CS 528: Mobile and Ubiquitous Computing

Lecture 6a: Multimedia Networking

Emmanuel Agu

Announcements

- Final Project 1 slide deadline now moved to:
  - Tue, Oct 24 at 10am
  - **Note:** Project 2 still due next Thursday (Oct 12)

- Quiz 3:
  - Beginning of next class (Oct. 12)
  - Covers slides from lectures 5 and 6 (today)

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, playout starts before downloading entire file
   - e.g., YouTube, Netflix, Hulu
   - Streaming Protocol used (e.g. Real Time Streaming Protocol (RTSP), HTTP streaming protocol (DASH))

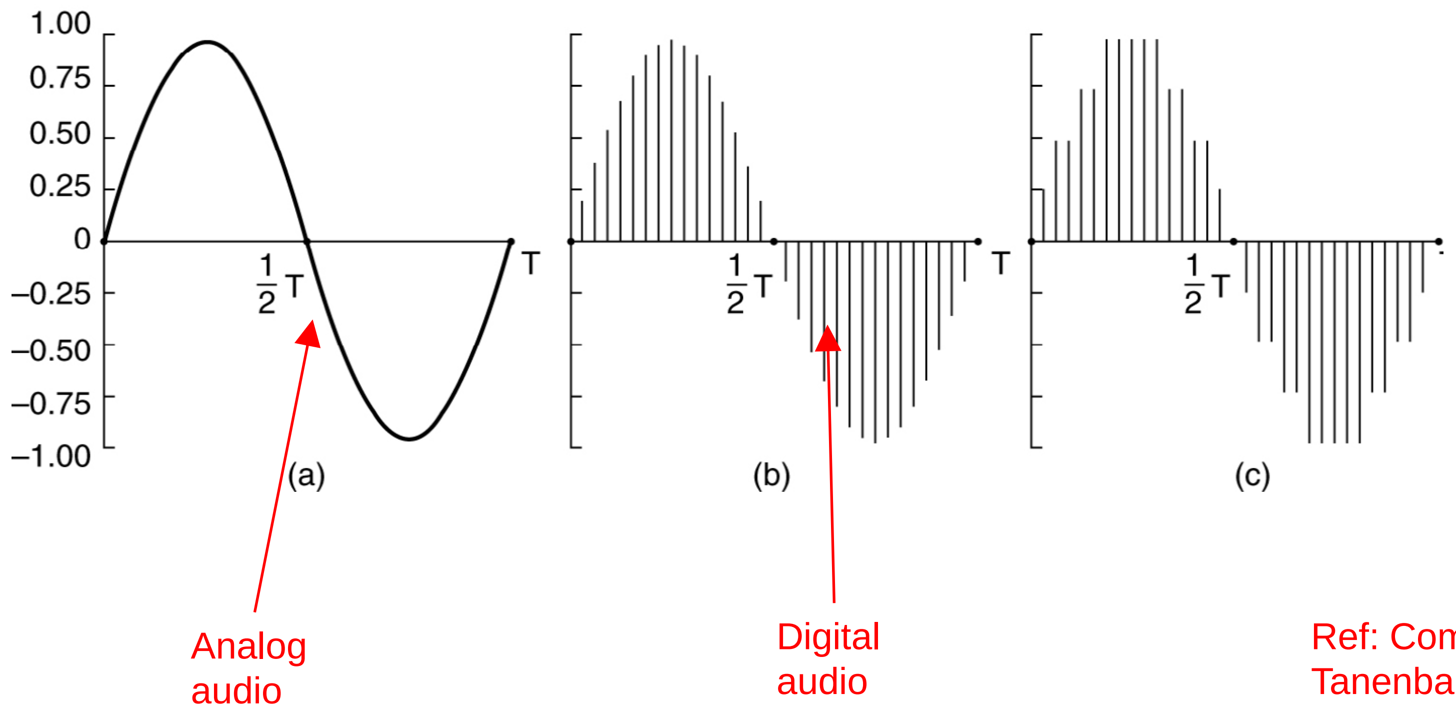
2. *streaming live* audio, video
   - e.g., live sporting event (e.g. watching football/soccer online)

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

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

Digital Audio

- Sender converts audio from analog waveform to digital signal
- E.g PCM: 8-bit samples 8000 times per sec (8000 Hertz)
- Digital signal is transmitted
- Receiver converts digital signal back into audio analog waveform



Ref: Computer networks (6th edition)
Tanenbaum, Feamster and Wetherall

Audio Compression

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

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

Video: Sequence of Images (or Frames)

- Can think of video as sending sequence of images. E.g. every 1/30 of a second
- **Digital image:** array of <R,G,B> pixels
- **Video:** sequence of images.

