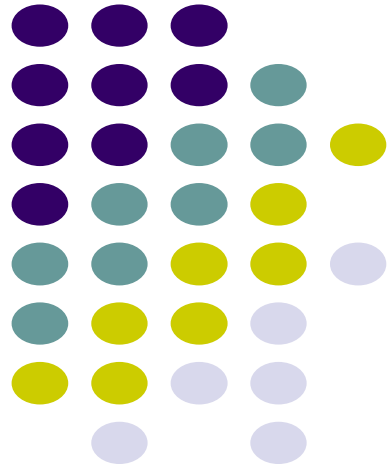
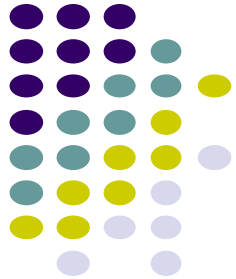
CS 528 Mobile and Ubiquitous Computing

Lecture 4b: Multimedia: Camera, Audio, Video and Sound

Emmanuel Agu

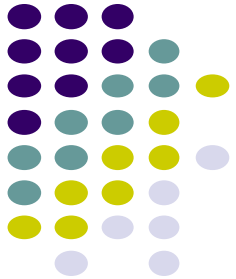




Android Nerd Ranch Ch 14
SQLite Databases

Database Support on Android

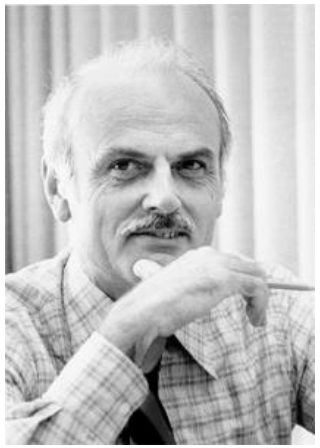
Ref: https://greenrobot.org/news/mobile-databases-sqlite-alternatives-and-nosql-for-android-and-ios/



- Mobile database definition:
  - Stationary database, mobile device can connect
  - Database stored on mobile device
- Stores structured information with defined fields
  - E.g. Name, height, weight, etc
- Mobile/ubicomp uses:
  - Smartphones, smartwatches, game consoles, IoT/home appliances, robots
- Why use mobile database?
  - Can work offline
  - Pre-retrieve all data at once => lower bandwidth requirements
  - Privacy: store user's personal information locally

Background on Databases

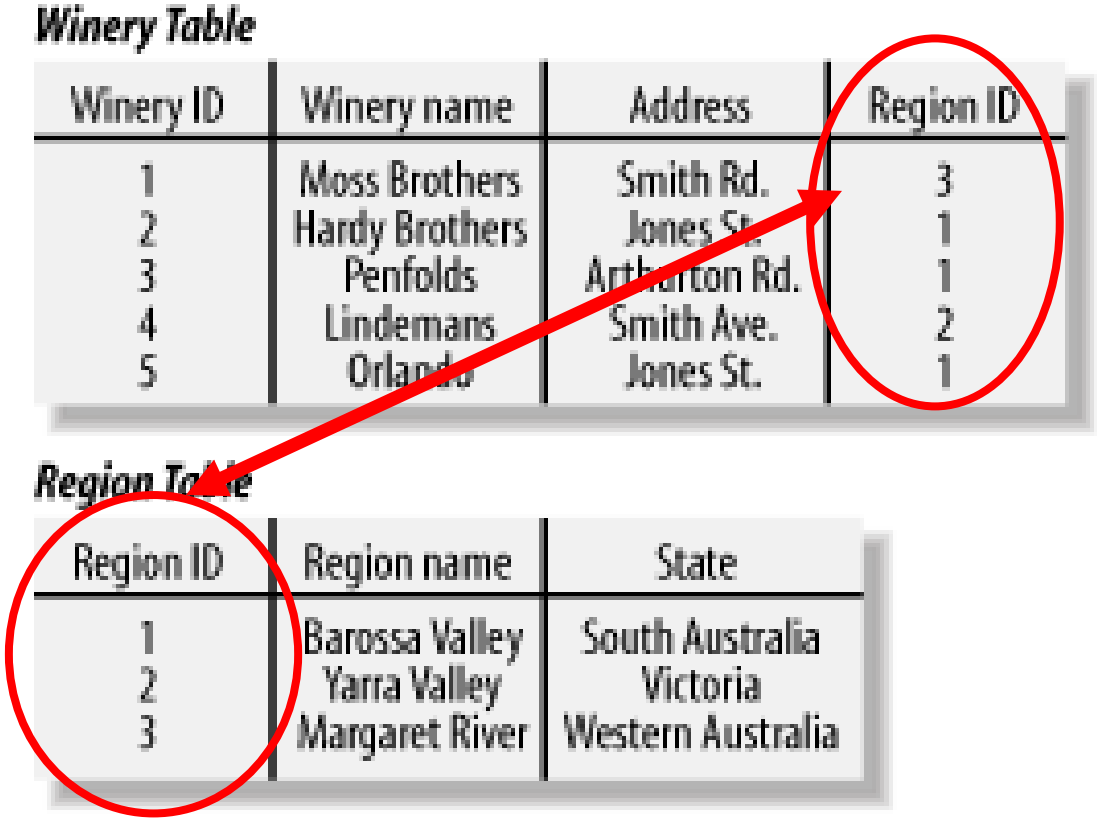
- **Note:** Google now have new database API (Room)
  - But we will use SQLite here, low-level, book uses it
- Relational DataBase Management System (RDBMS)
  - Introduced by E. F. Codd (Turing Award Winner)



- Relational Database
  - data stored in tables
  - relationships among data stored in tables
  - data can be accessed and viewed in various ways

Example Wines Database

- **Relational Data:** Data in different tables can be related



Winery Table

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

Ref: Web Database Applications with PHP and MySQL, 2nd Edition , by Hugh E. Williams, David Lane

Keys

- Each table has a key
- **Key:** column used to uniquely identify each row

KEYS

Winery Table

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

SQL and Databases

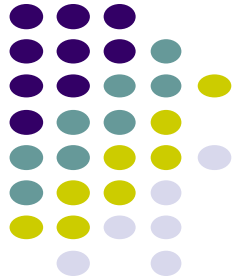
- **SQL:** language used to manipulate Relational Database (RDBMS)

