Audiosync Final Presentation

by Simon Grätzer & Jan Garcia

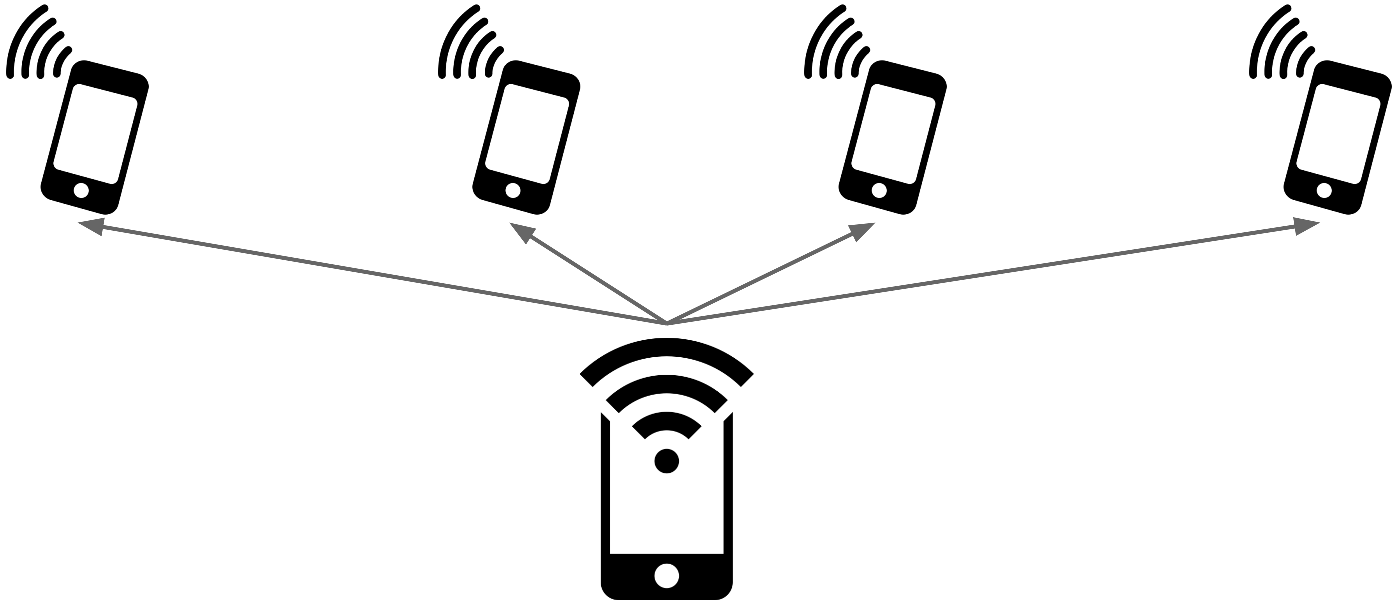
Organization

- Goals
- Sync. audio streaming easy?
- Development timeline
- Software architecture
- Network protocol
- Future work
- Demo

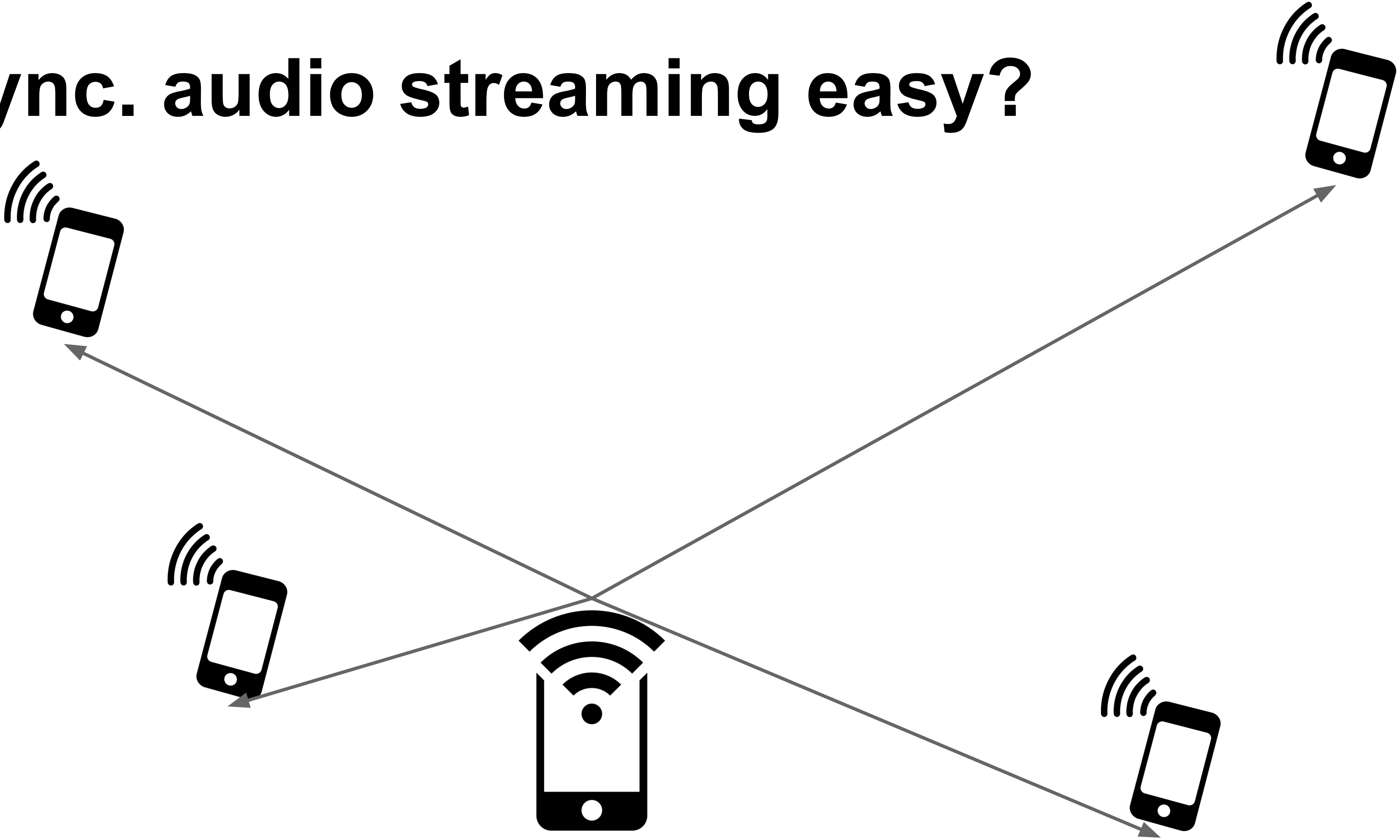
Goals

- Synchronous audio playback on multiple devices
- Wifi-Streaming
- One sender, many receivers
- Support for heterogeneous receivers
- No setup required

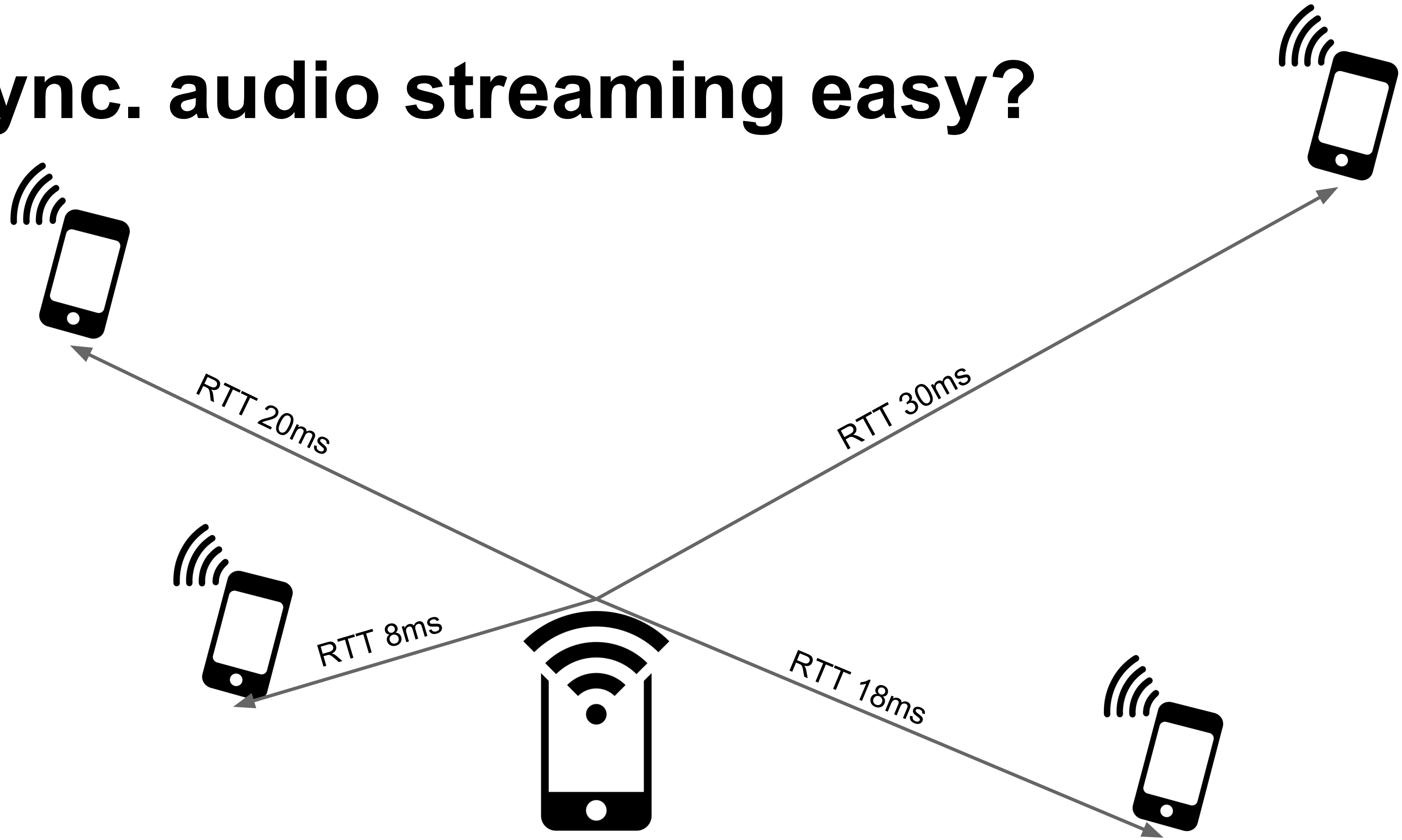
Sync. audio streaming easy?