Video Encoding

- **Redundancy:** Consecutive frames similar (1/30 secs apart)
- **Video coding (e.g. MPEG):** use redundancy *within* and *between* images to decrease # bits used to encode video
  - **Spatial redundancy:** within image
  - **Temporal redundancy:** 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*

frame *i+1*

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

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

MPEG: Spatial and Temporal Coding Example

- MPEG output consists of 3 kinds of frames:
  - **I (Intracoded)** frames:
    - JPEG-encoded still pictures (complete, 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

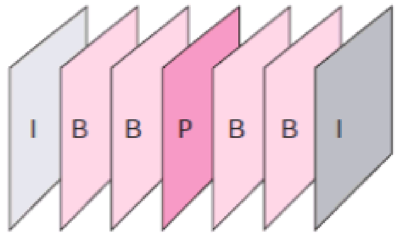
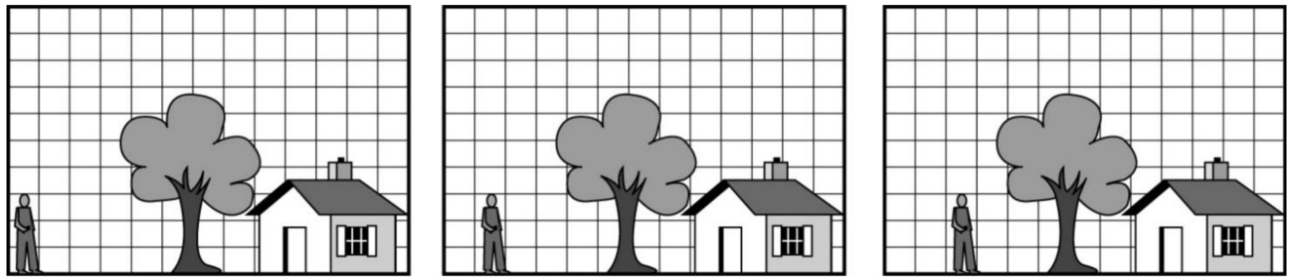
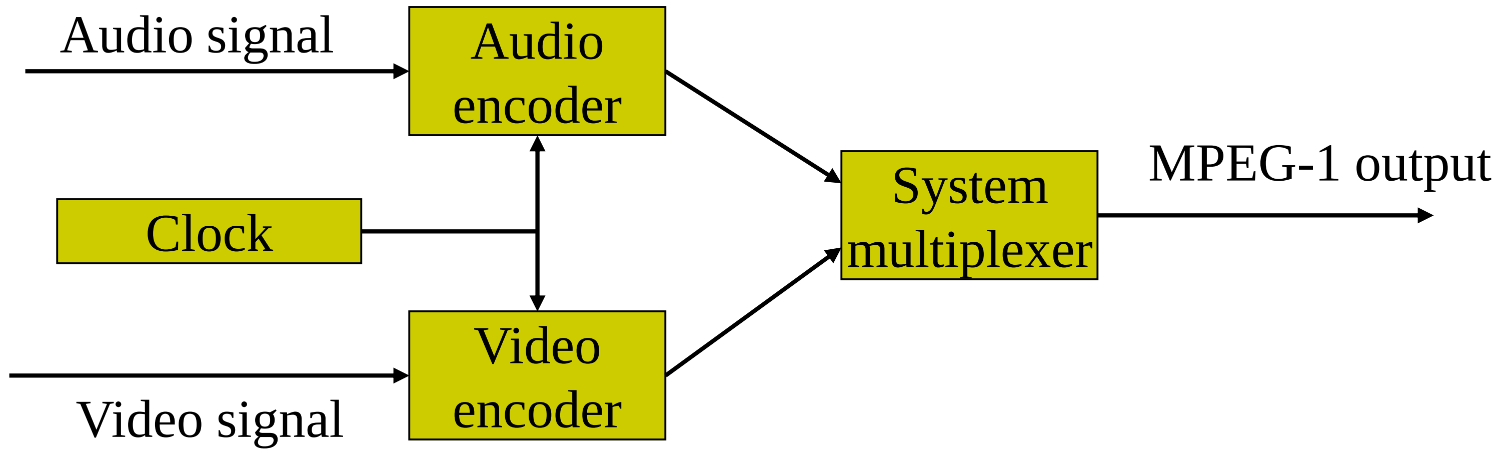