- SQL Commands:
  - **CREATE TABLE** - creates new database table
  - **ALTER TABLE** - alters a database table
  - **DROP TABLE** - deletes a database table
  - **SELECT** - get data from a database table
  - **UPDATE** - change data in a database table
  - **DELETE** - remove data from a database table
  - **INSERT INTO** - insert new data in a database table

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

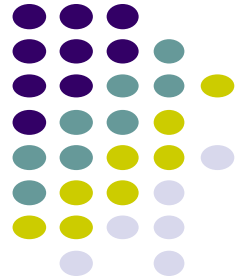
CriminalIntent Database



- **SQLite:** open source relational database
- SQLite implements subset of SQL (most but not all)
  - http://www.sqlite.org/
- Android includes a SQLite database
- **New:** Android higher level database library called Room
  - Allows easy creation and manipulation of SQLite databases
- **Goal:** Store crimes in CriminalIntent in SQLite database
- First step, define database table of **crimes**

_id	uuid	title	date	solved
1	13090636733242	Stolen yogurt	13090636733242	0
2	13090732131909	Dirty sink	13090732131909	1

CriminalIntent Database Schema



- Create **CrimeDbSchema** class to store **crime** database
- Define fields/columns of the Crimes database table

```
public class CrimeDbSchema {
    public static final class CrimeTable {
        public static final String NAME = "crimes";

        public static final class Cols {
            public static final String UUID = "uuid";
            public static final String TITLE = "title";
            public static final String DATE = "date";
            public static final String SOLVED = "solved";
        }
    }
}
```

Name of Table

Each Crimes Table has the following fields/columns

_id	uuid	title	date	solved
1	13090636733242	Stolen yogurt	13090636733242	0
2	13090732131909	Dirty sink	13090732131909	1

Crimes Table

SQLiteOpenHelper

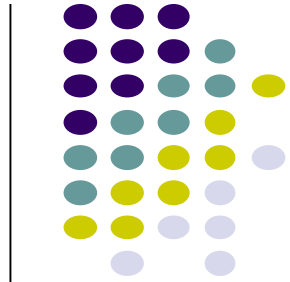
- **SQLiteOpenHelper** class used for database creation, opening and updating a **SQLiteDatabase**
- In **CriminalIntent**, create subclass of **SQLiteOpenHelper** called **CrimeBaseHelper**

```
public class CrimeBaseHelper extends SQLiteOpenHelper {
    private static final int VERSION = 1;
    private static final String DATABASE_NAME = "crimeBase.db";

    public CrimeBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, VERSION);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Used to create the database (to store Crimes)

Called the first time database is created

Use CrimeBaseHelper to open SQLite Database



```
public class CrimeLab {
    private static CrimeLab sCrimeLab;

    private List<Crime> mCrimes;
    private Context mContext;
    private SQLiteDatabase mDatabase;
    ...
    private CrimeLab(Context context) {
      mContext = context.getApplicationContext();
      mDatabase = new CrimeBaseHelper(mContext)
              .getWritableDatabase();
      mCrimes = new ArrayList<>();
    }
```

Open new writeable Database

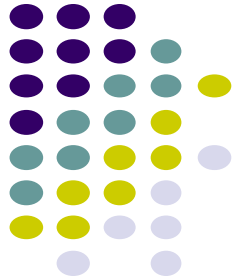
Create CrimeTable in onCreate( )

onCreate called first time
database is created

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + CrimeTable.NAME + "(" +
            " _id integer primary key autoincrement, " +
            CrimeTable.Cols.UUID + ", " +
            CrimeTable.Cols.TITLE + ", " +
            CrimeTable.Cols.DATE + ", " +
            CrimeTable.Cols.SOLVED +
            ")"
    );
}
```

Create CrimeTable in our new
Crimes Database

Writing Crimes to Database using ContentValues



- In Android, writing to databases is done using class **ContentValues**
- **ContentValues** is key-value pair
- Create method to create **ContentValues** instance from a **Crime**

```
    public Crime getCrime(UUID id) {
        return null;
    }

    private static ContentValues getContentValues(Crime crime) {
        ContentValues values = new ContentValues();
        values.put(CrimeTable.Cols.UUID, crime.getId().toString());
        values.put(CrimeTable.Cols.TITLE, crime.getTitle());
        values.put(CrimeTable.Cols.DATE, crime.getDate().getTime());
        values.put(CrimeTable.Cols.SOLVED, crime.isSolved() ? 1 : 0);

        return values;
    }
}
```

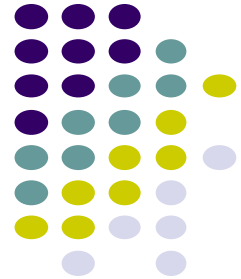
Takes Crime as input

key

value

Converts Crime to ContentValues

Returns values as output



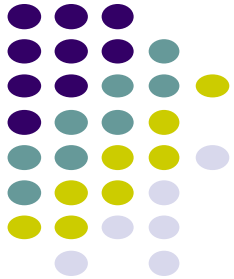
Firebase Cloud API

Firebase

- Mobile cloud backend service for
  - Analytics
  - Messaging
  - Authentication
  - Database
  - Crash reporting, etc

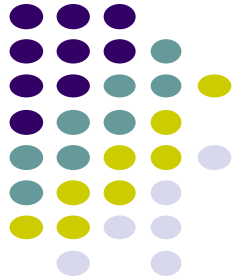
- Previously 3rd party company
- Acquired by Google in 2014
  - Now part of Google. See https://firebase.google.com/
  - Fully integrated, could speed up development. E.g. final project

Firebase

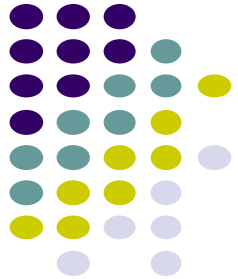


- Relatively easy programming, few lines of code
- E.g. to create database

```
FirebaseDatabase database = FirebaseDatabase.getInstance()
// write
database.child("users").child("userId").setValue(user);

// read / listen
database.child("users").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // ...
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
```