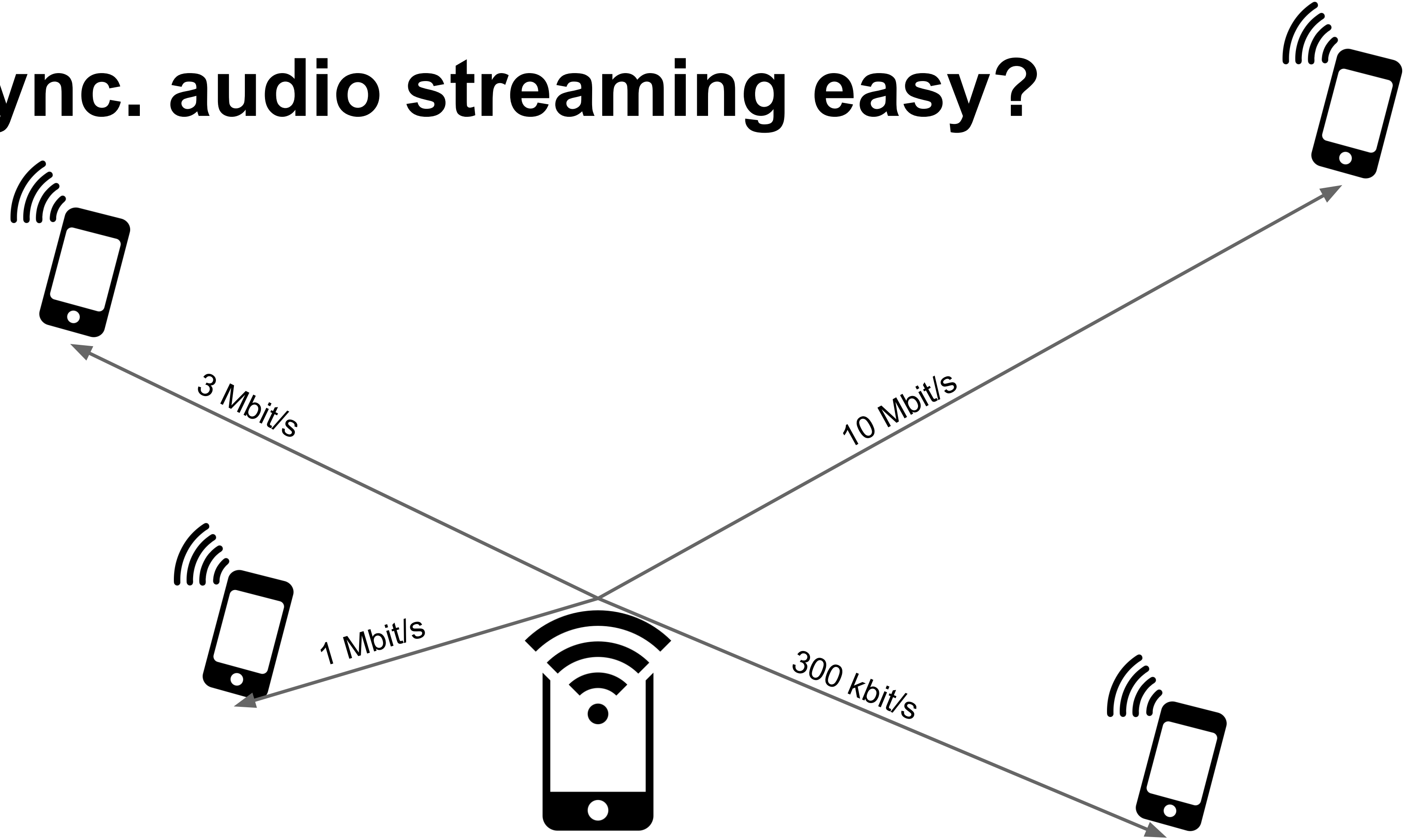
Sync. audio streaming easy?



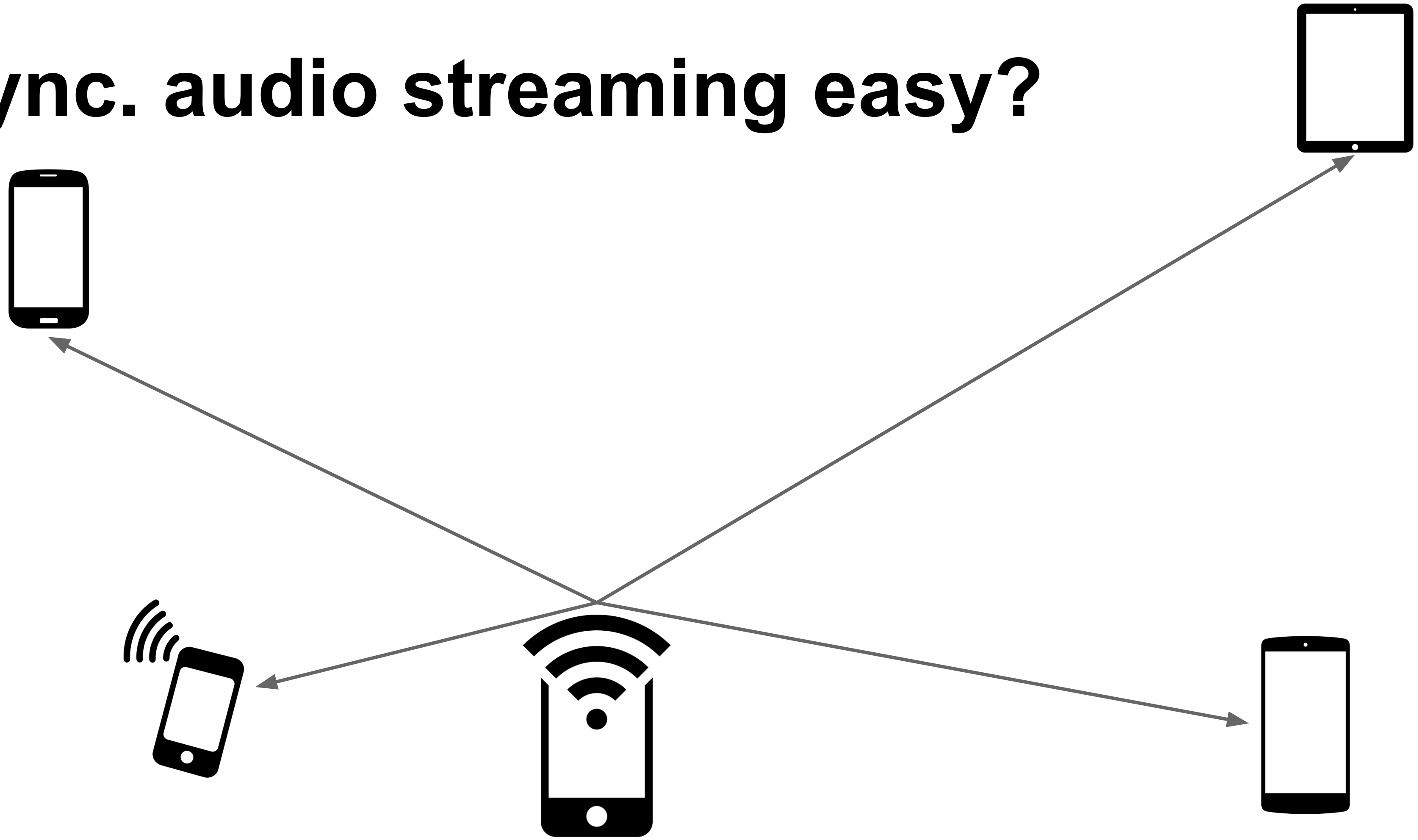
Sync. audio streaming easy?



