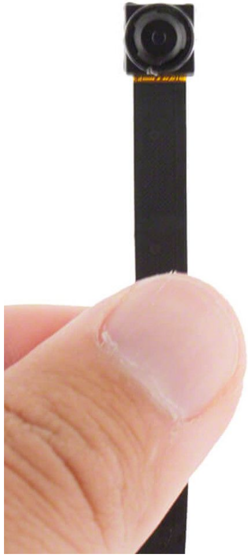
Carnegie Mellon University

S23 18500 Team C2: **WiSpider**

Presenter: Anish Singhani
Group: Ethan Oh, Anish Singhani, Thomas Horton King

Application / Use Case



- Hidden wireless devices are infringing privacy
  - Cameras, microphones, sensors, etc.
- We want to build a product to detect and locate these hidden Wi-Fi devices indoors
- Detect devices even if user is not connected to the same Wi-Fi network

Use-Case Requirements

Requirement	Metric	Reason
Cost	< $150	Accessible to a variety of end-users
Size/Weight	< 10lbs, 1ft^3	Carryable, easy to transport and use
Detection Rate	> 90%	Catch frequently-transmitting devices
Scan Time	< 5 min	Relatively quick for end-user to scan area
Lateral Accuracy	< 1m	Enough accuracy that user can then search manually within the detection zone
Detection Range	> 10m	Detect hidden devices in a large space
Detection Resolution	< 0.5m	Distinguish devices located near each other

Solution Approach

- Building a product with which a user can walk around a room and it will detect Wi-Fi devices and build a map of their locations
  - Limited to indoors, tracking non-moving devices

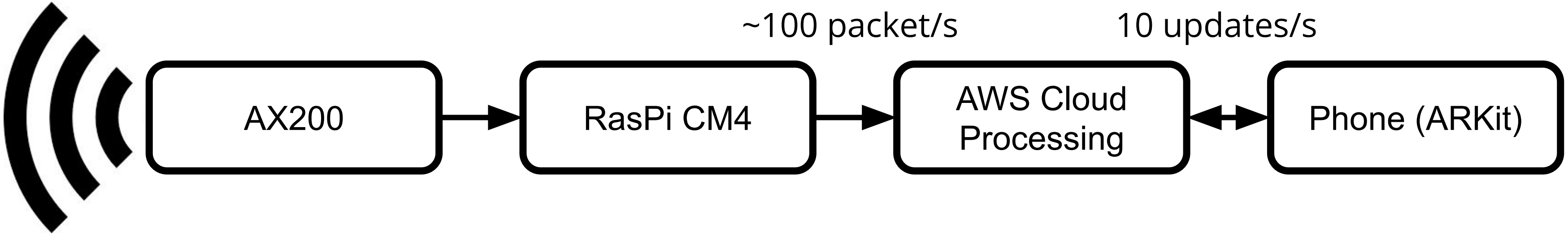
- Show the locations of detected devices to the end-user in an Augmented Reality visualization
  - Using the same cameras for self-localization and AR, hence there is no offset between visualization and tracking origin
  - User can then find devices manually within the ~1m detection zone

System Design - Components

- Device detection
    - Passively-sniffing Wi-Fi packets to find devices
- Device pinging
    - Sending 802.11 pings to devices to measure ToF/RSSI/CSI
- Device location tracking
    - Grid histogram filter to estimate location of devices from ToF/RSSI/CSI
- Self-localization + AR visualization
    - Show user the relative location of detected devices

Design - Physical Layer

- AX200 WiFi Chip for receiving
  - PicoScenes to recover ToF, RSSI, CSI
- Directional + omnidirectional antennas
- Raspberry Pi for control and data uplink
  - Interfaces with AX200 WiFi card
- Phone for Localization and AR Visualization