Multimedia Networking: Basic Concepts



Multimedia networking: 3 application types

- Multimedia refers to audio and video. 3 types

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))

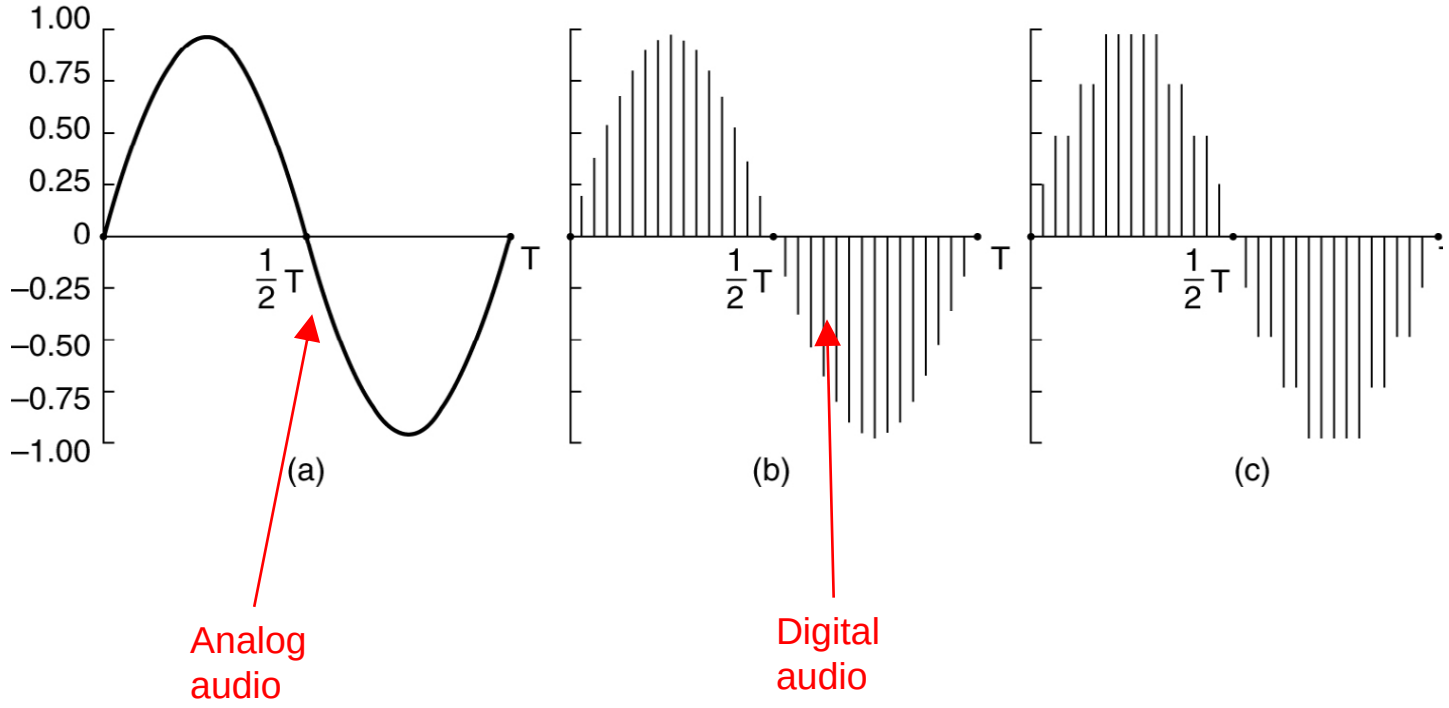
2. *streaming live* audio, video
   - e.g., live sporting event (futbol)

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

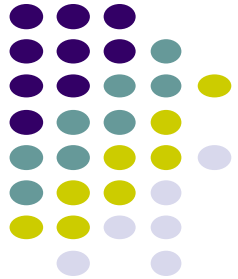
Credit: Computer Networks (6th edition),
By Kurose and Ross

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



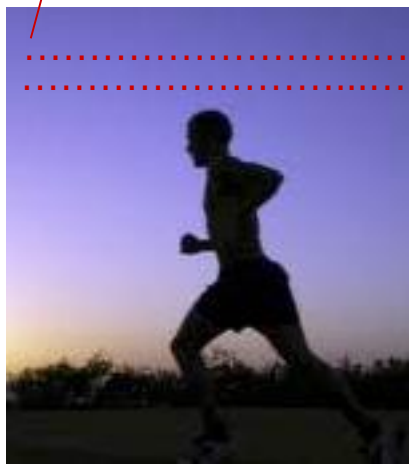
- Audio CDs:
  - 44,100 samples/second
  - Uncompressed audio, requires 1.4Mbps to transmit real-time

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

Video Encoding

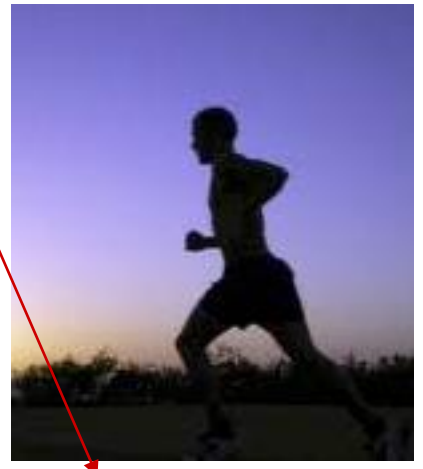
- **Digital image:** array of <R,G,B> pixels
- **Video:** sequence of images
- **Redundancy:** Consecutive frames mostly same (1/30 secs apart)
- **Video coding (e.g. MPEG):** use redundancy *within* and *between* images to decrease # bits used to encode video
  - **Spatial** (within image)
  - **Temporal**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

