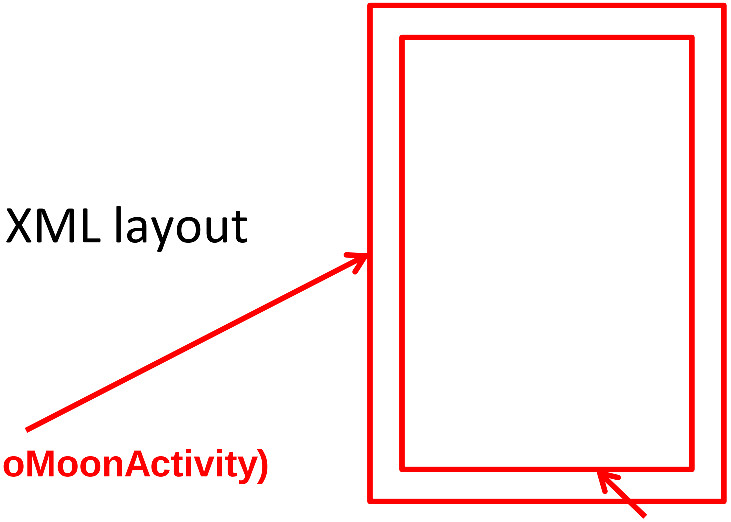


HelloMoon App

- HelloMoon app will have:
  - 1 activity (**HelloMoonActivity**) that hosts **HelloMoonFragment**
- **AudioPlayer** class will be created to encapsulate **MediaPlayer**



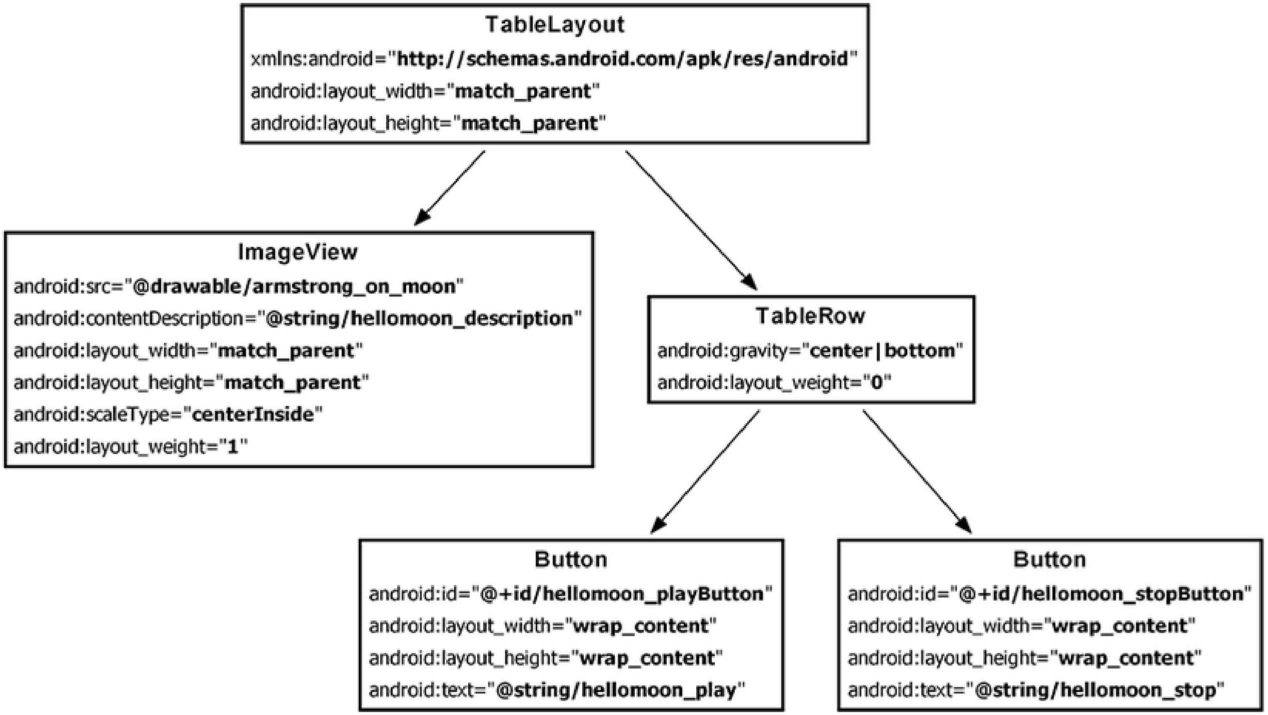
- First set up the rest of the app:
  1. Define fragment's XML layout
  2. Create fragment java class
  3. Modify the activity (java) and its XML layout to host the fragment



