Rā ltime wireless g aana stre am ing

Final Project Presentation

under the guidance of

Prof. Kavi Arya

Bharat Reddy, Diptesh Kanojia, and Vinod Saini

CS684: Embedded Systems Project

CSE Department, IIT Bombay

27th October, 2016

Roadmap

- Problem Statement
- Requirements
- Work Allocation & Timeline
- Implementation
- Innovation & Challenges
- Tasks Completed
- Testing
- Reusability
- Future Enhancements

Problem Statement

- To develop a device which can stream audio from PC / Mobile to speakers.
- The device should make use of Wi-Fi wireless protocol to stream data.
- It should also be able to stream audio at a higher bitrate.
- The device should be able to perform without any lags whatsoever.

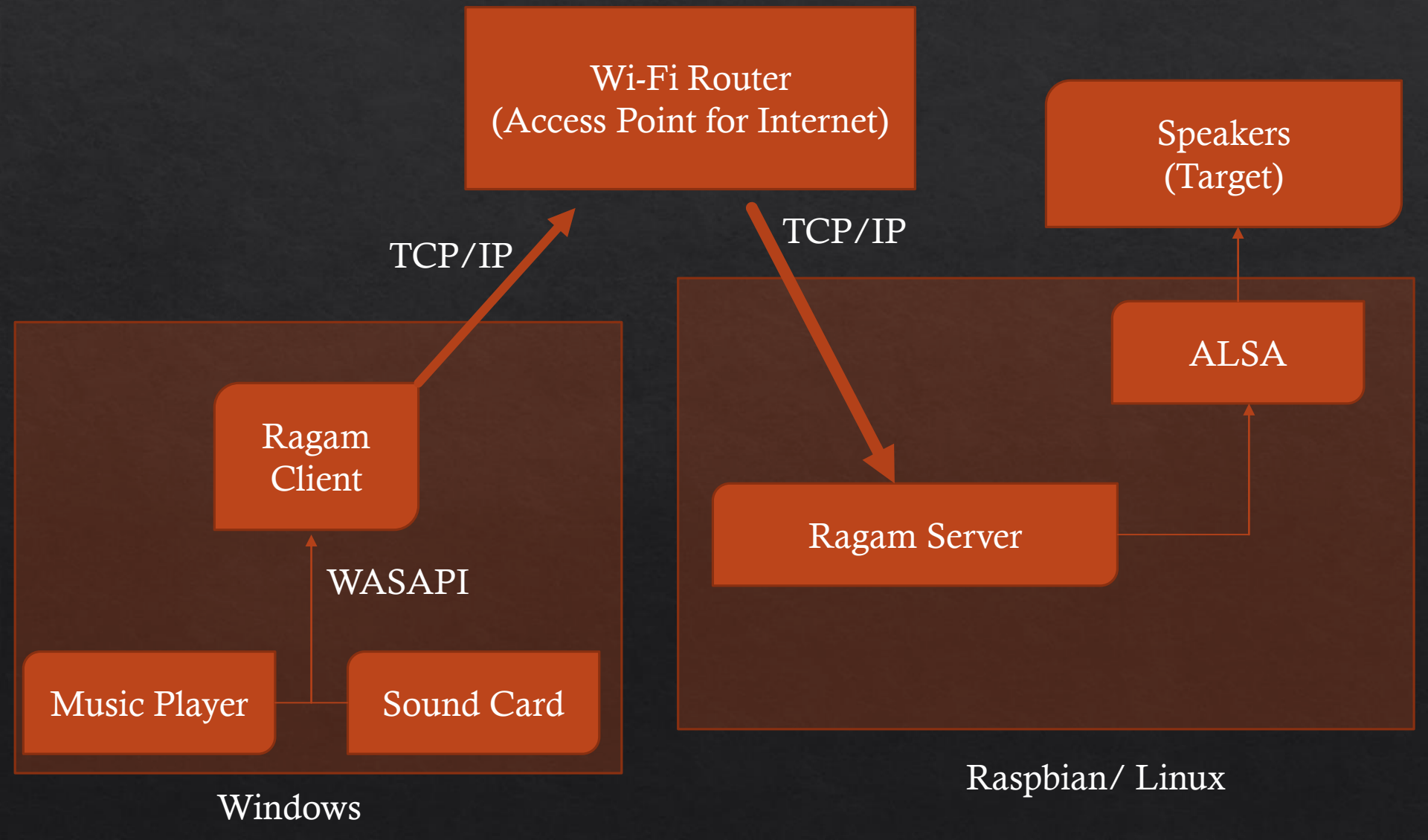
Requirements

- Raspberry Pi 3~~/ Arduino (primarily testing)~~
- Wireless router (primarily testing)
- PC / Laptop / Mobile Device (programming and testing)
- ~~USB sound card (good quality)~~
- Wires to connect (Power, LAN, Audio cable etc.)
- C/C++11
- RaspbianOS / ~~Ubuntu~~ (Server Side)
- Power Sockets with Electric Supply!