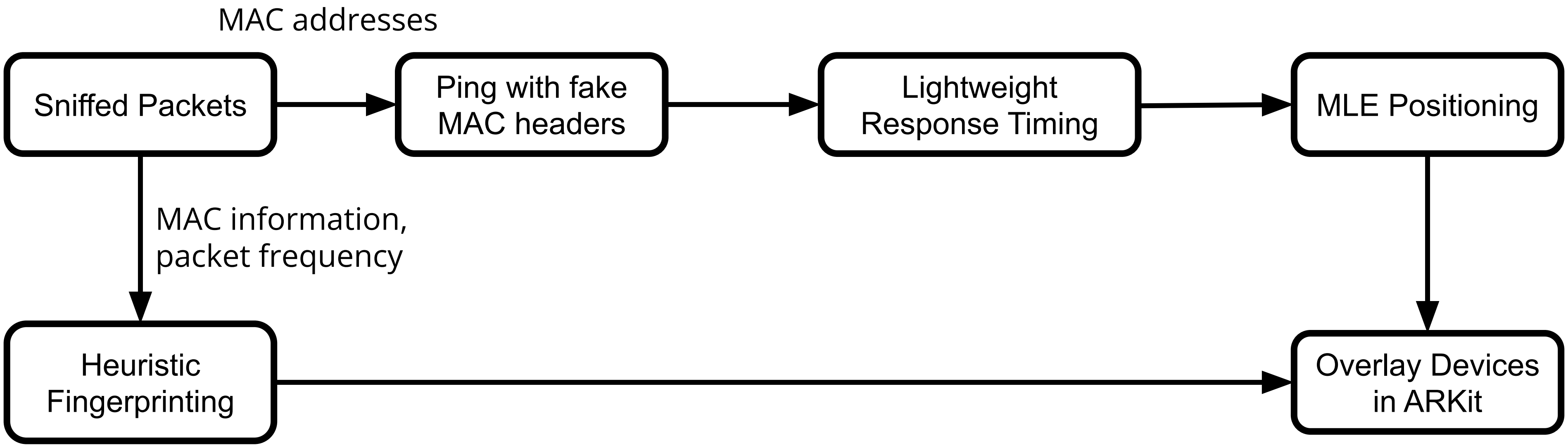


~100 packet/s

10 updates/s

AX200 → RasPi CM4 → AWS Cloud Processing ↔ Phone (ARKit)