Activity (HelloMoonActivity)

Fragment (HelloMoonFragment)

Defining the Layout for HelloMoonFragment



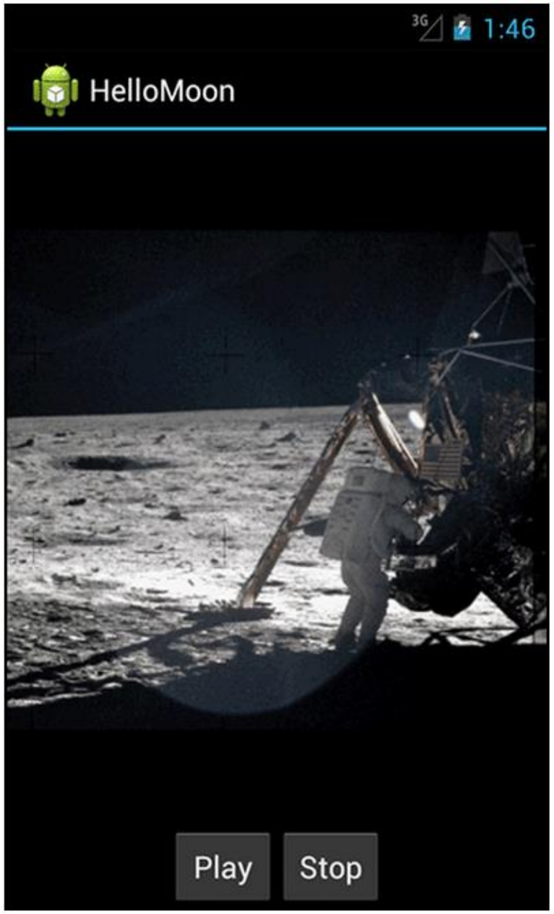
TableLayout
xmlns:android="**http://schemas.android.com/apk/res/android**"
android:layout_width="**match_parent**"
android:layout_height="**match_parent**"

ImageView
android:src="**@drawable/armstrong_on_moon**"
android:contentDescription="**@string/hellomoon_description**"
android:layout_width="**match_parent**"
android:layout_height="**match_parent**"
android:scaleType="**centerInside**"
android:layout_weight="**1**"

TableRow
android:gravity="**center|bottom**"
android:layout_weight="**0**"

Button
android:id="**@+id/hellomoon_playButton**"
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/hellomoon_play**"

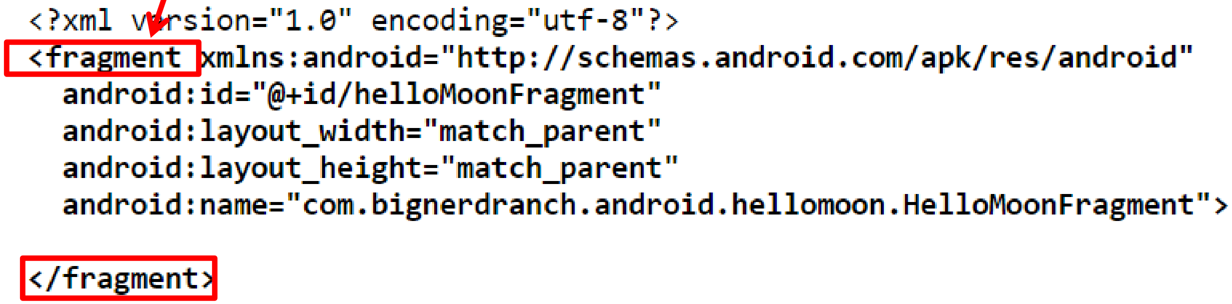
Button
android:id="**@+id/hellomoon_stopButton**"
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/hellomoon_stop**"