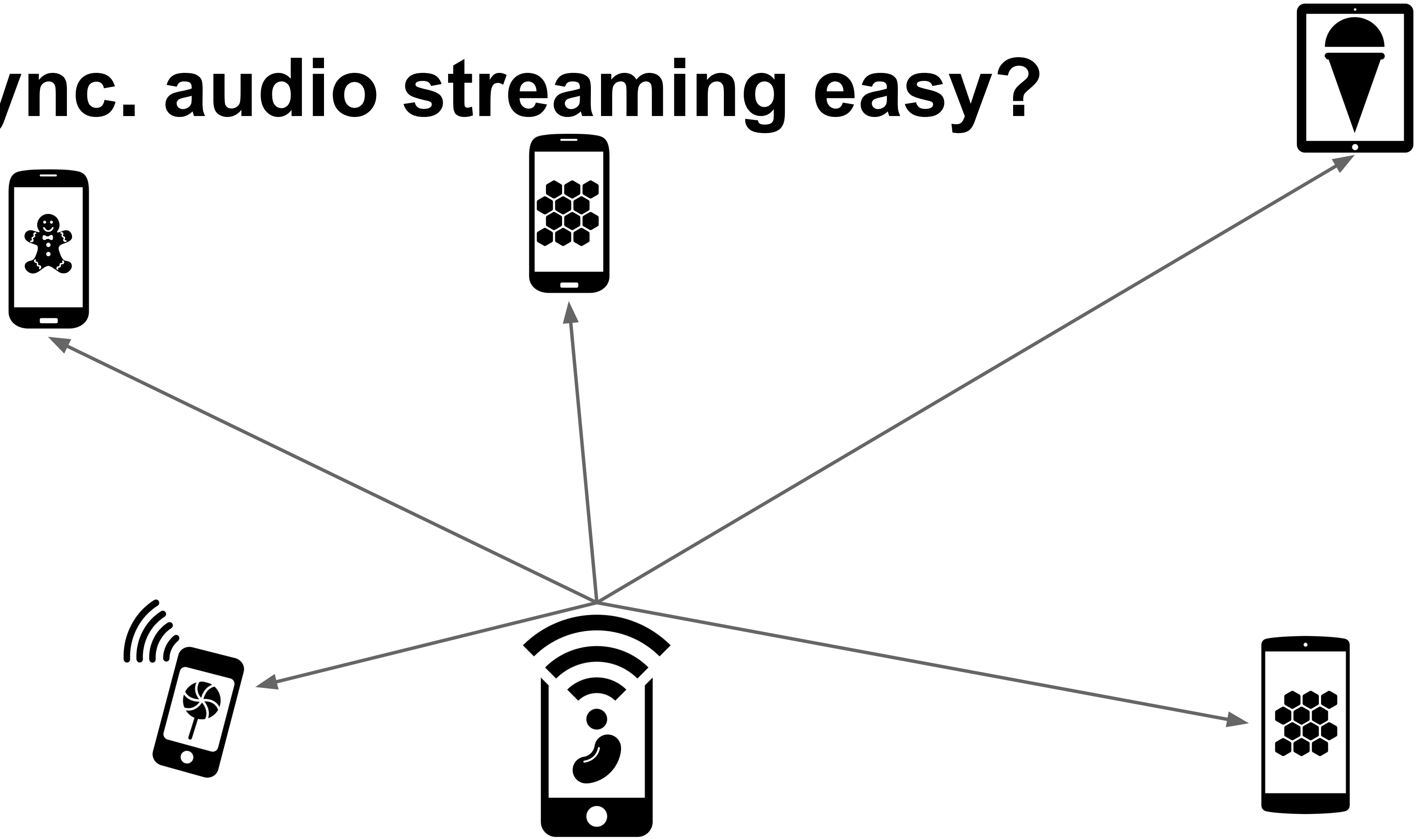
Sync. audio streaming easy?



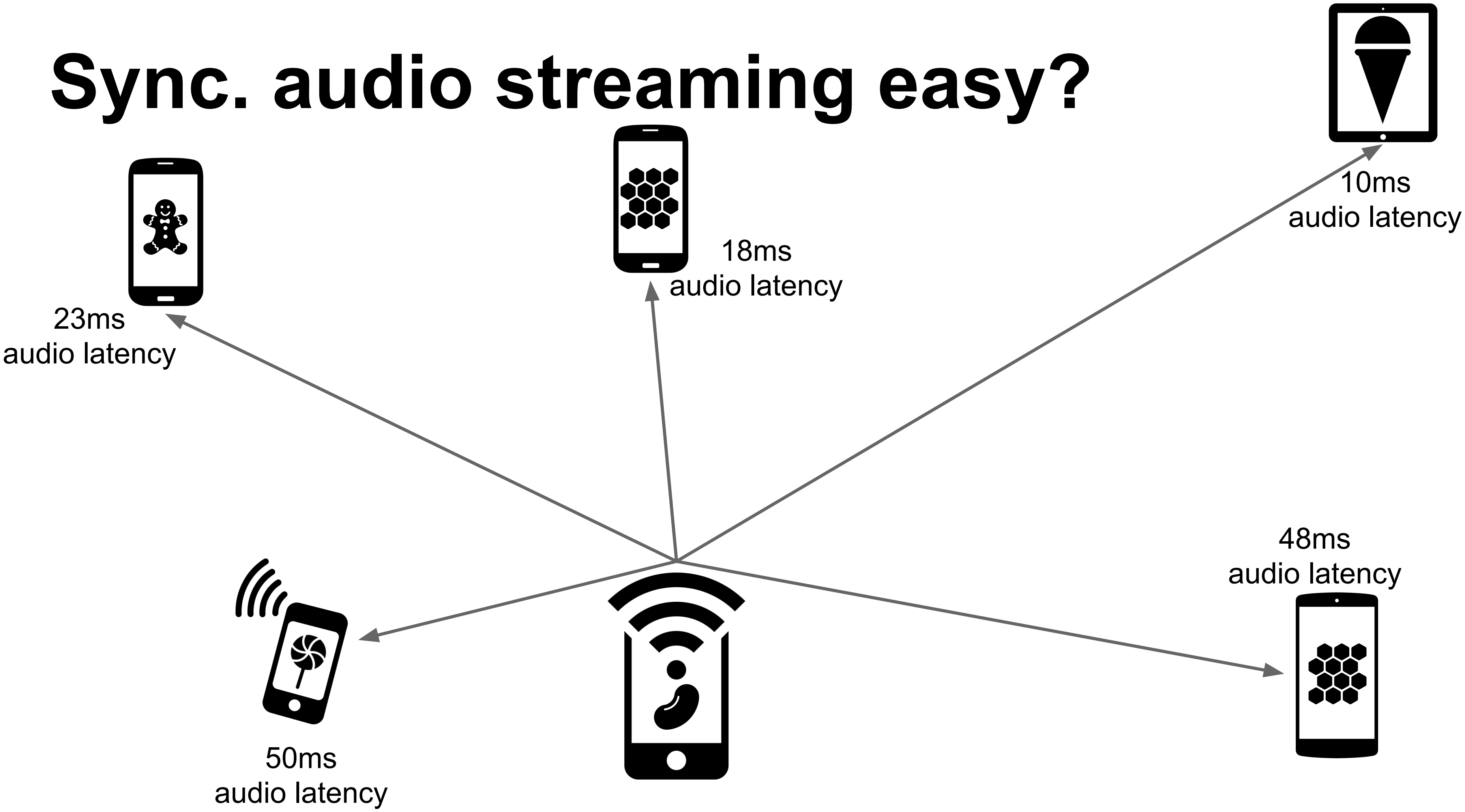
Sync. audio streaming easy?



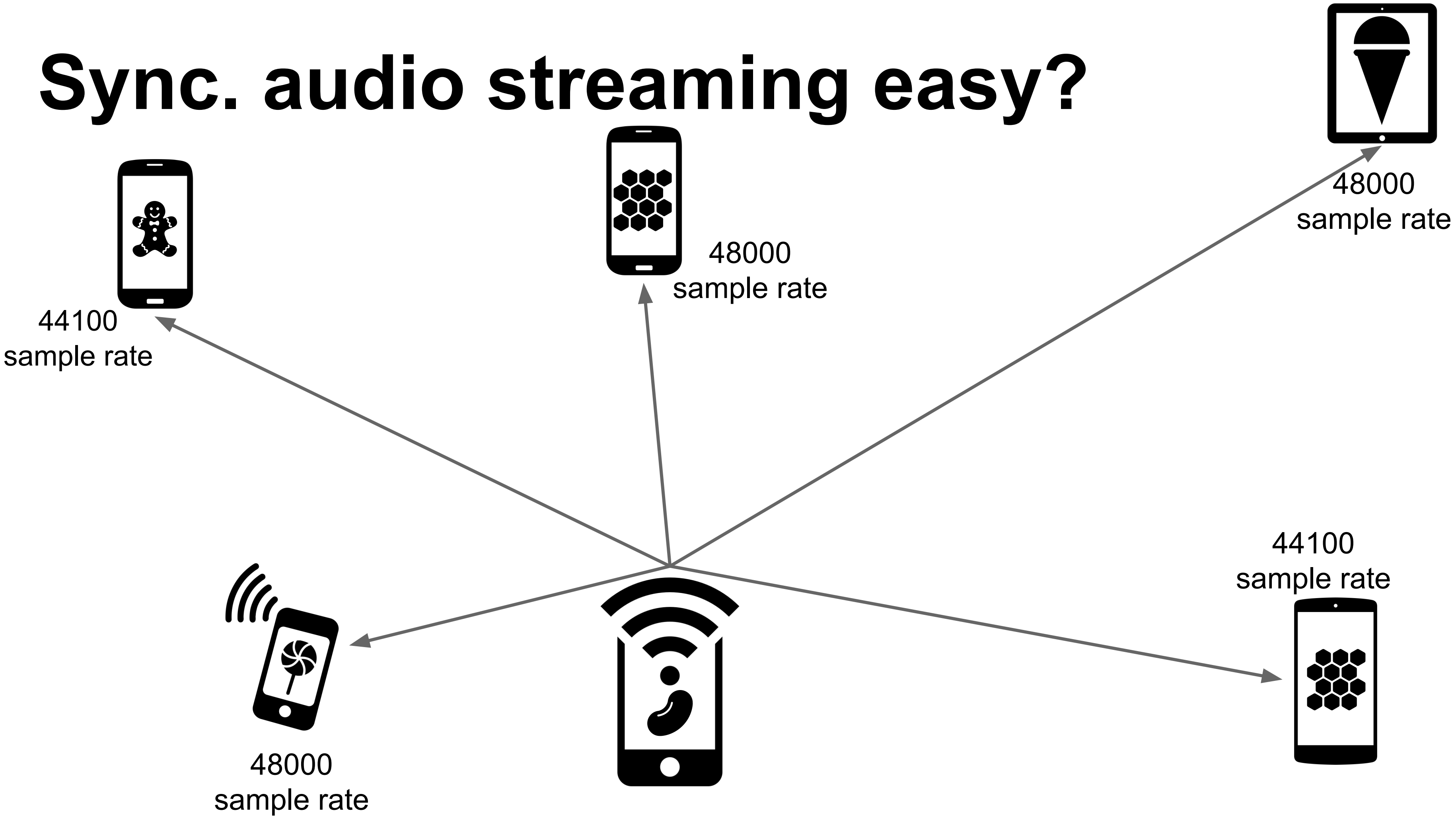
Sync. audio streaming easy?