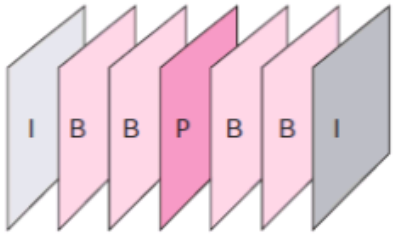
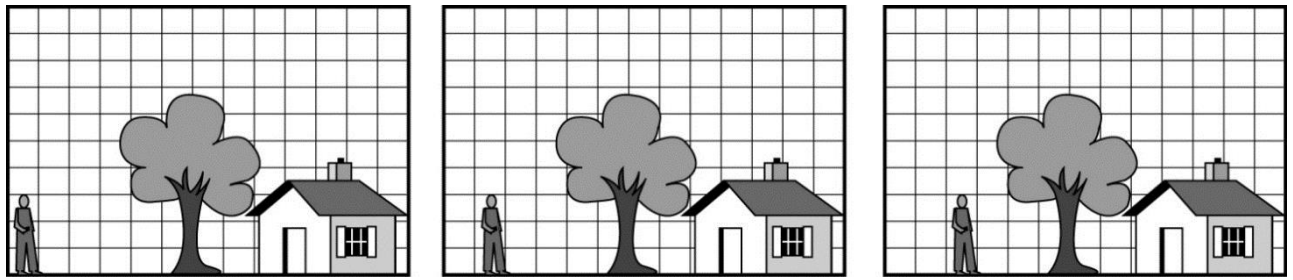


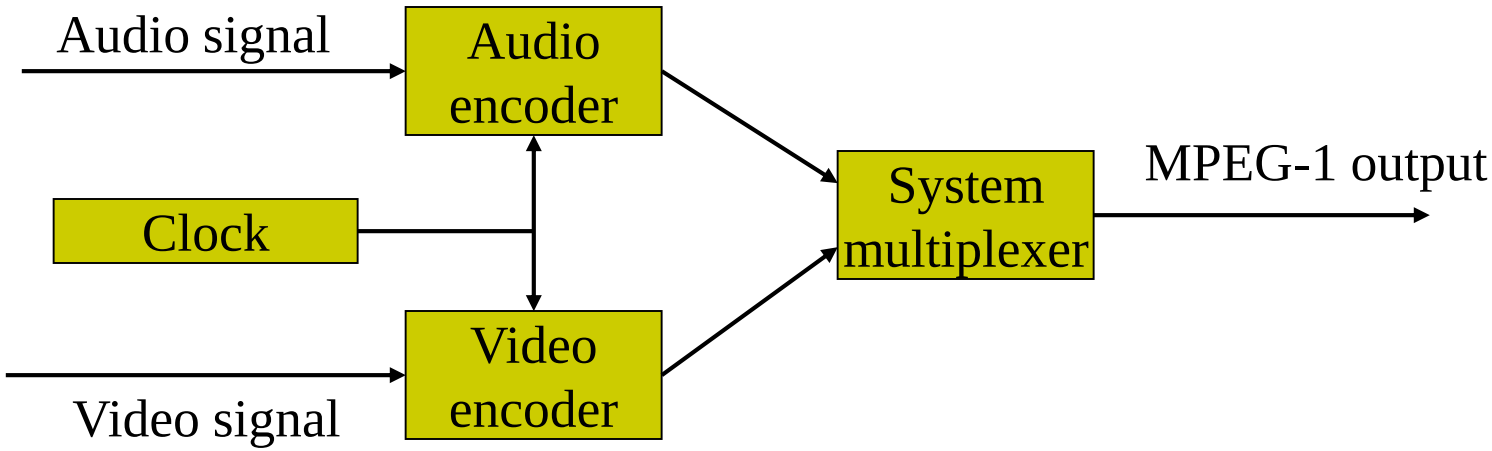
Fig1: MPEG frames



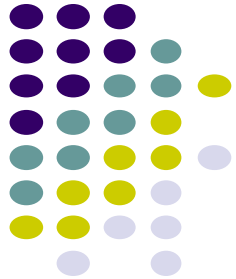
3 consecutive frames

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



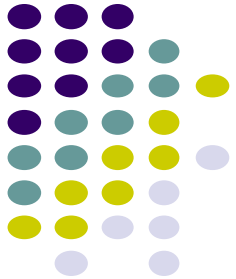
- Sample MPEG rates:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)



Playing Audio and Video in Android

MediaPlayer

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



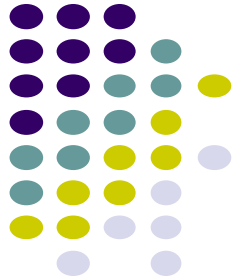
- Android Classes used to play sound and video
  - **MediaPlayer:** Plays sound and video
  - **AudioManager:** plays only audio

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

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

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
  1. **User interface**, user interaction
  2. Handle **Transmission errors**: retransmissions, interleaving
  3. **Decompress** audio
  4. **Eliminate jitter:** Playback buffer (Pre-download 10-15 secs of music)

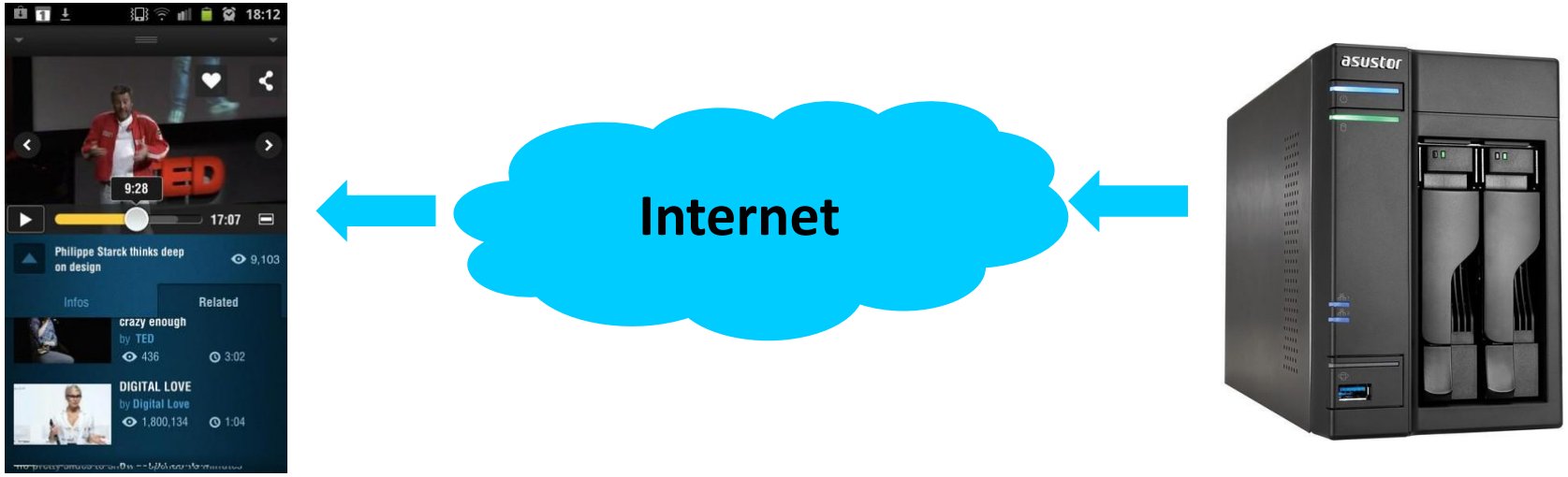
Using Media Player:

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:

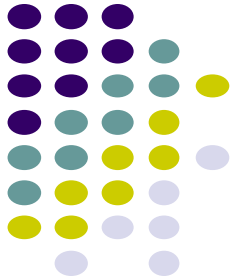
```
<uses-permission android:name="android.permission.INTERNET" />
```



- 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



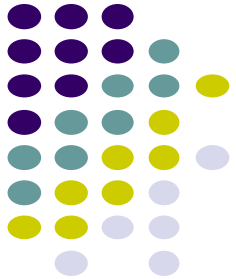
- To play audio file saved in app's **res/raw/** directory

```
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

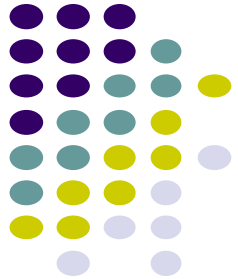
Step 2: Create MediaPlayer Object, Start Player



- 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;
```

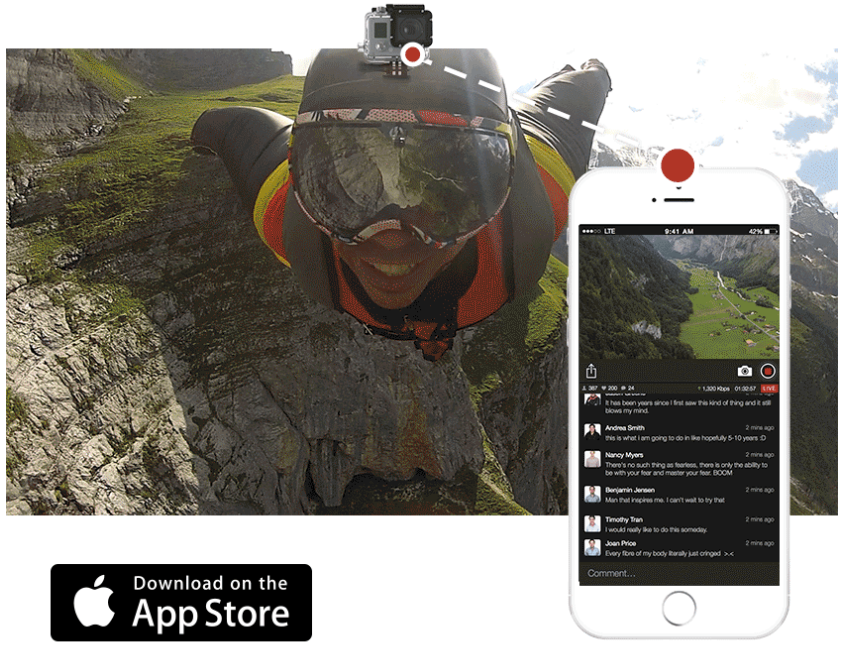
- **MediaPlayer in a Service:** Can play media (e.g. music) in background while app is not running
  - Start MediaPlayer as service

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), or other 3rd party APIs

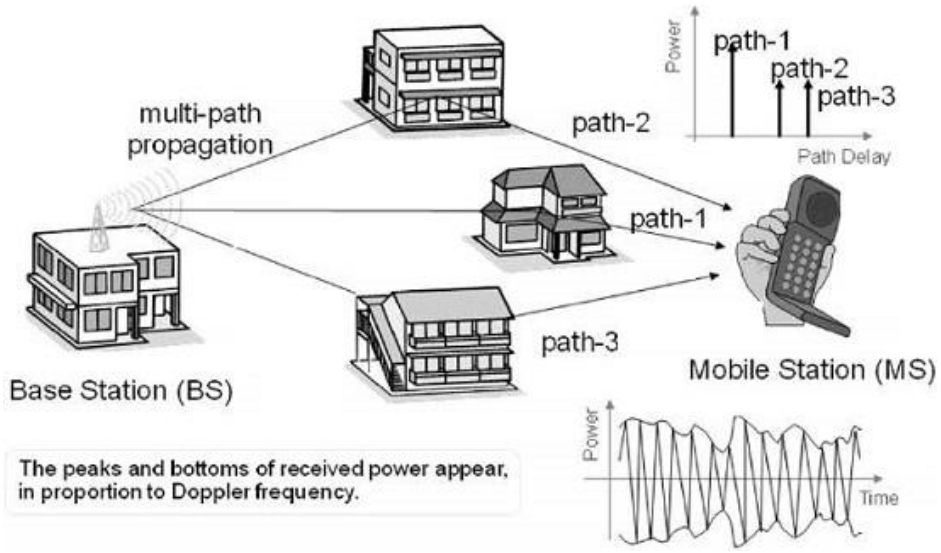
Facebook Live



Live GoPro

Live Streaming Bandwidth Issues

- On WiFi, bandwidth is adequate, high quality video possible
- Cellular links:
  - Low bandwidth,
  - Variable bandwidth (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



mobiLivUp Live Streaming

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

- **Scenario:** Multiple smartphones in same area
- **mobiLivUp approach: Live video upstreaming 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 Access Point