Fig1: MPEG frames


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
- If audio and video not synchronized, weird artifacts result



- Sample MPEG usage, 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

Player and UI

https://developer.android.com/guide/topics/media-apps/media-apps-overview

- 2 options to integrate multimedia functionality (audio or video) into an app
  - **MediaPlayer:**
    - Basic functionality
    - Supports most common audio/video formats and data sources
  - **ExoPlayer:**
    - Open source library
    - Supports high-performance features such as DASH
      - Dynamic Adaptive Streaming over HTTP (DASH), streaming protocol
      - Switches video stream between bit rates based on network performance, to keep video playing
    - Only available with Android 4.1 or higher

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 (E.g. user start/stop video playing)
  2. Handle **Transmission errors**: packet retransmissions, interleaving
  3. **Decompress** audio
  4. **Eliminate jitter:** Manage playback buffer (Pre-download 10-15 secs of music)

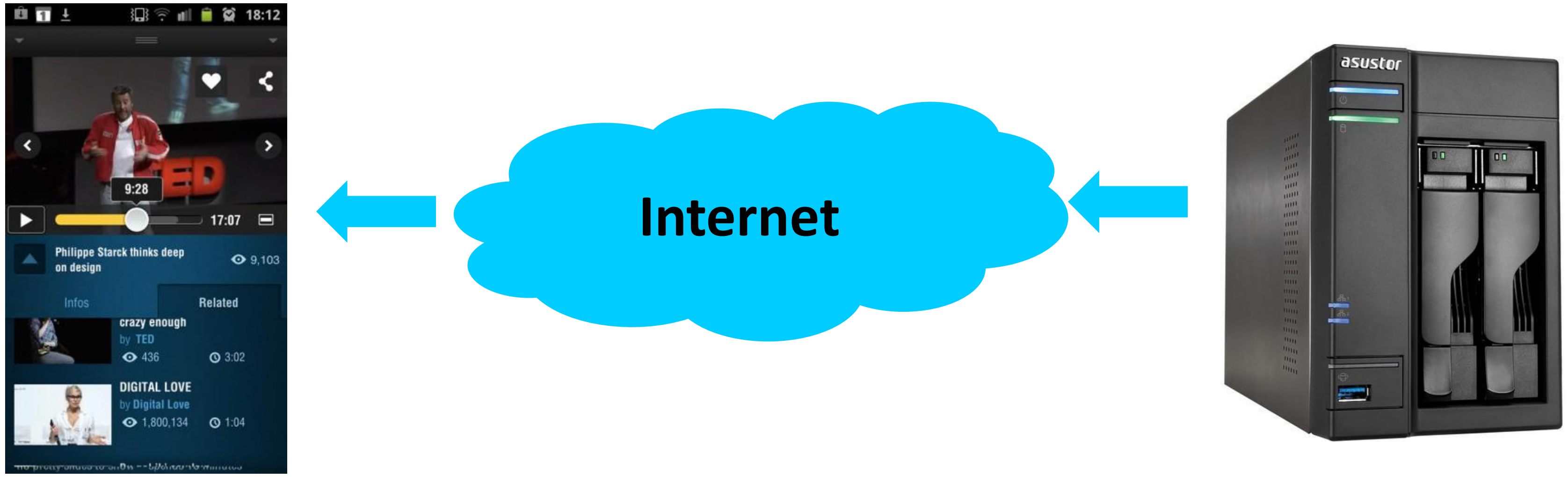
Using Media Player:

https://developer.android.com/guide/topics/media/mediaplayer

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 video file stored on user's smartphone, put video/audio files in **res/raw** folder