Sync. audio streaming easy?

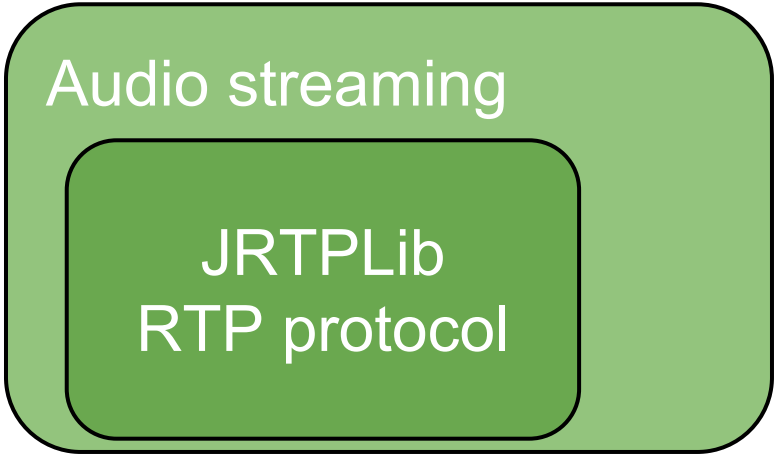


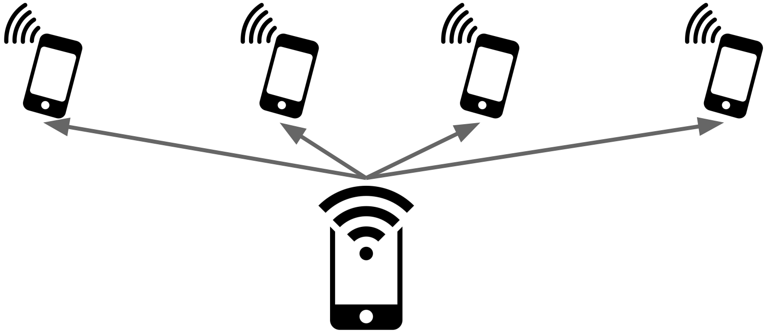
Sync. audio streaming easy?



Step 1: Streaming some audio

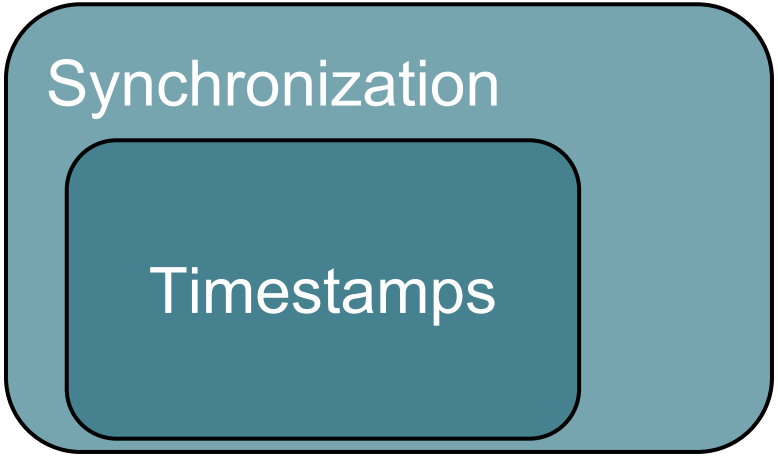
- Wifi broadcast
  - Too slow, 1 MBit/s
- Unicast, stream simultaneously
  - Existing protocol: RTP

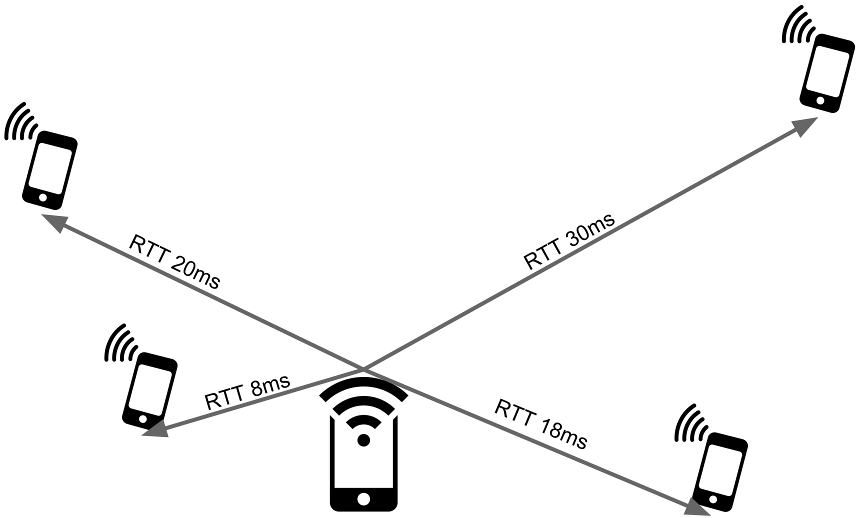




Step 2: Synchronize playback times

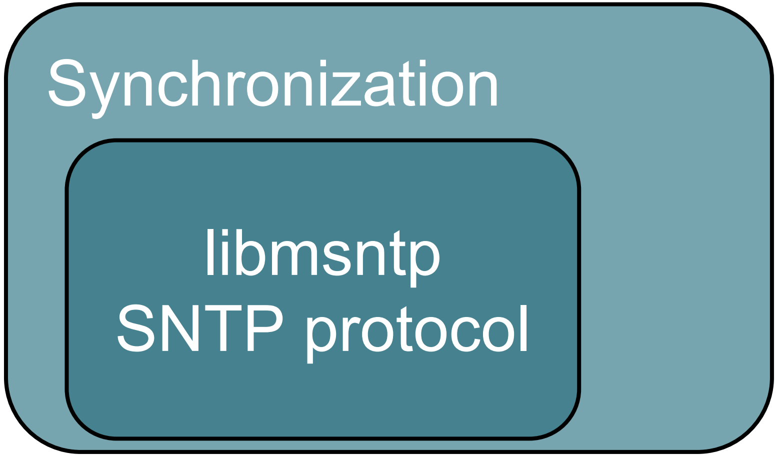
- Extend RTP protocol (extension header)
- Audio packets have timestamps
    - Time when to start playing packet

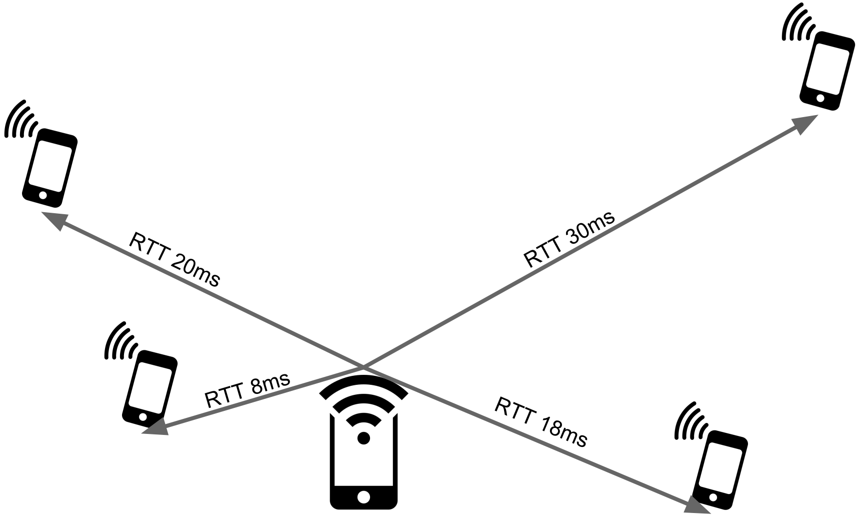




Step 3: Synchronize device times

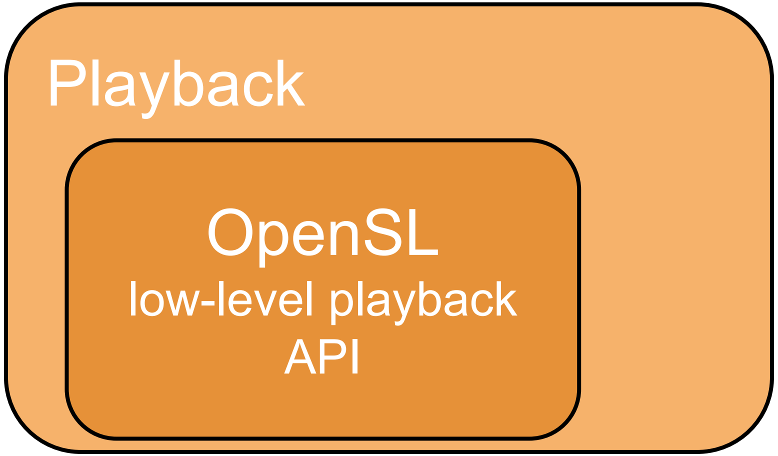
- Existing solution: NTP protocol
- We use a simplified version
  - SNTP