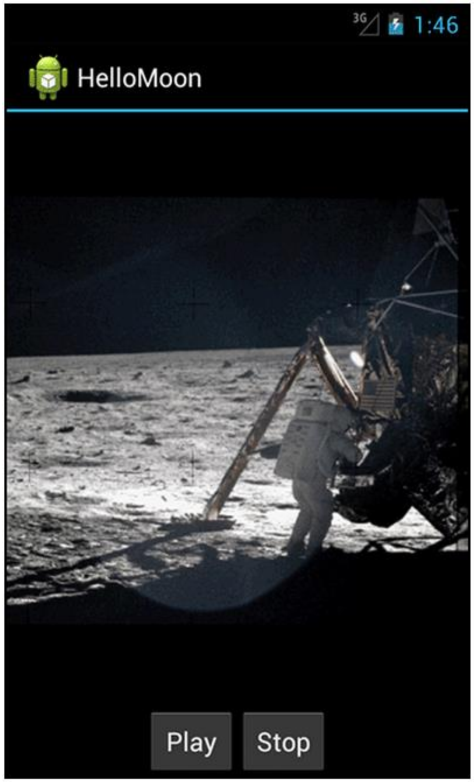
Define XML for HelloMoon UI (fragment_hello_moon.xml)

Creating a Layout Fragment

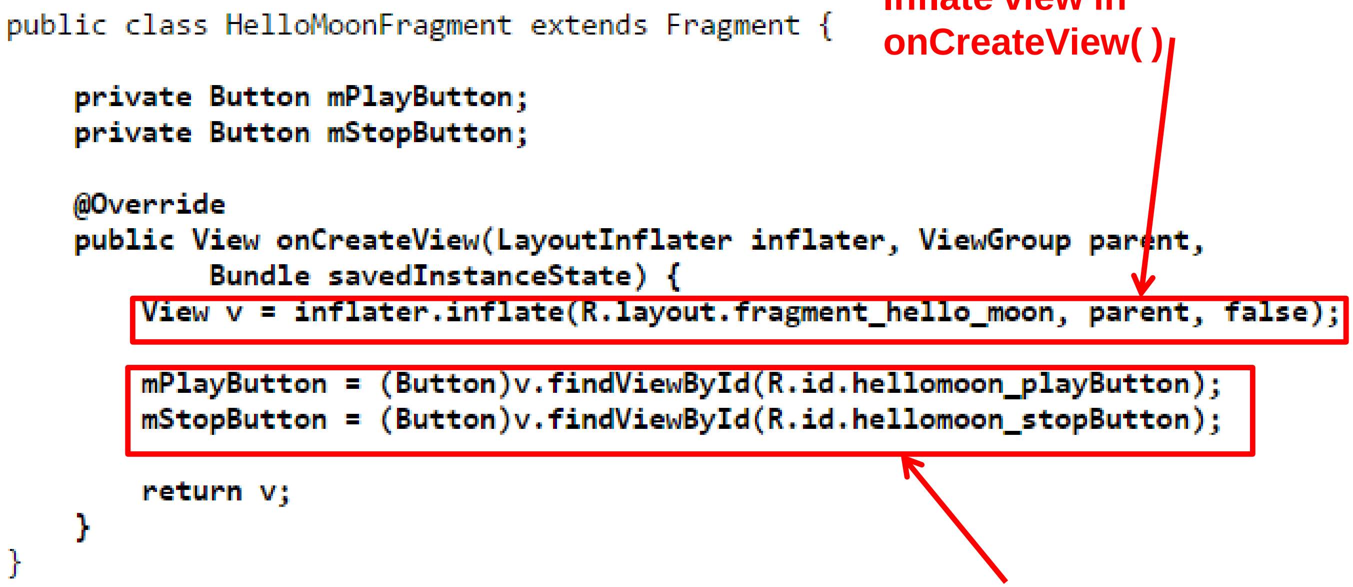
- Previously added Fragments to activity's java code
- **Layout fragment:** Can also add fragments to hosting Activity's XML file
- We will use a layout fragment instead
- Create activity's XML layout (**activity_hello_moon.xml**)
- **Activity's** XML layout file contains/hosts fragment



```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/helloMoonFragment"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:name="com.bignerdranch.android.hellomoon.HelloMoonFragment">

</fragment>
```