Using Media Player:

https://developer.android.com/guide/topics/media/mediaplayer

Step 1: Wake Lock Permission in AndroidManifest

- Android may dim screen to save power while video playing
- If player needs to keep screen from dimming or processor from sleeping, use:
  - **MediaPlayer.setScreenOnWhilePlaying( )**, or
  - **MediaPlayer.setWakeMode( )**
- For both commands, make Android Manifest request:

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

Using MediaPlayer

Step 2: Create MediaPlayer Object, Start Player

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

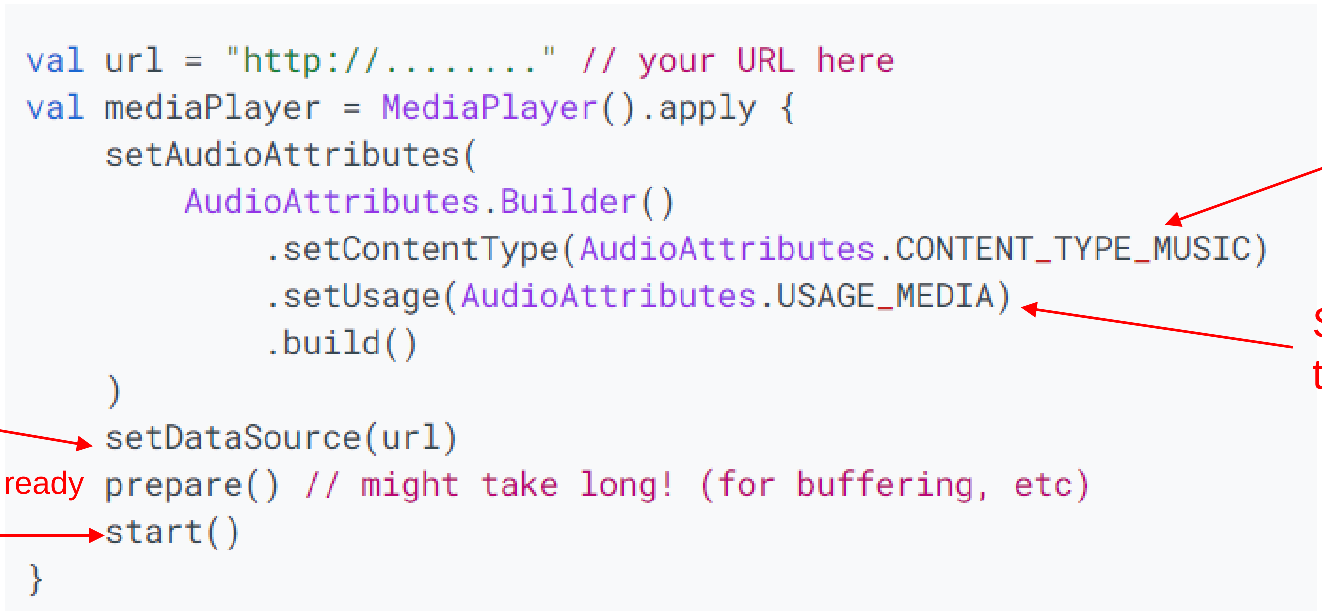
```
var mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1)
mediaPlayer.start() // no need to call prepare(); create() does that for you
```

- **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
https://developer.android.com/guide/topics/media/mediaplayer

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



```
val url = "http://........" // your URL here
val mediaPlayer = MediaPlayer().apply {
    setAudioAttributes(
        AudioAttributes.Builder()
            .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
            .setUsage(AudioAttributes.USAGE_MEDIA)
            .build()
    )
    setDataSource(url)
    prepare() // might take long! (for buffering, etc)
    start()
}
```

Sets attribute describing content type of audio signal. E.g. speech or music

Sets attribute describing intended use of the audio signal. E.g. alarm or ringtone