Design - Algorithmic Layer


MAC addresses

Sniffed Packets

Ping with fake MAC headers

Lightweight Response Timing

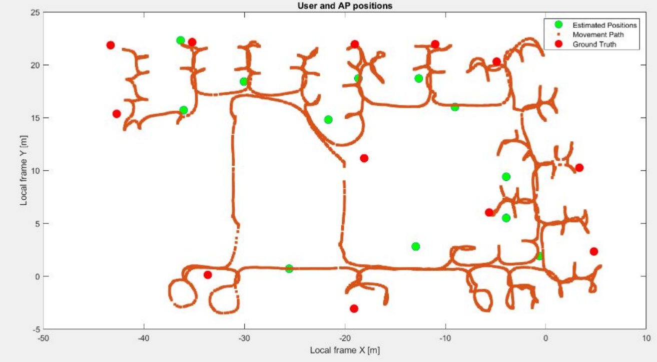
MLE Positioning

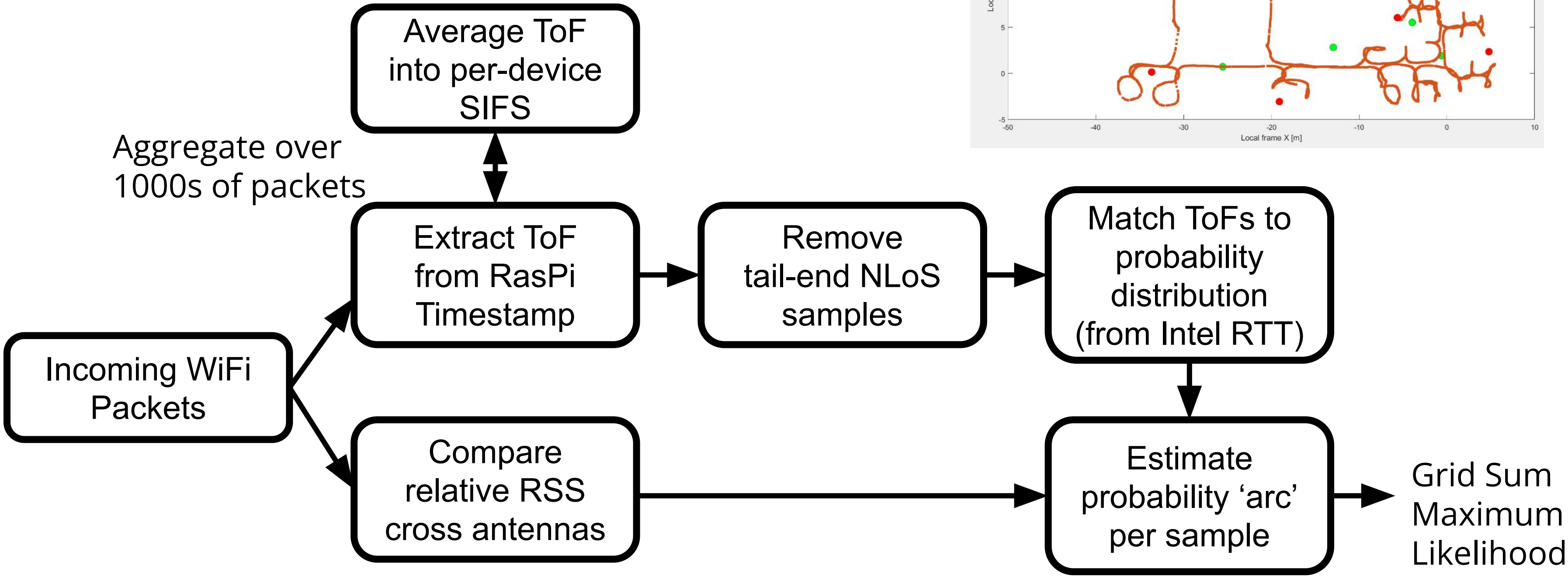
MAC information, packet frequency

Heuristic Fingerprinting

Overlay Devices in ARKit

Signals / Localization System





Average ToF into per-device SIFS

Aggregate over 1000s of packets

Incoming WiFi Packets

Extract ToF from RasPi Timestamp

Remove tail-end NLoS samples

Match ToFs to probability distribution (from Intel RTT)

Compare relative RSS cross antennas

Estimate probability ‘arc’ per sample

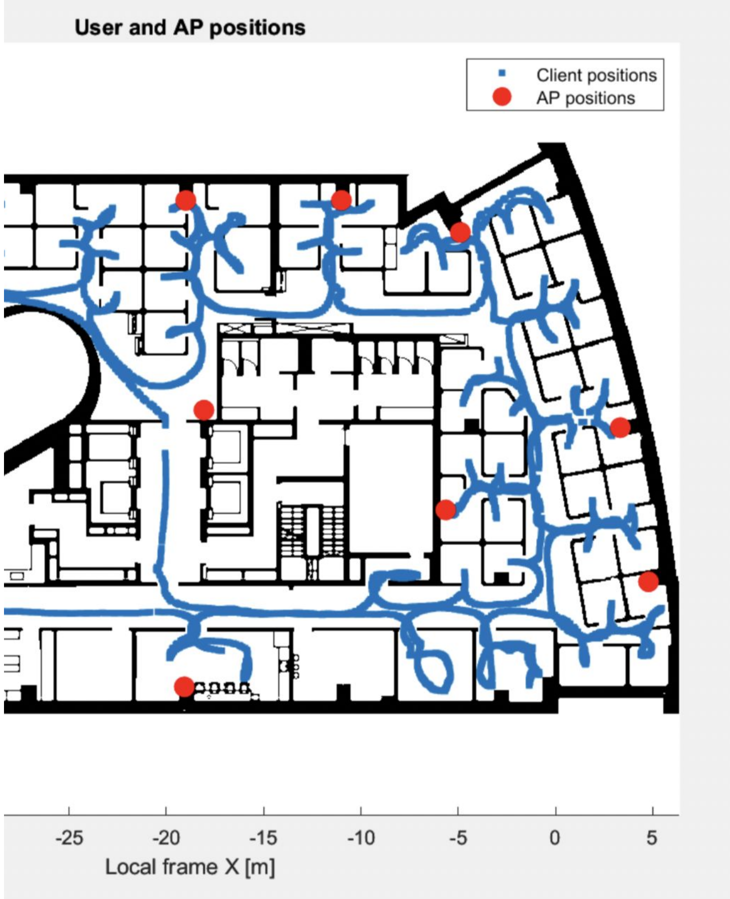
Grid Sum Maximum Likelihood

Implementation Plan

- Wireless data collection: AX200 Wi-Fi card + RasPi CM4 + PicoScenes
  - Off-the-shelf hardware / open-source software
  - Custom mechanical assembly of antenna array
- Wireless packet construction: Python w/ Scapy library
  - Custom code to craft packets + Linux network libraries to transmit them
- Localization/filtering algorithms: MATLAB scripts
  - Custom implementation of algorithms to detect and track devices
- Self-Localization + AR Visualization: ARKit on mobile phone
  - Will develop an interface using ARKit/WebXR to handle visualization

Testing Procedure

- De-risking: Manually sanity-testing each component before integration
  - Allows us to find workarounds to things that don't work as expected
- Set up testbed in a room, try various different antenna movement patterns
  - Zig-zag through the room
  - Spiraling out from center of room
  - Rotating in-place at different points