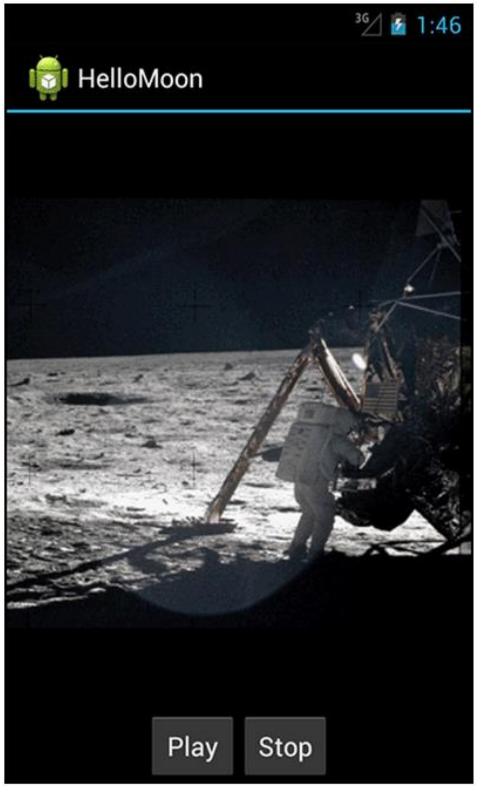
Set up HelloMoonFragment.java



```
public class HelloMoonFragment extends Fragment {

    private Button mPlayButton;
    private Button mStopButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_hello_moon, parent, false);

        mPlayButton = (Button)v.findViewById(R.id.hellomoon_playButton);
        mStopButton = (Button)v.findViewById(R.id.hellomoon_stopButton);

        return v;
    }
}
```

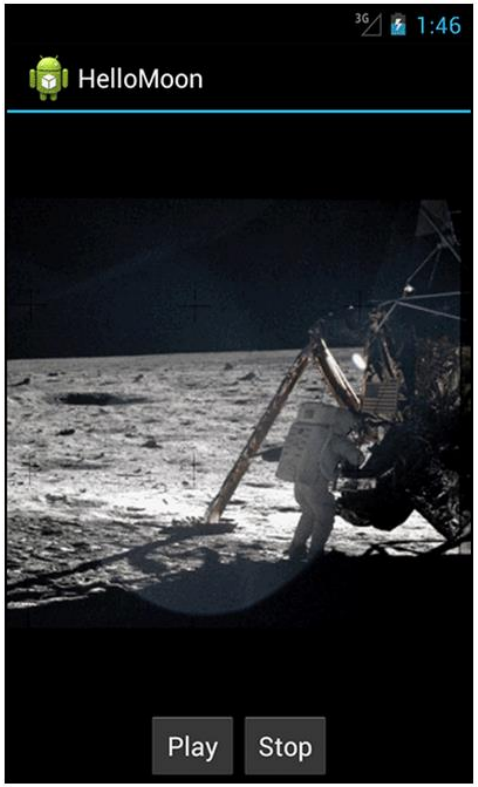
Inflate view in onCreateView( )