Set URL to get video

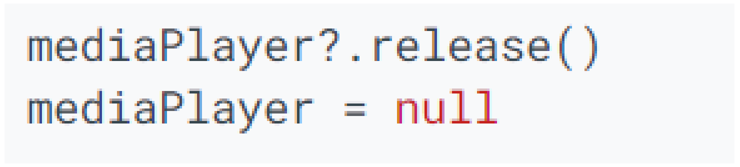
Blocks till MediaPlayer is ready

Start playing video

- Can also play audio from URI
    - E.g. foo://example.com:8042/over/there?name=ferret#nose

Releasing the MediaPlayer

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



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

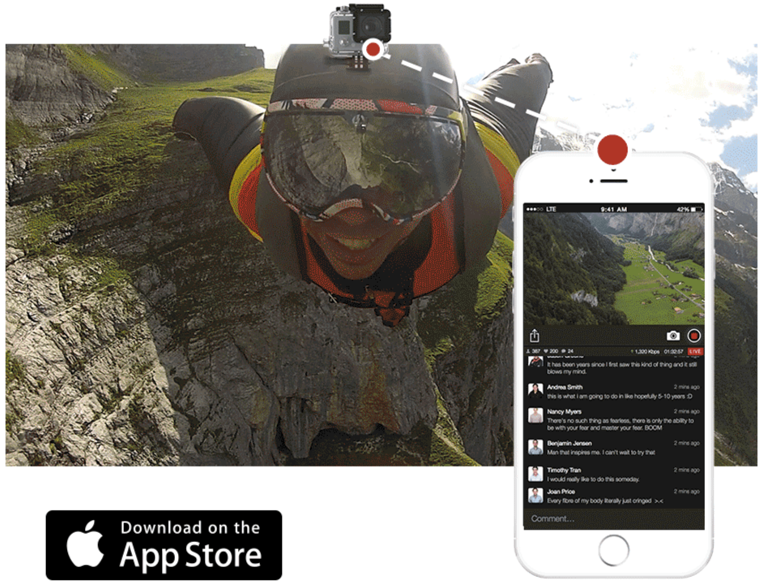
- **MediaPlayer in a Service:** Can play media (e.g. music) in background while app is not running
  - Start MediaPlayer as service
  - To avoid screen going to sleep while service is running, use wake lock

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 messaging protocol by Adobe), or other 3rd party APIs
  - RTMP enables live video streaming over the internet
  - Chops large video files into smaller packets that can be sent separately, reassembled at destination

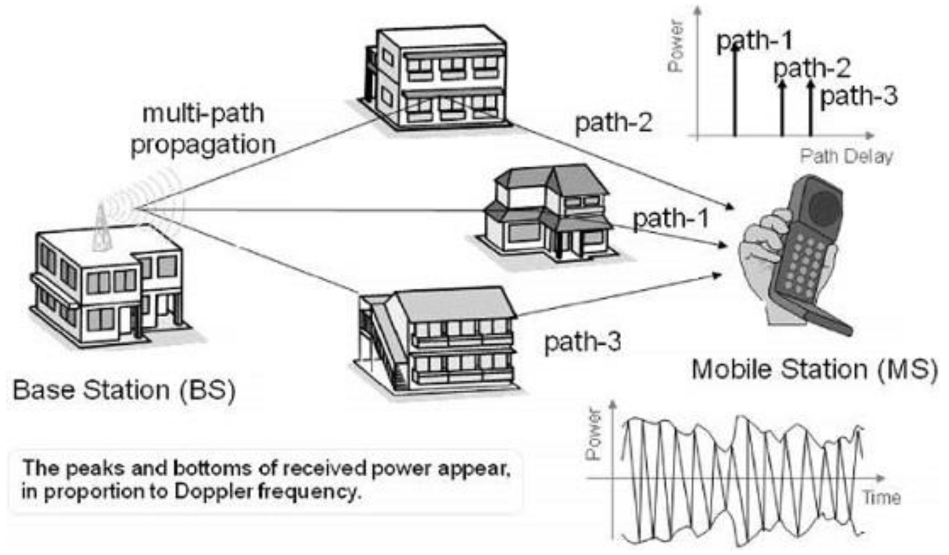
Facebook Live



Live GoPro

Live Streaming Bandwidth Issues

- On WiFi, bandwidth is adequate, high quality video possible
- Cellular links challenges:
  - Lower bandwidth than WiFi
  - Variable bandwidth (multi-path fading)
    - Even when standing still
  - Links optimized for download not upload
- Video quality increasing faster than cellular bandwidths
  - Ultra HD, 4k cameras now available on many smartphones makes bandwidth issues worse