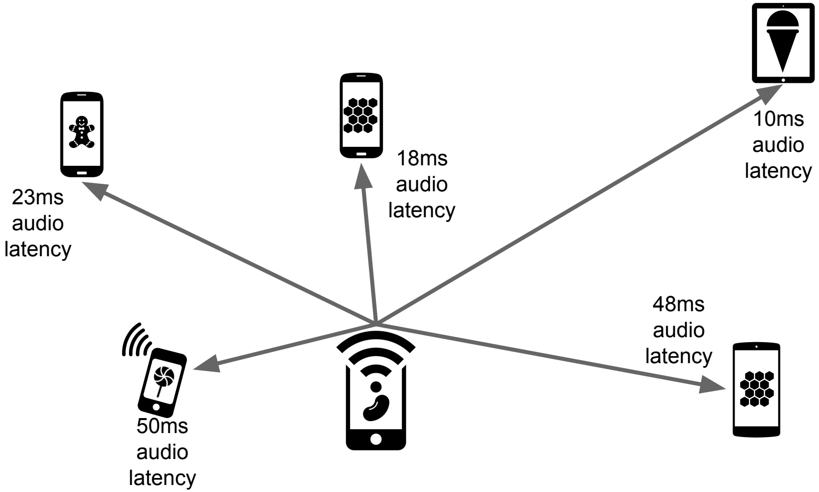




Step 4: Reduce playback latency

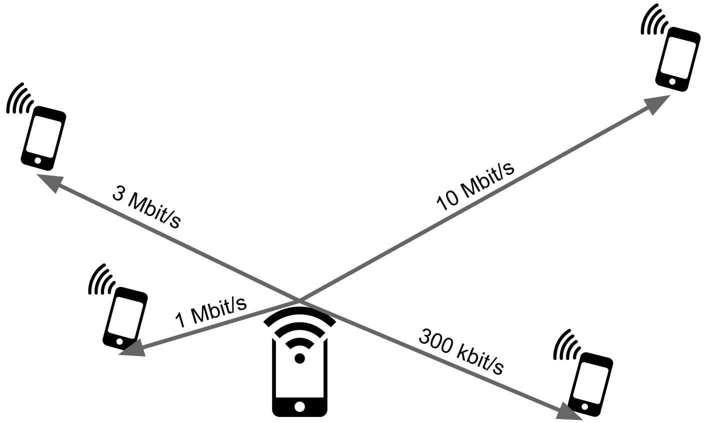
- High level APIs
  - easy to use, higher latency
- Low level APIs
  - harder to use, low latency



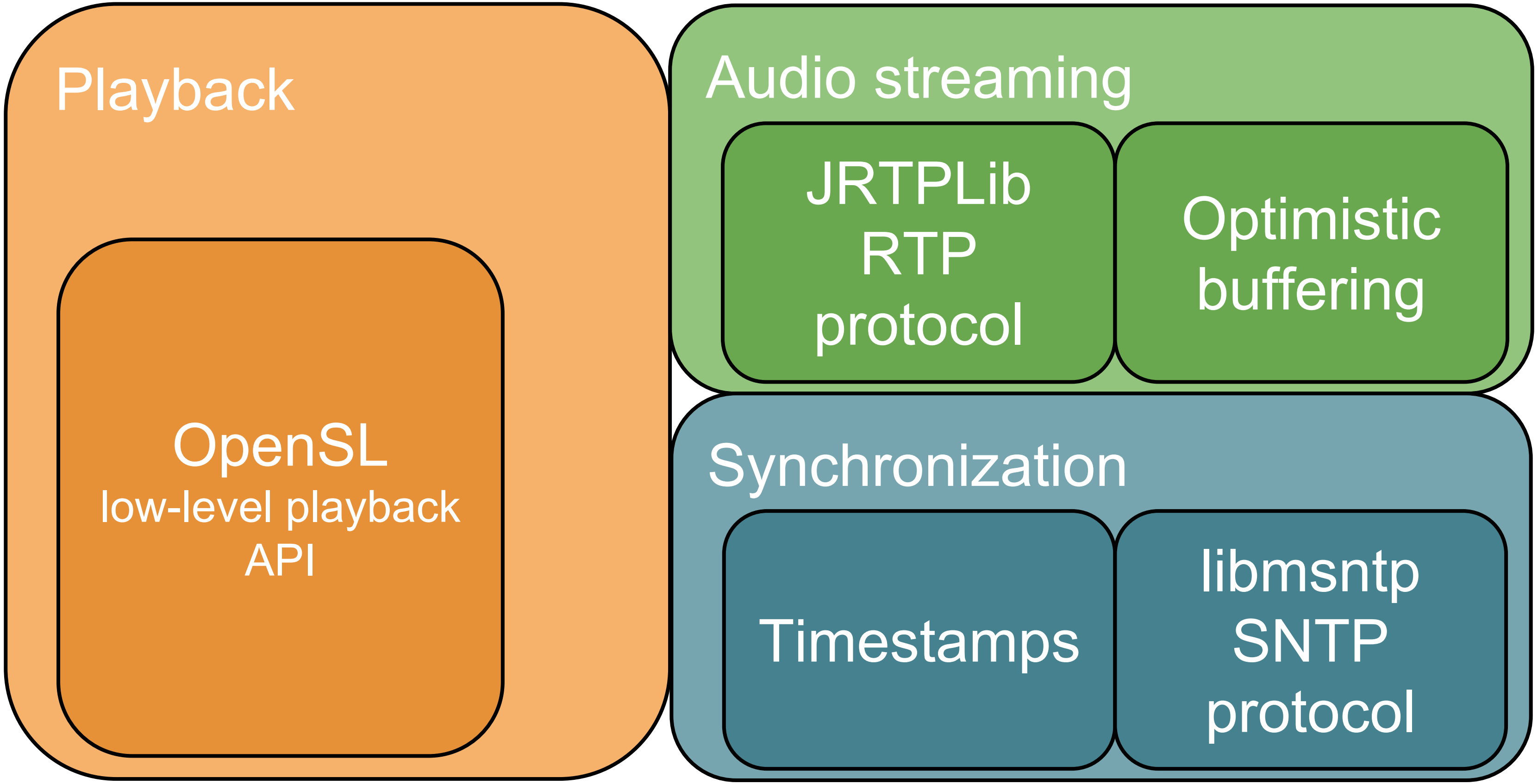


Step 5: Regard individual data rates

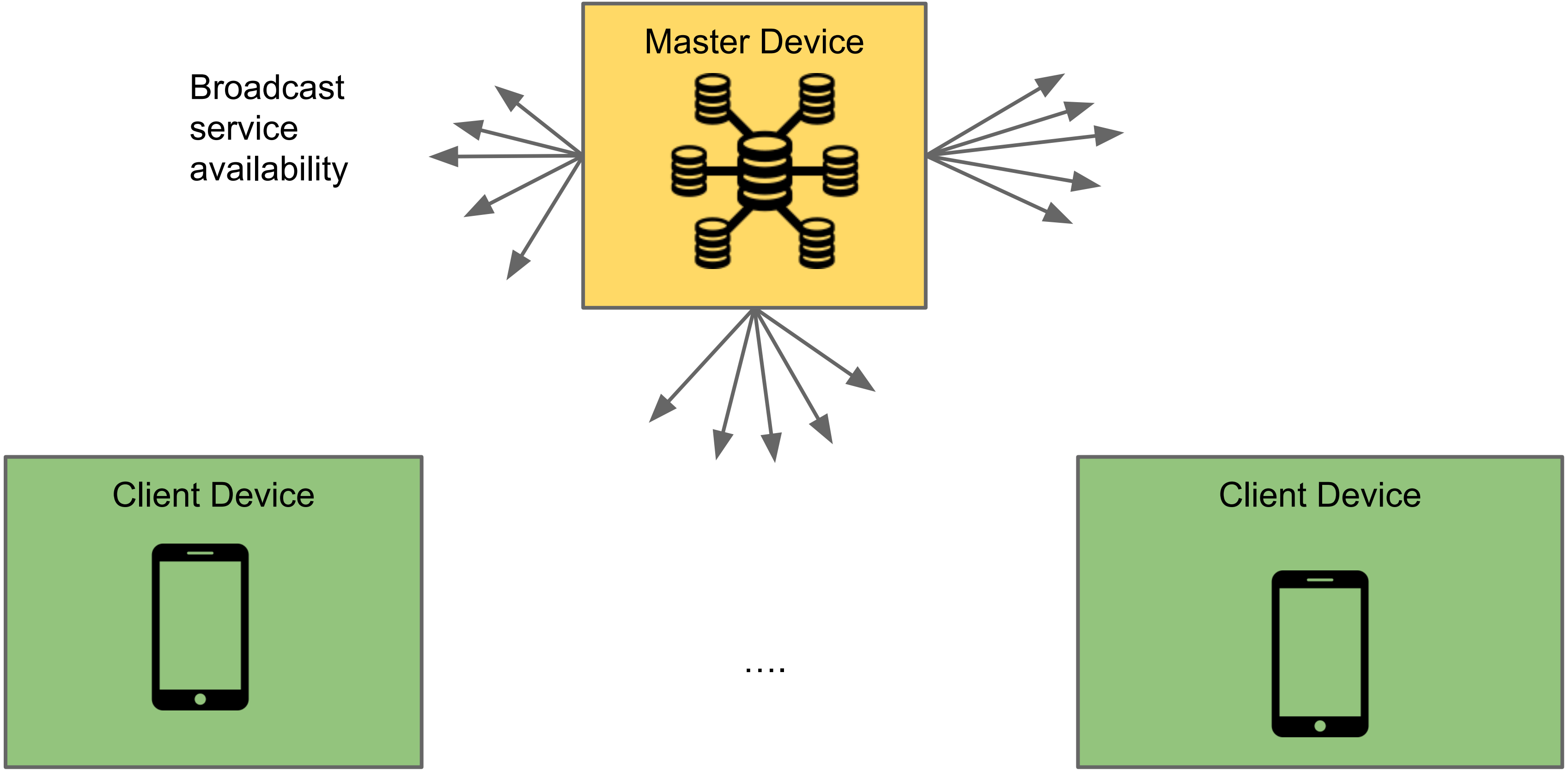
- Our solution
  - optimistic buffering
- Future work
  - report data rate, adjust it in the sender