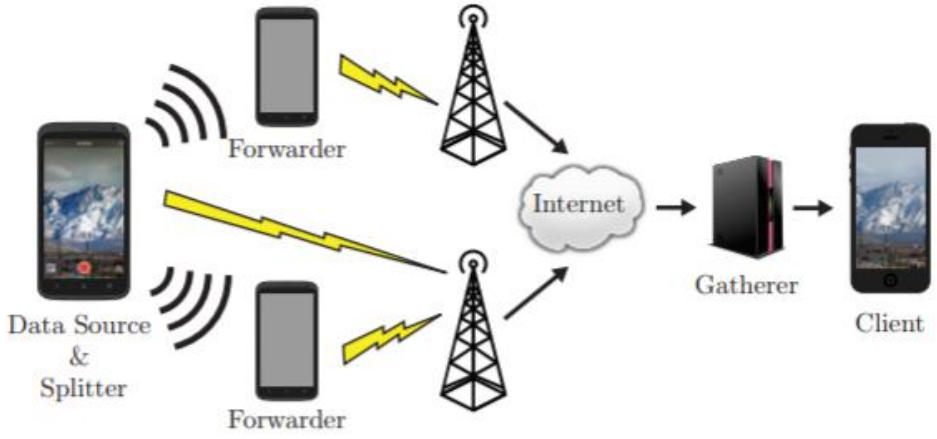
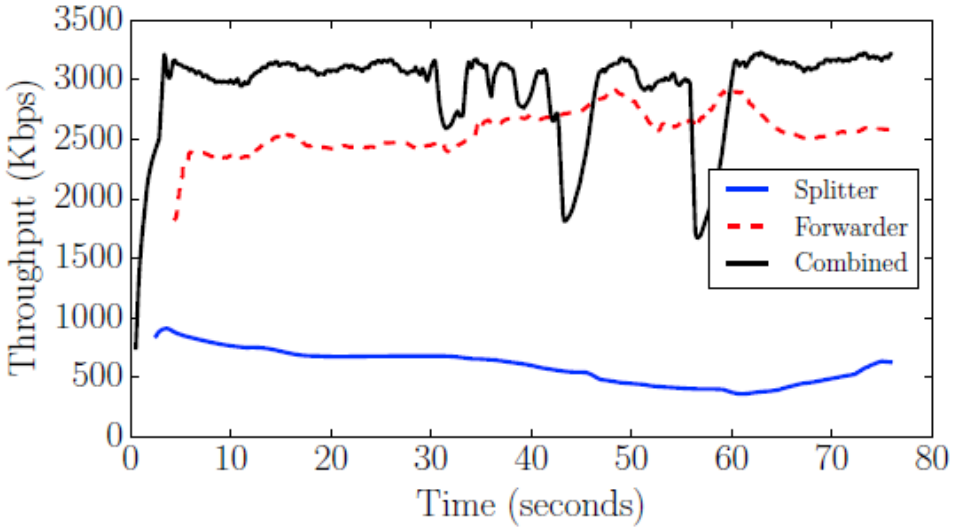


Fig. 1. General architecture of mobiLivUp. Data passes from the splitter to forwarders, then to the gatherer through their cellular connections.

Live Streaming

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

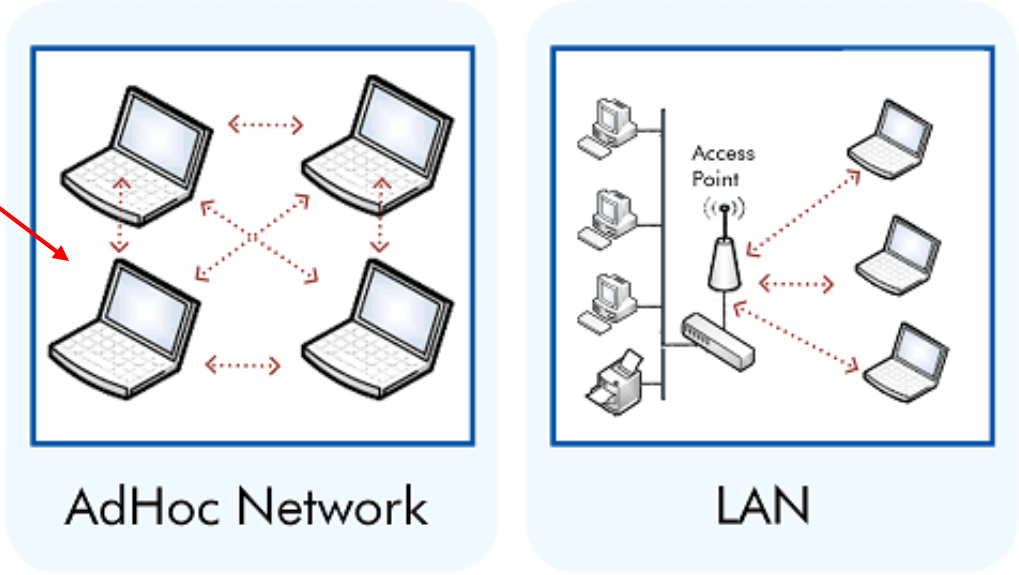
- **Results:** 2 smartphones 88% throughput increase vs 1 phone

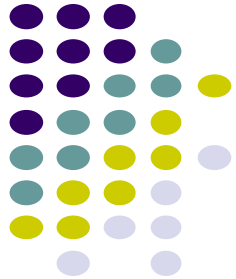


- **Issues:**
  - Video packets travel/arrive out of order
  - Incentives for forwarding nodes?