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
  - Smartphone uses/combines neighbors bandwidth to improve video quality
  - Streaming smartphone: WiFi Direct connection to neighbors
  - WiFi Direct allows smartphones connect directly, no Access Point

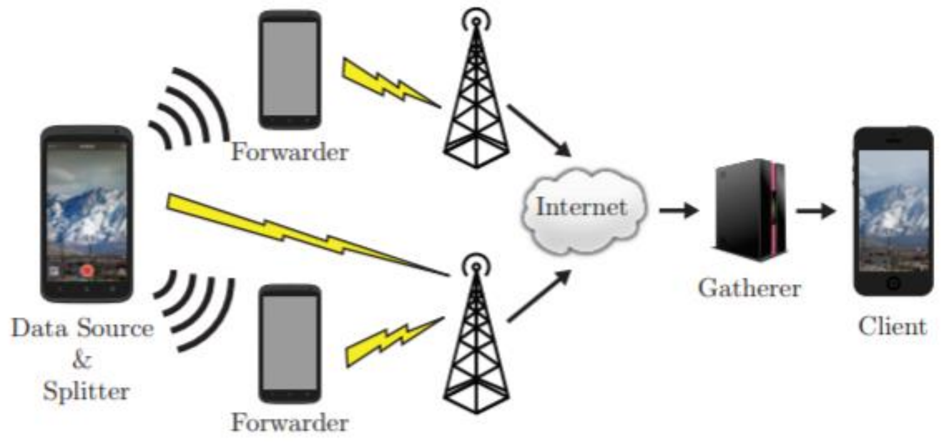
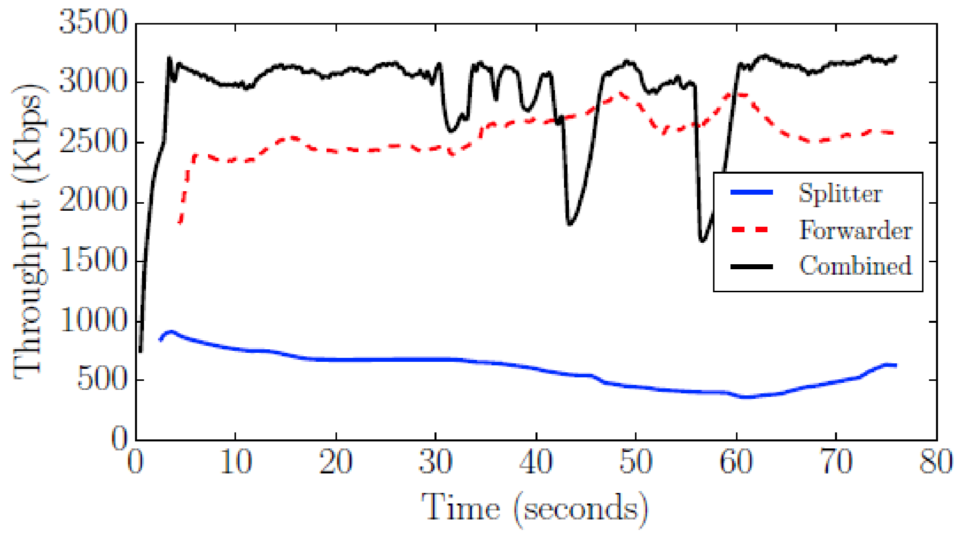


Fig. 1. General architecture of mobiLivUp. Data passes from the splitter to forwarders, then to the gatherer through their cellular connections.