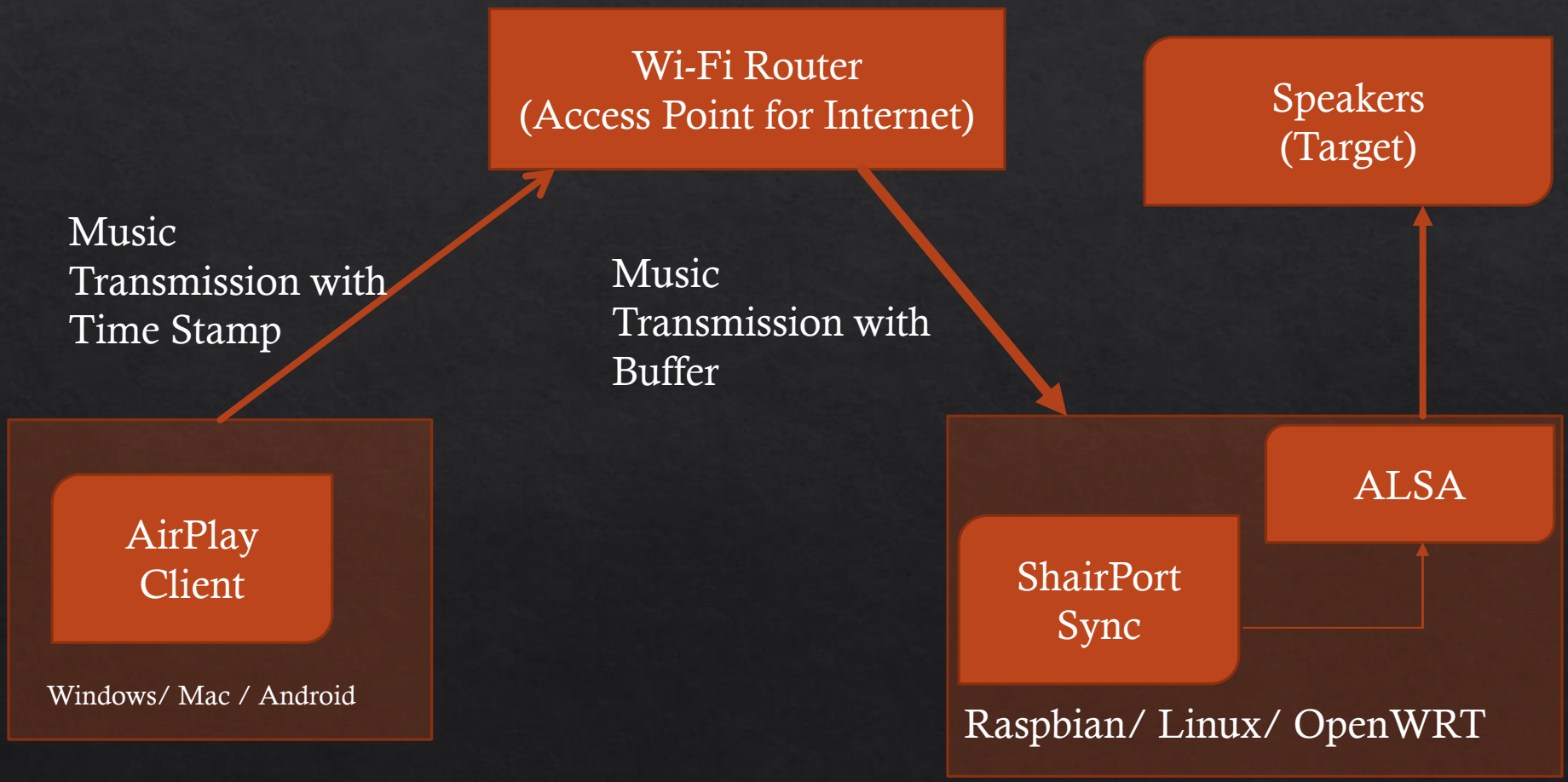
Work allocation & Timeline

- Work Allocation
  - Bharath:
    - Sending data from Client side using WASAPI
    - Initial Setup
  - Diptesh and Vinod
    - ***Trying multiple other available implementations which failed. (PiMusicBox, Mopidy Server, MPDroidServer, and SoundWire Server, paprefs in Ubuntu, SWYH in Windows, nAudio in Windows)***
    - Receiving data on Server side using ALSA driver in Ubuntu.
    - Setup (configuring WiFi module on RPi to connect to local WiFi).
    - Implementation 2 with Shairport - Sync
- Timeline:
  - ☑ Procurement and Setup - 1 Week
  - ☑ Server Side setup – 2 weeks
  - ☑ Client side Audio Streaming API – 2 Week
  - ☑ Quality Control and Testing – few days

Implementation-1



Implementation-2



Innovation & Challenges

- We have used the Windows Audio Session API for capturing sound data from the output buffer.
- This allows us to capture data, which is then transferred to Server Side or can be saved as a wave file.
- Audio lag reduction, and underrun errors on Raspberry Pi while using ALSA API.
- Buffer Allocation due to low memory of Raspberry Pi.
- Understanding of WAS and ALSA API

Tasks Completed - I

- We can play low bitrate Audio at server side without any lag whatsoever (through iTunes).
- We can play highest available bitrate Audio (ALAC) files also without any lag on the server side.

Tasks Completed - II

- Capturing audio on the client side using the WASAPI which can be written to a WAV file.
- Socket connection establishing and sending data to Raspberry Pi.
- Playing generic system audio on Raspberry Pi device with minimal lag (low latency).
- In this case, we are capturing audio from sound card, so it doesn't matter which audio player we use, this being a limitation of our previous implementation.

Testing

- User Experience
  - Delay time
    - Method - I
      - Client Side (Windows PC) - > Server Side (RPi) - > Sepakers
    - Method - II
      - Client Side (Windows PC) - > Server Side (RPi) - > Sepakers
      - Client Side (Android) - > Server Side (RPi) - > Sepakers