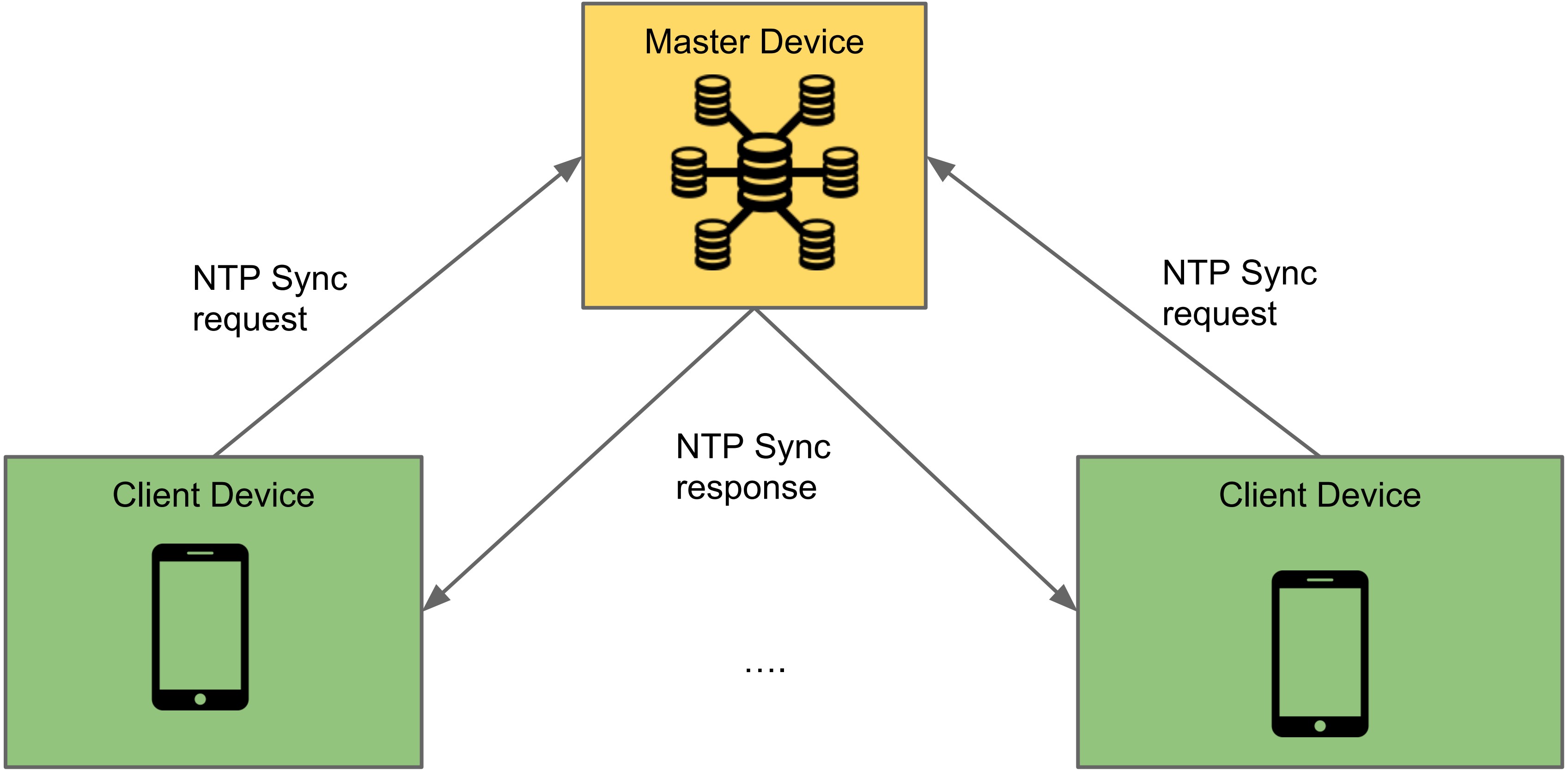
Audiosync architecture



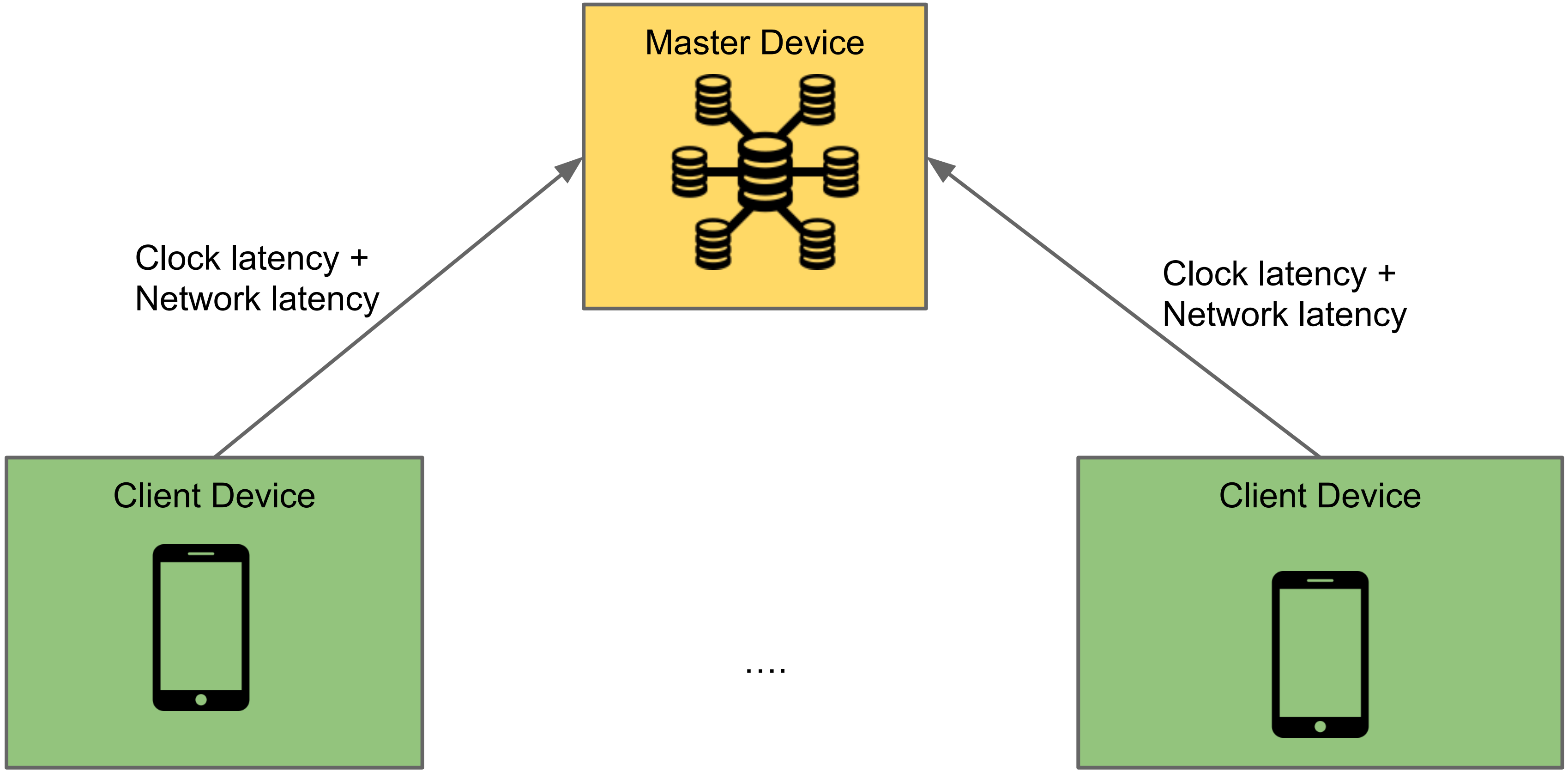
Network protocol step 1



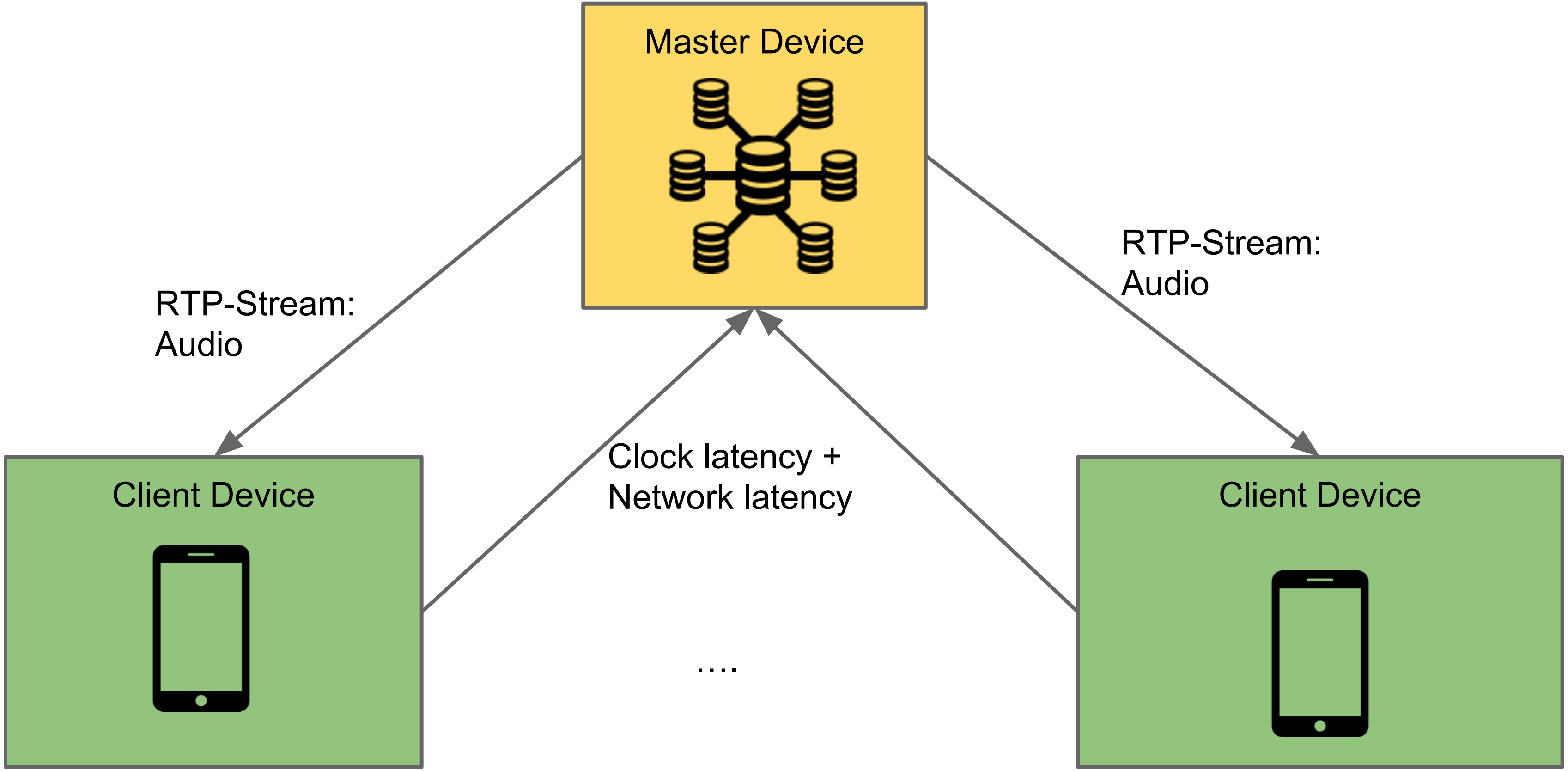
Network protocol step 2



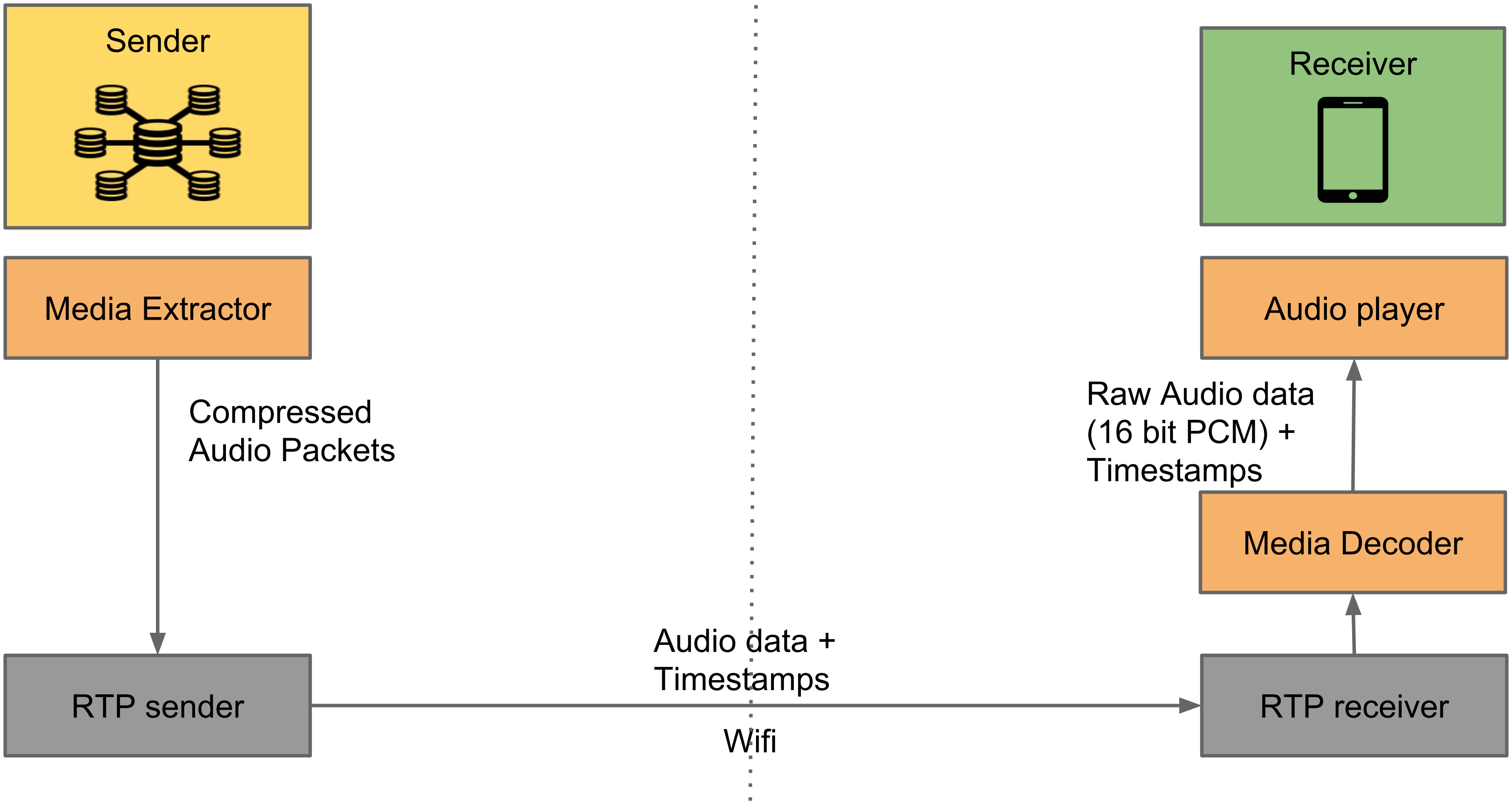
Network protocol step 3



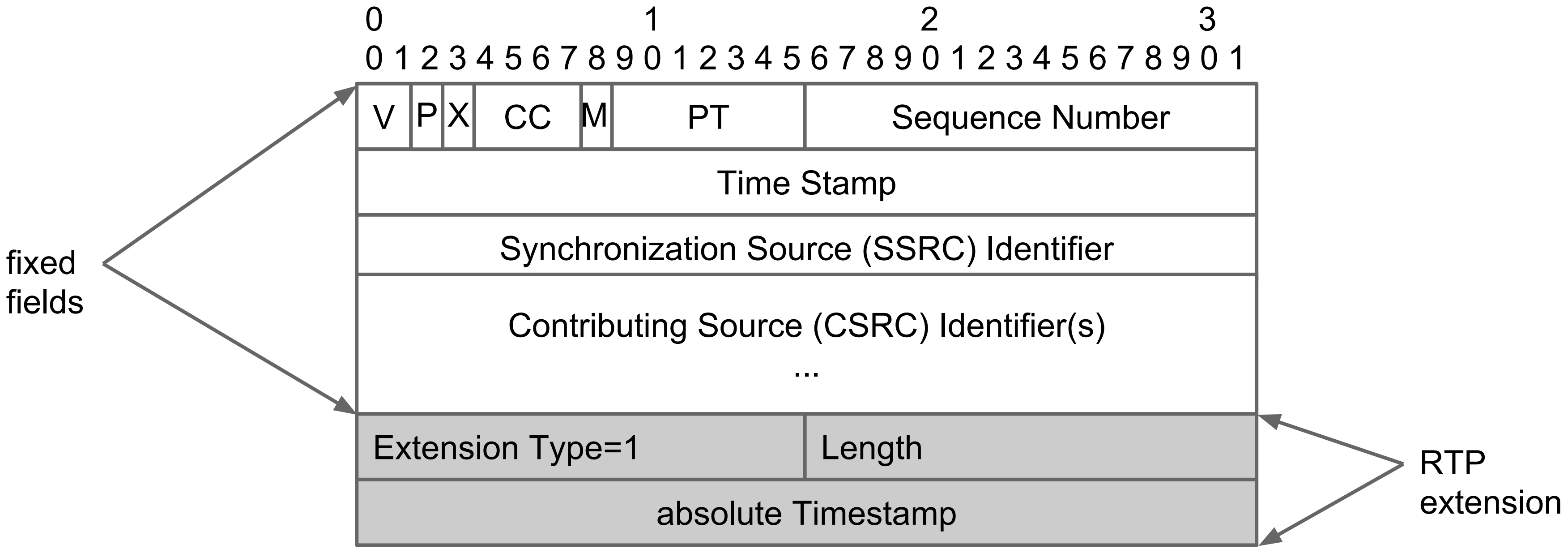
Network protocol step 4



Audio Data Streaming Architecture



RTP Protocol Timestamp-Extension



- RTP packets are timestamped
- Relative Timestamp for position in audiofile
- Absolute time of when the content should be played

Audio Player Implementation

- Key piece of the entire system
- Responsible for matching the playback time with the timestamps