Get handle to Start, Stop buttons



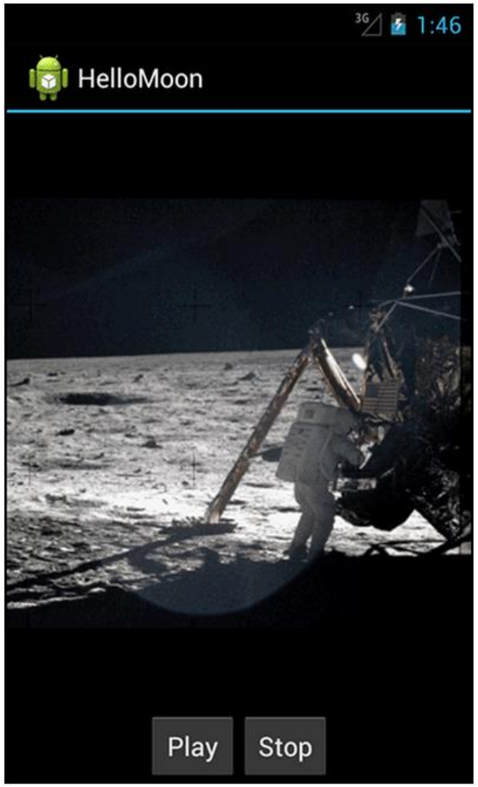
Create AudioPlayer Class encapsulates MediaPlayer

```
public class AudioPlayer {

    private MediaPlayer mPlayer;

    public void stop() {
        if (mPlayer != null) {
            mPlayer.release();
            mPlayer = null;
        }
    }

    public void play(Context c) {
        mPlayer = MediaPlayer.create(c, R.raw.one_small_step);
        mPlayer.start();
    }
}
```



Hook up Play and Stop Buttons

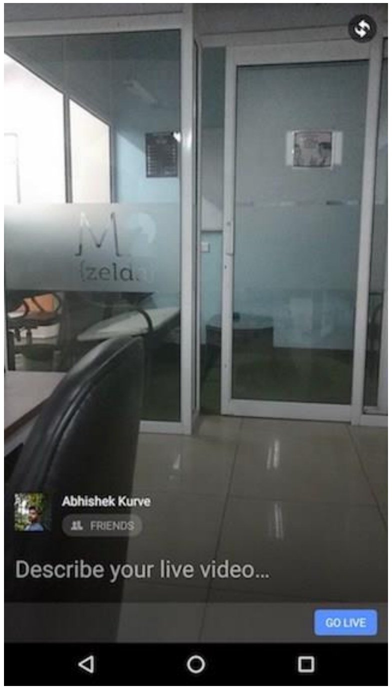
```
public class HelloMoonFragment extends Fragment {
    private AudioPlayer mPlayer = new AudioPlayer();
    private Button mPlayButton;
    private Button mStopButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_hello_moon, parent, false);

        mPlayButton = (Button)v.findViewById(R.id.hellomoon_playButton);
        mPlayButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mPlayer.play(getActivity());
            }
        });

        mStopButton = (Button)v.findViewById(R.id.hellomoon_stopButton);
        mStopButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mPlayer.stop();
            }
        });
        return v;
    }
```