      - Client Side (Apple Devices) - > Server Side (RPi) - > Sepakers
  - Audio Quality
    - Method - I
      - Client Side (Windows PC) - > Server Side (RPi) - > Sepakers.
    - Method - II
      - Client Side (Windows PC) - > Server Side (RPi) - > Sepakers
      - Client Side (Android) - > Server Side (RPi) - > Sepakers
      - Client Side (Apple Devices) - > Server Side (RPi) - > Sepakers

Reusability

- Shairport Sync is a open source library which creates a server on Raspberry Pi / Ubuntu / Debian / OpenWRT.
- In WASAPI, We have tried to implement the code using functions which can be re-used.
- We have mostly use the WASAPI, which supports function calling, and thus even our part of the code is reusable.
- Our server side code for Implementation-II are C programs which can be run individually, and could also be called from crontab of the server during bootup.

Future Enhancements

- Music Streaming to an FM frequency.
- We need to have iTunes (or Airplay supportive client side), we could later obliterate this limitation if we implement an open source uPnP / BubblePnP Server.
- Currently for implementation two, we can only use Windows, we will have to write the same OS level audio capturing for Linux.
  - Most of the parameter in implementation are hard-coded due to time constraint, we could have a configuration file for them.
  - Downsampling the file before sending data to server to reduce bandwidth.
  - Supporting multiple audio format

References

- http://make.witworks.com/preorder/trippy [5th September 2016]
- https://gramofon.com/ [15th September 2016]
- https://www.bose.com/en_us/products/speakers/wireless_speakers/soundtouch-10-wireless-system.html [15th September 2016]
- http://www.ebay.in/itm/Bose-Soundtouch-10-/272349437205
- Google Images (Non – Copyright images like Clipart used from OpenClipart, and ClipartPanda)
- https://msdn.microsoft.com/en-us/library/windows/desktop/dd371455(v=vs.85).aspx
- http://alsa-project.org/main/index.php/Main_Page

Thank you! ☺

Questions ?

27th October, 2016