Integration Testing

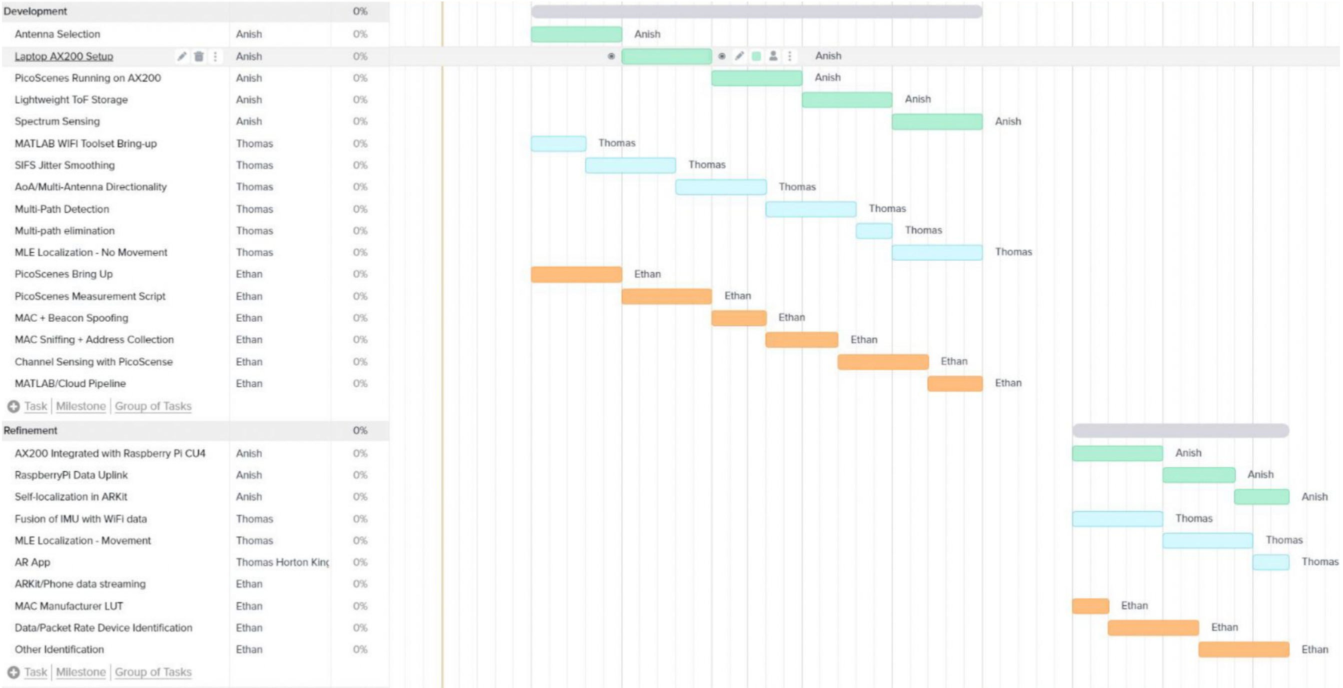
- Set up testbed with a variety of Wi-Fi devices
  - Different channels and data rates to test detection accuracy and range
  - Different kinds of devices: cameras, microphones, sensors, etc.
- Set an origin point, measure ground-truth location of each device
- Capture a variety of test traces (3min, 5min, 10min long traces)
  - Verify positional accuracy against ground-truth
  - Verify percentage of devices detected (and time taken to detect)
- Visually test that AR locations match the physical coordinates

Project Management

Anish - Antenna & sensing bringup, AR localization & visualization

Thomas - Tracking/filtering algorithm, Multipath/noise handling, Device localization

Ethan - Software bringup, MAC spoofing, Device metadata collection



Development		0%
Antenna Selection	Anish	0%
Laptop AX200 Setup	Anish	0%
PicoScenes Running on AX200	Anish	0%
Lightweight ToF Storage	Anish	0%
Spectrum Sensing	Anish	0%
MATLAB WIFI Toolset Bring-up	Thomas	0%
SIFS Jitter Smoothing	Thomas	0%
AoA/Multi-Antenna Directionality	Thomas	0%
Multi-Path Detection	Thomas	0%
Multi-path elimination	Thomas	0%
MLE Localization - No Movement	Thomas	0%
PicoScenes Bring Up	Ethan	0%
PicoScenes Measurement Script	Ethan	0%
MAC + Beacon Spoofing	Ethan	0%
MAC Sniffing + Address Collection	Ethan	0%
Channel Sensing with PicoScense	Ethan	0%
MATLAB/Cloud Pipeline	Ethan	0%
Task | Milestone | Group of Tasks		
Refinement		0%
AX200 Integrated with Raspberry Pi CU4	Anish	0%
RaspberryPi Data Uplink	Anish	0%
Self-localization in ARKit	Anish	0%
Fusion of IMU with WiFi data	Thomas	0%
MLE Localization - Movement	Thomas	0%
AR App	Thomas Horton King	0%
ARKit/Phone data streaming	Ethan	0%
MAC Manufacturer LUT	Ethan	0%
Data/Packet Rate Device Identification	Ethan	0%
Other Identification	Ethan	0%
Task | Milestone | Group of Tasks		