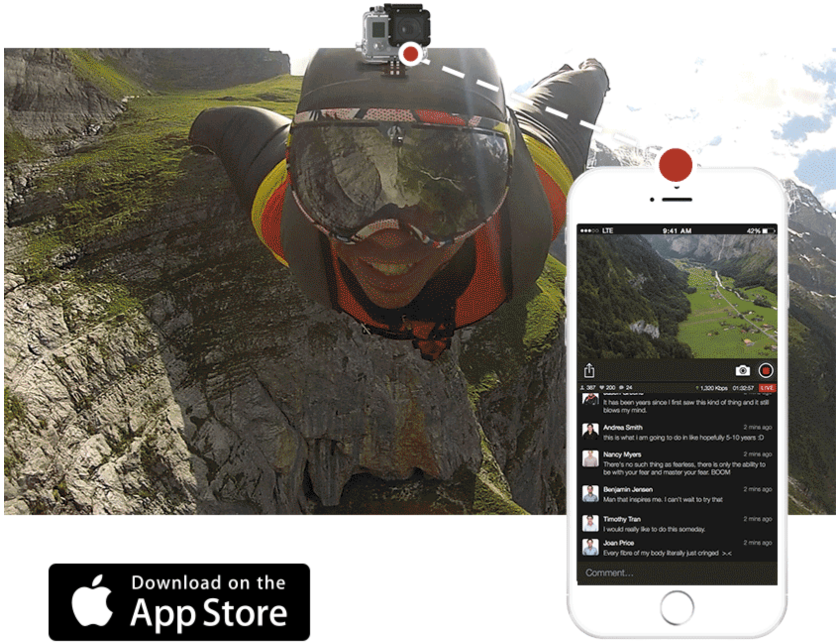
Live Streaming

Live Streaming

- Live streaming extremely popular now (E.g. going Live on Facebook)
- A person can share their experiences with friends
- Popular **live streaming apps** include Facebook, Periscope
- Also possible on **devices** such as Go Pro
- Uses RTMP (real time protocol by Adobe), supported by many 3rd party APIs



Facebook Live



Download on the App Store

Live GoPro

Live Streaming Bandwidth Issues

- WiFi bandwidth adequate, high quality video possible
- Cellular links:
  - Low bandwidth,
  - Variable (multi-path fading) even when standing still
  - Optimized for download not upload
- Video quality increasing faster than cellular bandwidths
  - Ultra HD, 4k cameras makes it worse, now available on many smartphones

Live Streaming

P Lundrigan *et al*, Mobile Live Video Upstreaming, International Teletraffic Congress, 2016

- **Scenario:** Multiple smartphones in same area
- **Approach: Live upstreaming of video using neighbors:**
  - Cell protocol guarantees each smartphone slice of cell bandwidth
  - Use/Combine neighbors bandwidth to improve video quality
  - Streaming smartphone: WiFi Direct connection to neighbors
  - WiFi Direct allows smartphones connect directly, no AP

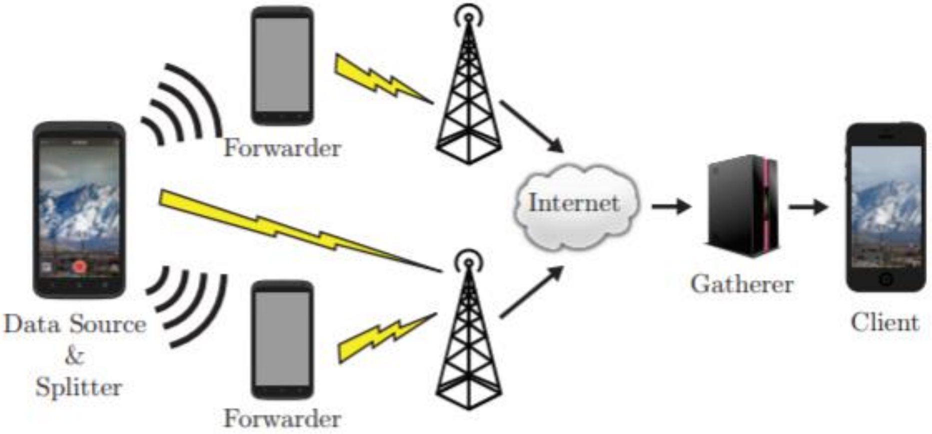
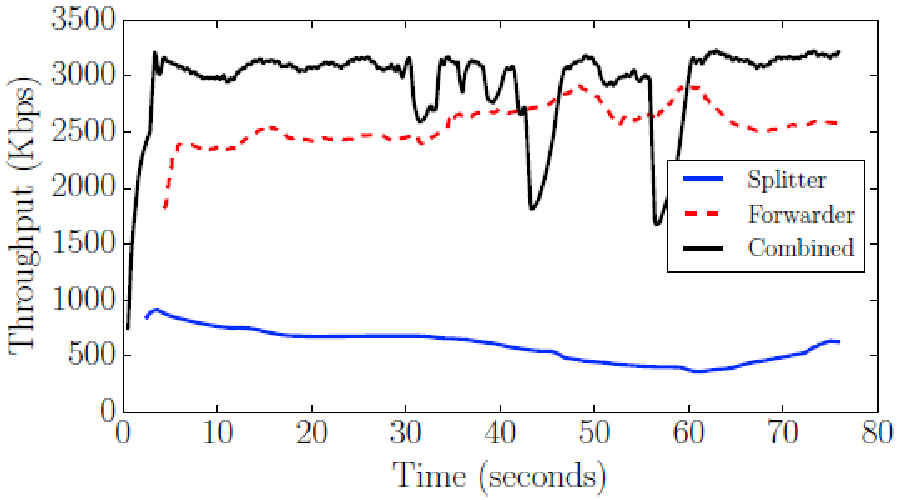