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



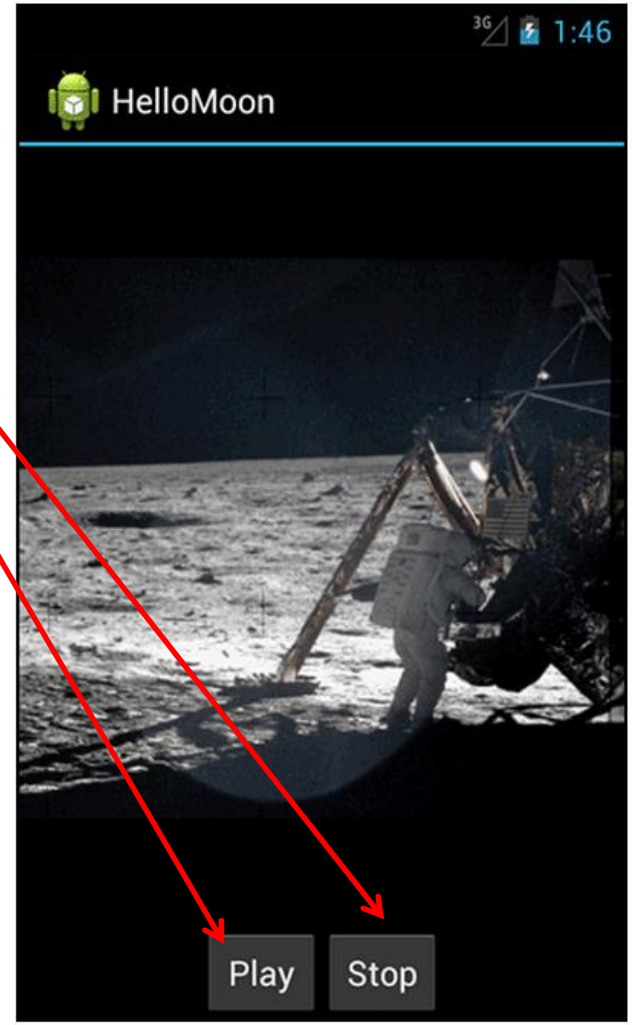


Playing Audio File using MediaPlayer
Example from Android Nerd Ranch 1st edition

MediaPlayer Example to Playback Audio

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

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

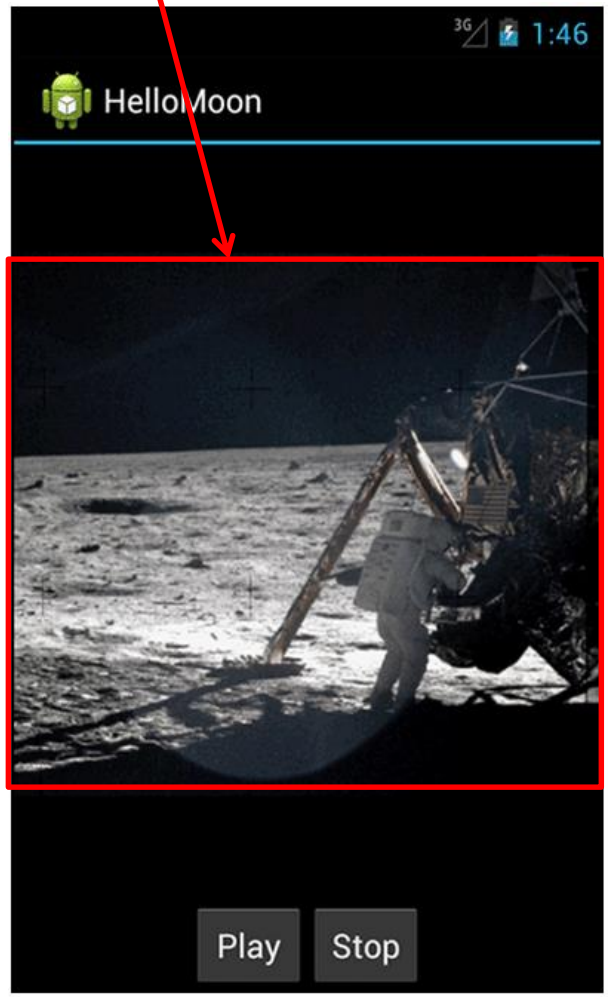


HelloMoon App

- 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
  - Play, Stop, Image description..

```
<?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>
```