→ Sometimes easier said then done

Assumptions about audio playback

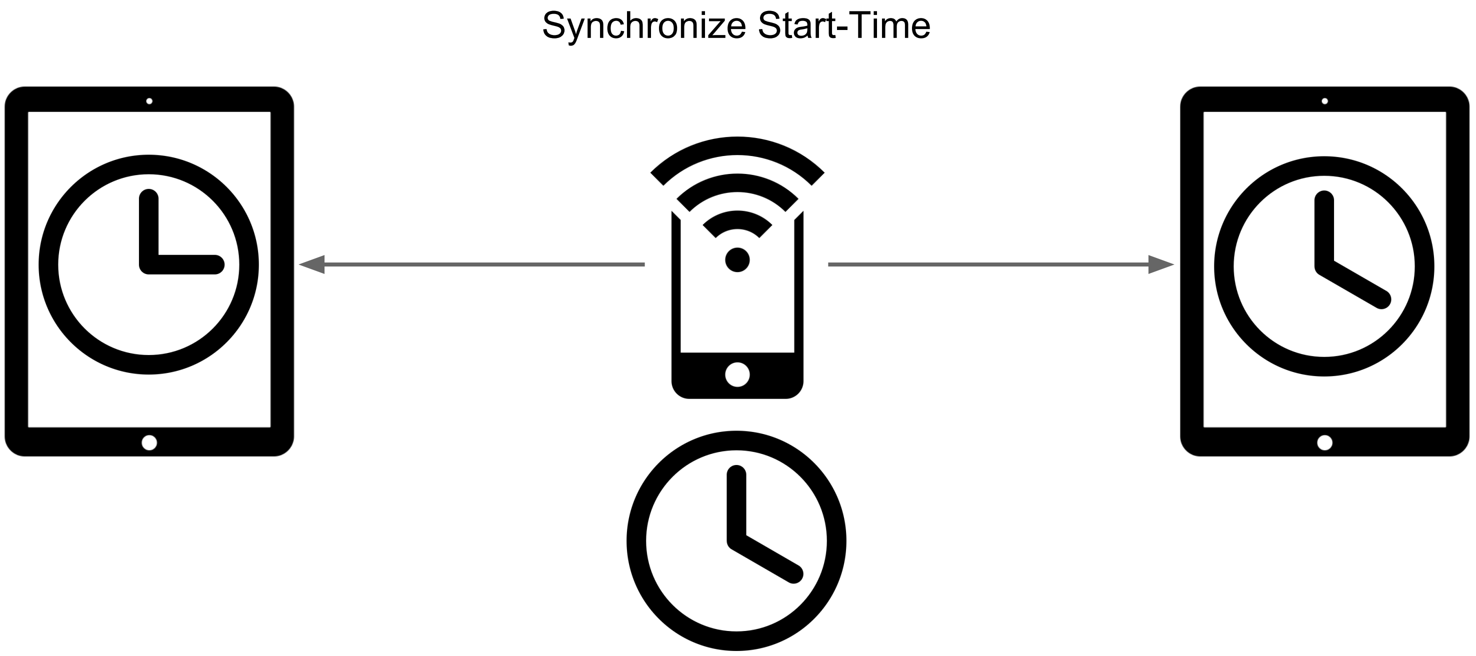
- Starting playback will have no big latency
- Playback speed will remain approximately constant
- Playback speed is the same for all songs
- Same device model’s will have the same playback rate and latencies

→ Nothing of this holds true

Evaluation Setup

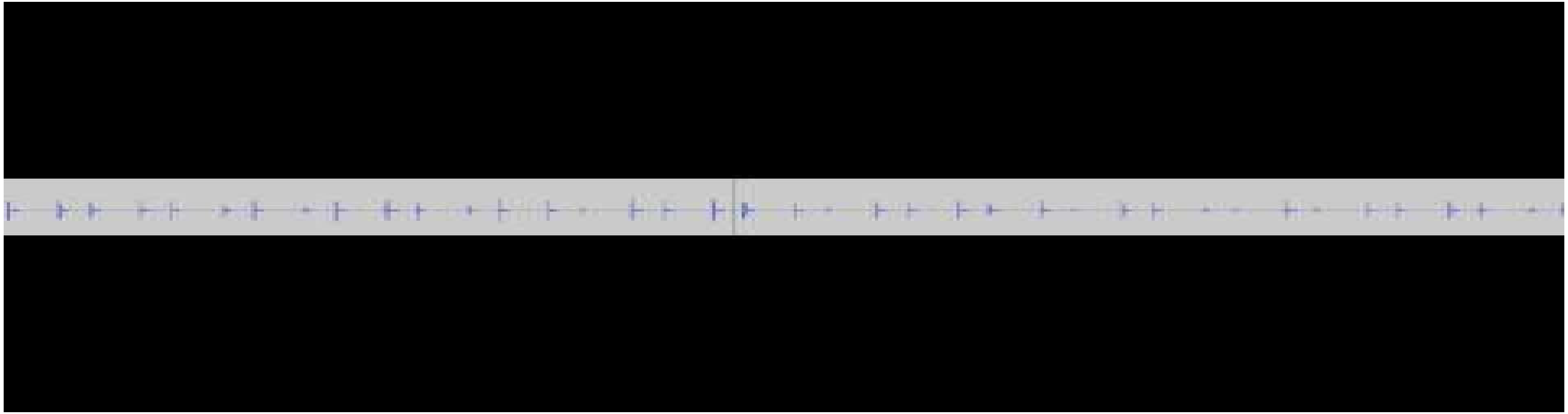
Two Nexus 7 (Model 2013) Tablets

Audio demo time



Synchronizing Playback Start and Skip or Pause playback accordingly

- Try to use NTP clock offsets to control exact start moment
- Reduce differences by skipping or pausing

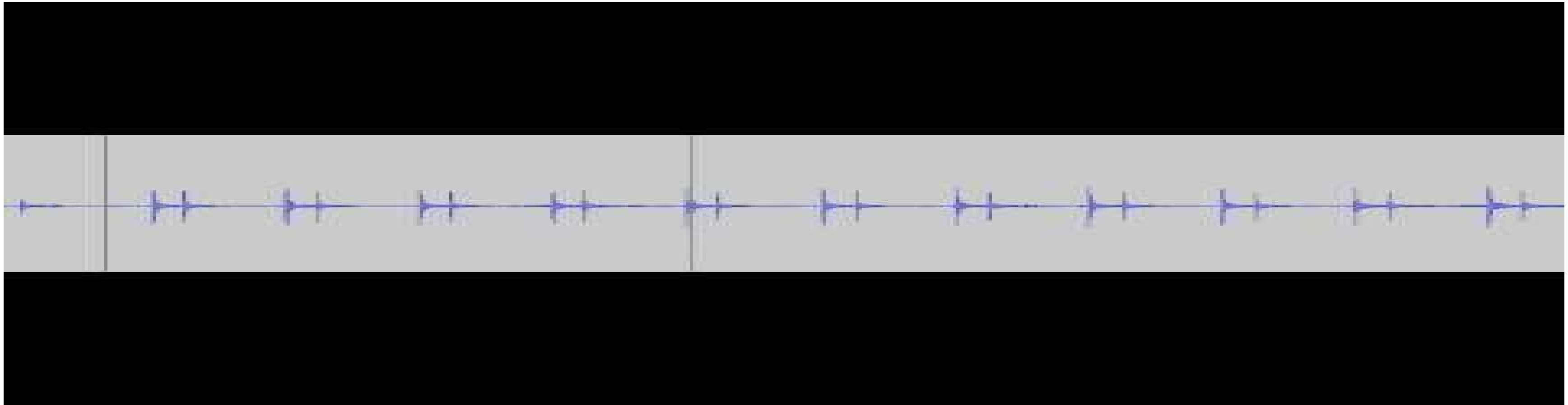


Audio demo time

Playback-Rate compensation

Control Playback Rate

- For 4 seconds disable skipping / pausing and measure playback speed
- Then enable skipping / pausing for the accumulated time difference
- Playback is only off by a constant amount (~200ms)

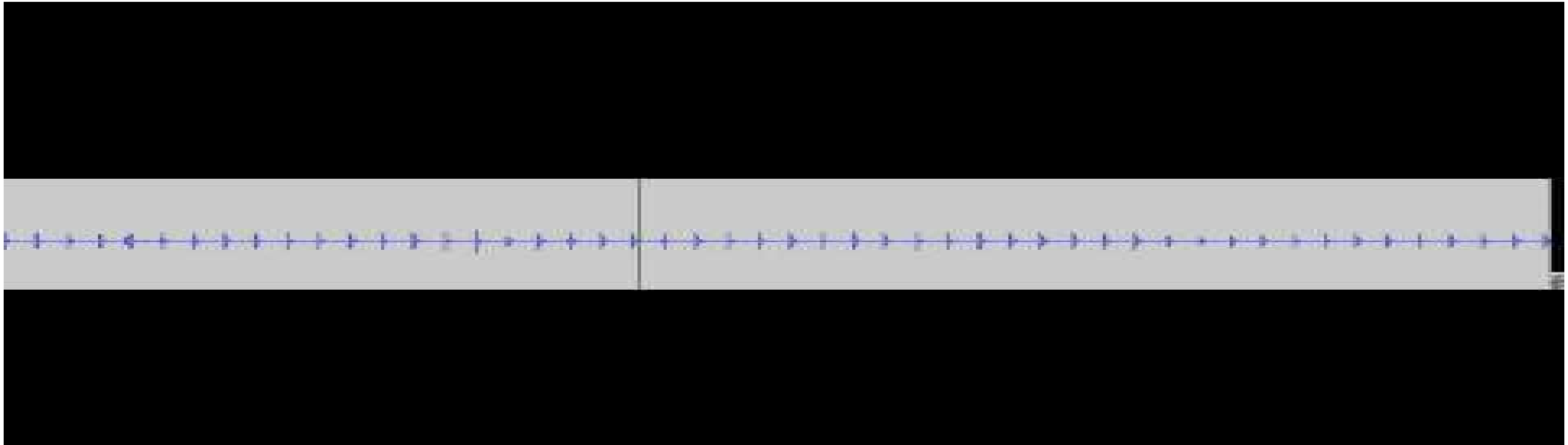


Take Device Latency into Account

Add some static latency to the lagging device

Sometimes the playback jumps

After 4 seconds the rate is re-adjusted



Future work

- Using OpenSL is not enough
  - take device-specific playback latencies into account
  - Measure system latency while using all our audio effects
- QOS
  - Measure data rate and adjust audio quality
  - E.g. decrease playback rate on all devices if we don’t have enough bandwidth

References

- **An Internet Protocol Sound System** (2004): Bob Atkinson, Tom Blank, Michael Isard, James D (JJ) Johnston, and Kirk Olynyk
- Iconography: Aaron K. Kim, Creative Stall, Pantelis Gkavos, Kevin Kwok, Roberto Chiaveri, Martin Jordan, David Lopez, Alessandro Suraci, Edward Boatman, Mario Bieh from Noun Project

Thank you for your attention!