Fig. 1. General architecture of mobiLivUp. Data passes from the splitter to forwarders, then to the gatherer through their cellular connections.

Live Streaming

P Lundrigan *et al*, Mobile Live Video Upstreaming, International Teletraffic Congress, 2016

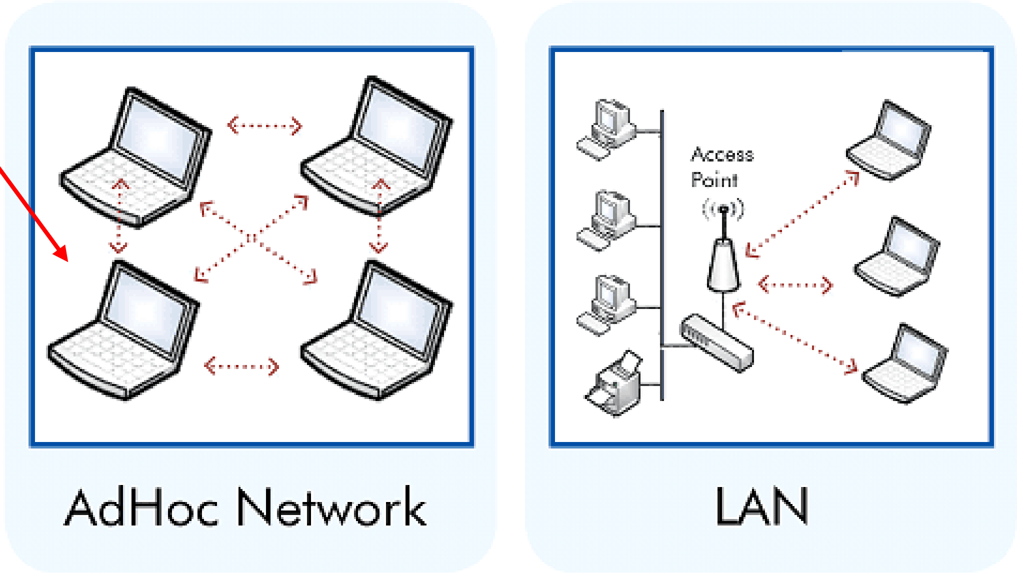
- **Results:** 2 smartphones 88% throughput increase vs 1 phone



- **Issues:**
  - Video packets travel/arrive out of order
  - Incentives for forwarding nodes?

Ad Hoc Vs Infrastructure WiFi Mode

- **Infrastructure mode:** Mobile devices communicate through Access point
- **Ad Hoc Mode:** Mobile devices communicate directly to each other (no AP required)
- **WiFi Direct** is new standard to be used for ad hoc WiFi mode



References

- Head First Android
- Android Nerd Ranch, 2nd edition
- Busy Coder's guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014