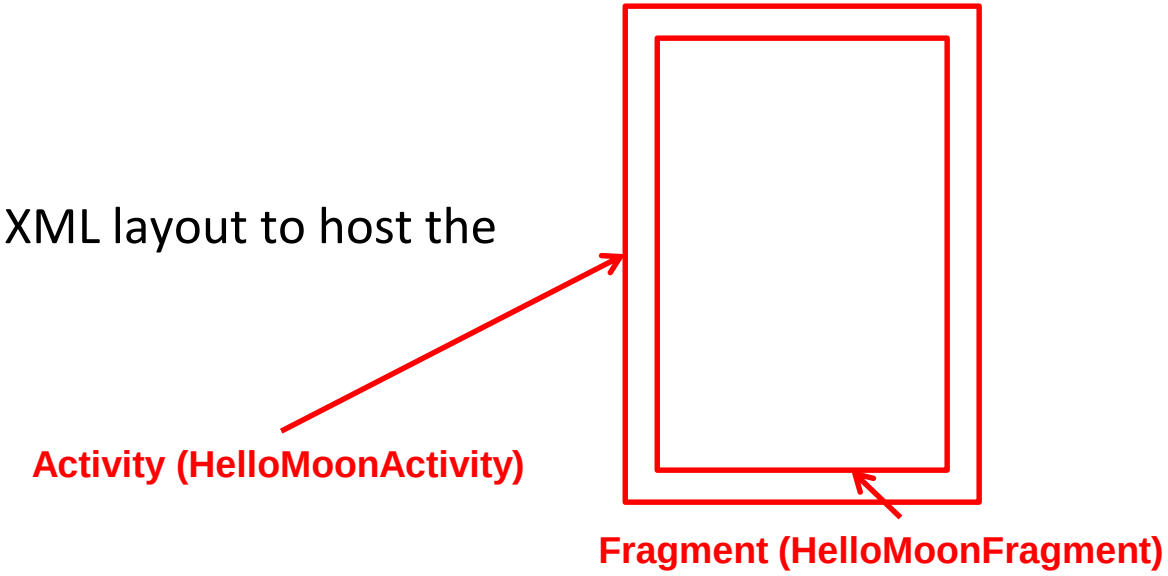
armstrong_on_moon.jpg



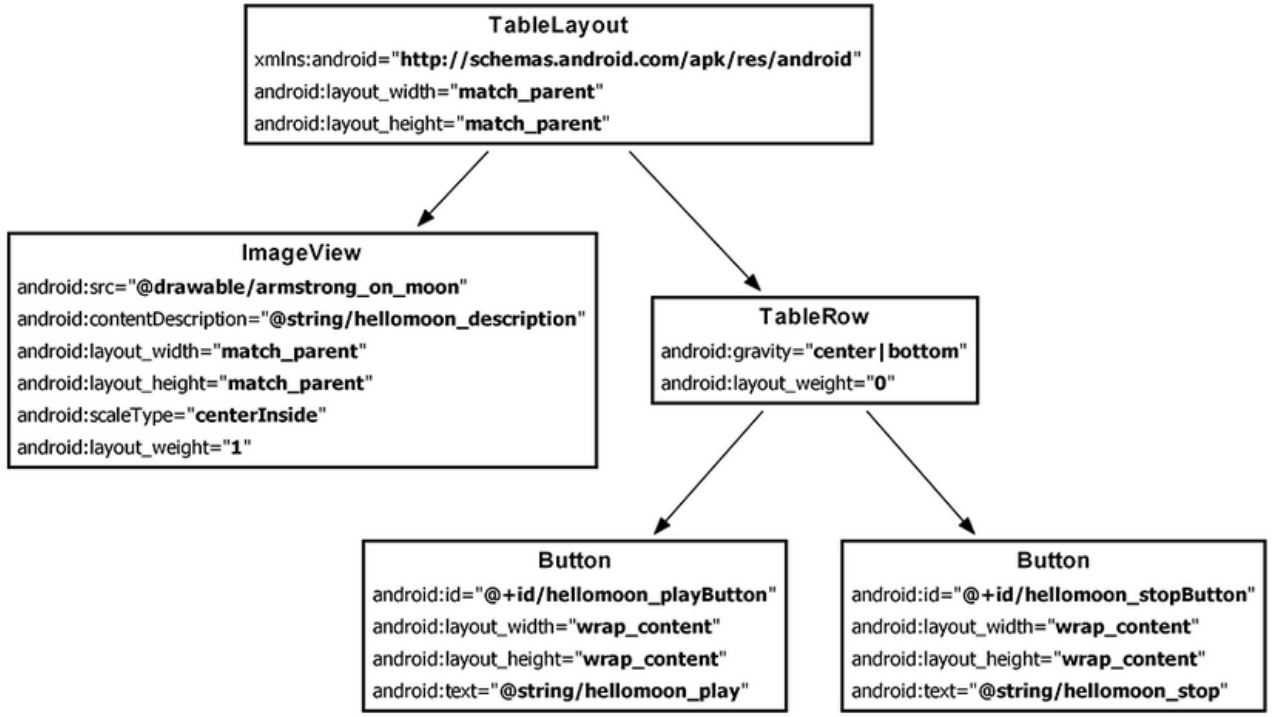
HelloMoon App

- 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



Defining the Layout for HelloMoonFragment



Define XML for HelloMoon UI (fragment_hello_moon.xml)

Creating a Layout Fragment

- **Layout fragment:** Add fragments to hosting Activity's XML file
- 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>
```

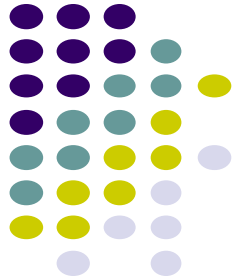
Set up HelloMoonFragment.java

```
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

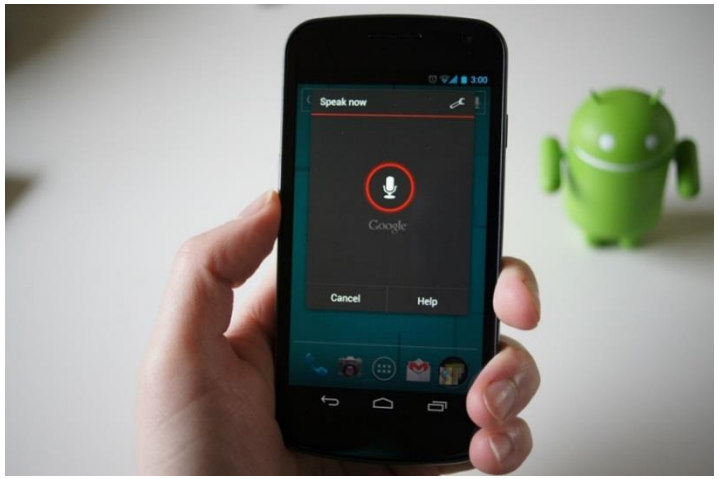
```
public class HelloMoonFragment extends Fragment {
    private AudioPlayer mPlayer = new AudioPlayer();
    private Button mPlayButton;
    private Button mStopButton;

    @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;
    }
```

Speech: Android Support

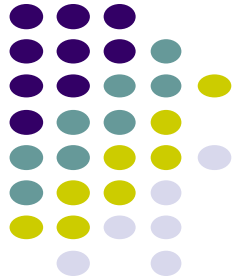
Speaking to Android

http://developer.android.com/reference/android/speech/SpeechRecognizer.html
https://developers.google.com/voice-actions/

- **Speech recognition:**
  - Accept inputs as speech (instead of typing) e.g. dragon dictate app?
  - Note: Requires internet access
- Two forms
  1. **Speech-to-text**
     - Convert user's speech to text. E.g. display voicemails in text
  2. **Voice Actions:** Voice commands to smartphone (e.g. search for, order pizza)

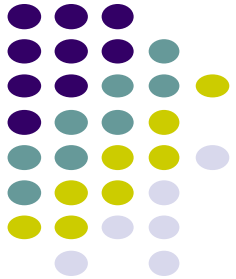


Speech to text



Quiz

Quiz Next Class



- Quiz 2 next class: Sept. 23, first 20 minutes of class
- Covers:
  - Lectures 3 and 4
  - Code assigned with those classes

References

- Google Camera “Taking Photos Simply” Tutorials, http://developer.android.com/training/camera/photobasics.html
- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014
- Android Nerd Ranch, 1st edition