Live Streaming

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

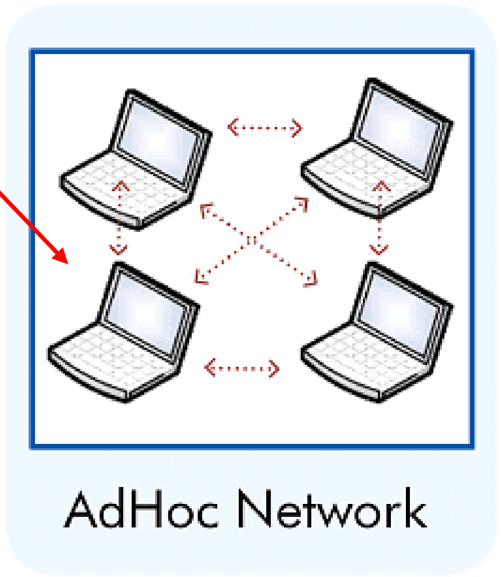
- **Results:** 2 smartphones 88% throughput increase vs 1 phone

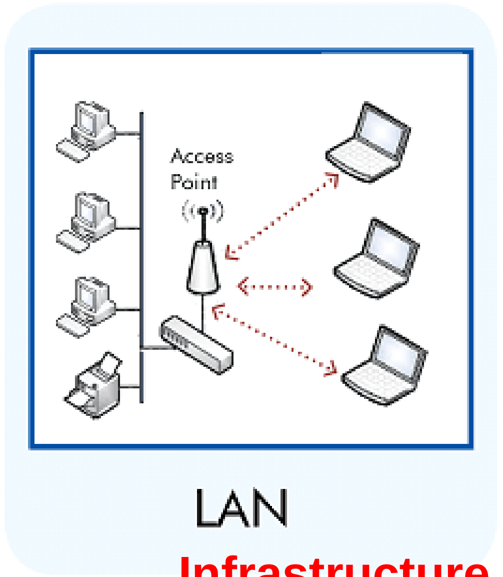


- **Issues:**
  - Video packets travel/arrive out of order
  - Incentives for forwarding nodes?

WiFi: Ad Hoc Vs Infrastructure Mode

- **WiFi: 2 Modes:**
  - **Infrastructure mode:** Mobile devices communicate through Access point (AP)
  - **Ad Hoc Mode:** Mobile devices communicate directly to each other (no AP required)
- **WiFi Direct** is new standard for ad hoc WiFi mode





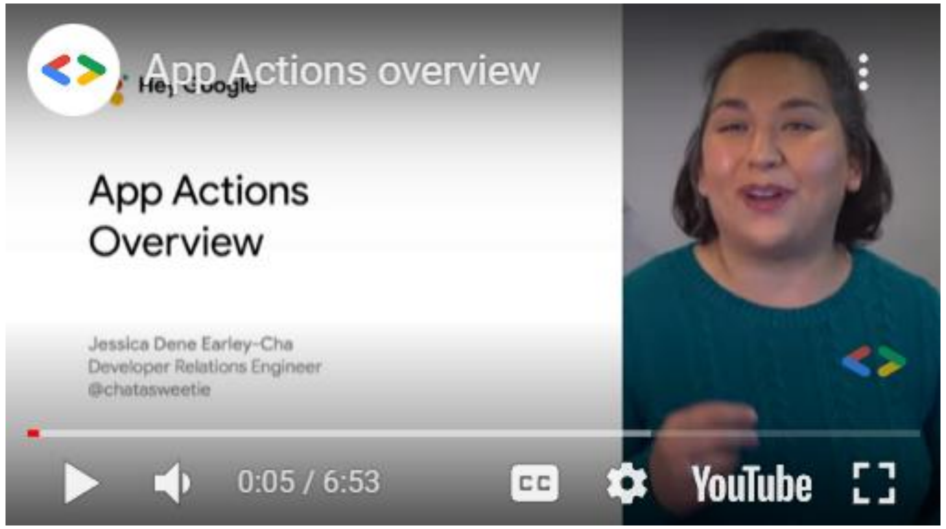
Infrastructure mode

Google Assistant for Android

Google Assistant for Android

https://developer.android.com/guide/app-actions/overview

- Google Assistant enables voice-forward control of Android apps
- Users can use voice to:
  - Launch apps
  - Perform tasks
  - Access content
- E.g. Hey Google, start a run on example app



References

- Android Nerd Ranch, 5th edition
- Google Android Tutorials
- Network programming with Android, https://slideplayer.com/slide/8471042/
- Android Networking, https://developer.android.com/training/basics/network-ops