Characterization and Application of Android WiFi-RTT API

Team Members: Yi Song, Yifan Zhang

Overview

- Introduction
- Background
- Android WiFi RTT API
- Hardware and Test Scenarios
- Data Analysis
- Application
- Demo
- Q&A

Introduction

Use **Google WiFi RTT API** and **Google AP/WILD AP** to :

Goal 1: Characterizing WiFi RTT precision across different platforms and scenarios.

Goal 2: Provide a Proof-of-Concept application for WiFi RTT-based car localization in parking lots

Some facts and backgrounds

- WiFi Fine Time Measurement (FTM) protocol introduced in IEEE 802.11-2016 (REV 802.11mc)
- WiFi FTM protocol is based on time-of-flight
- WiFi Certified Location™ was introduced in 2017
- Google introduced an Android API that support WiFi Round-Trip-Time(RTT) on I/O 2018
- Most initial verification experiments online are based on customized open platform
- There are few devices/APs support that protocol, we used Google's Pixel Phone as devices, Google WiFi and WILD AP as the access points in our project
- Mainly Qualcomm and Intel WiFi chips are used in the WiFi FTM supported devices/APs

Technical backgrounds - WiFi RTT

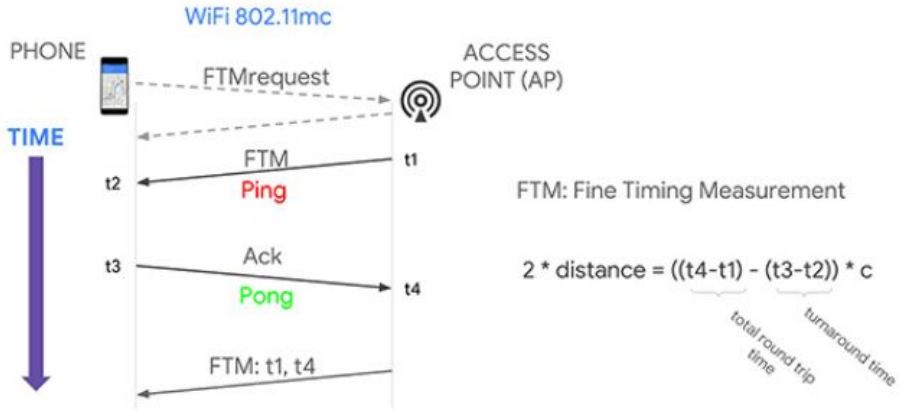


Figure retrieved from Frank van Diggelen, Roy Want and Wei Wang

- Device not Sync.
- STA in ASAP mode
- Multiple FTMs per burst
- $$RTT = \frac{1}{n}(\sum_{k=1}^{n} t_4(k) - \sum_{k=1}^{n} t_1(k)) - \frac{1}{n}(\sum_{k=1}^{n} t_3(k) - \sum_{k=1}^{n} t_2(k))$$

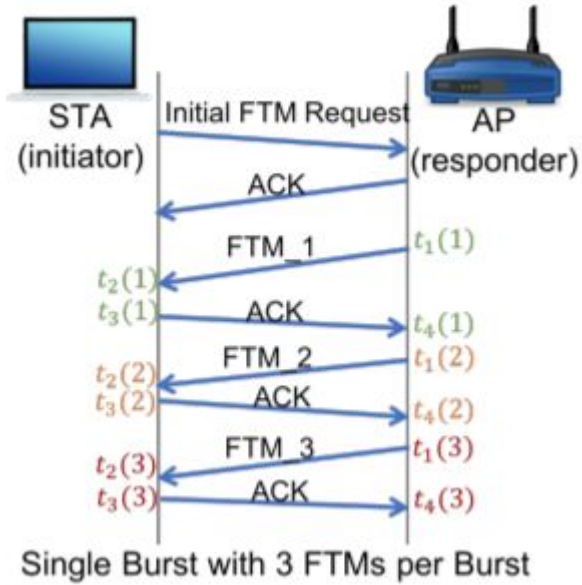


Figure retrieved from M. Ibrahim, et. al, Verification: Accuracy Evaluation of WiFi Fine Time Measurements on an Open Platform

Android RTT API

Procedures:

1. Search for RTT-capable APs (based on WiFi beacons information elements)
2. Make FTM requests
3. Receive of FTM packets and send acknowledgment (upon request approval) (~8 FTMs per burst)
4. Receive of all the timestamps from AP and calculate distance

Features:

- Means and Variances of all FTMs are retrievable
- The API is included in the Fused Location Provider for accurate indoor localization in the future
- Down to one-meter accuracy

Using the API

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
mLocationPermissionApproved =
        ActivityCompat.checkSelfPermission(this, permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED;
```

```
if (scanResult.is80211mcResponder()) {
    newList.add(scanResult);
}
```

- Need to declare *ACCESS_FINE_LOCATION* in manifest file
- Can check if device support RTT using *FEATURE_WIFI_RTT* feature
- Can check if an AP support RTT by calling *is80211mcRepsonder()*
- System service *WIFI_RTT_RANGING_SERVICE* need to be granted permission for using the *WiFiRTTManager* in Android
- RTT result could fail

Hardwares

Device:

Pixel 2 running Android P

SoC: Qualcomm Snapdragon 835

AP:

Google WiFi with latest firmware update

SoC: Qualcomm IPQ4019 Wave 2

AP:

WILD AP

Wireless Chip: Intel 8260AC

CPU: Intel Celeron J3455

https://www.wi-fi.org/news-events/newsroom/wi-fi-certified-location-brings-wi-fi-indoor-positioning-capabilities

Setups

To power the APs: Car 12V supply -> 12V to 110V AC adapter -> Power Outlets/Extension Cord -> APs

Use tape measure to locate the APs and verify location

Tested Scenarios

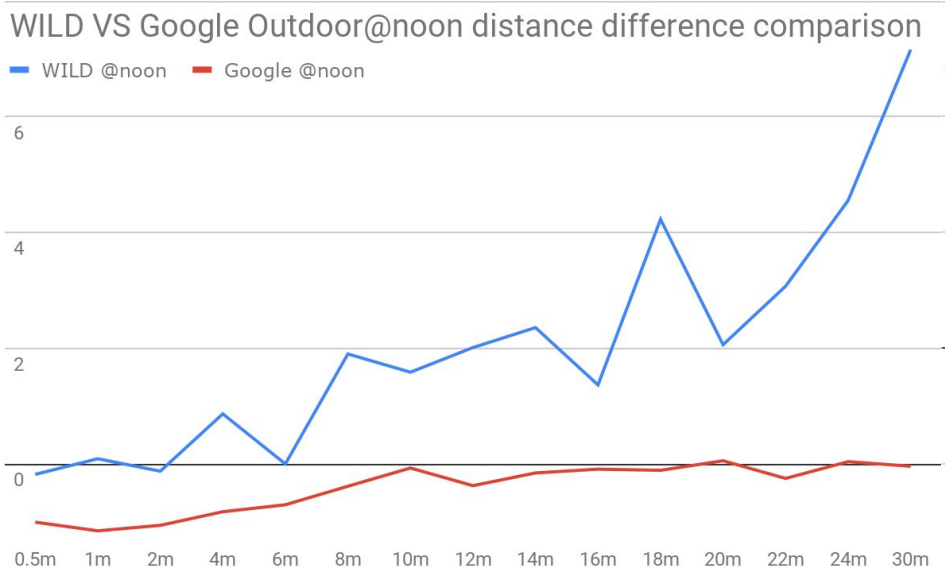
Ranging: (tested for both Google and WILD)

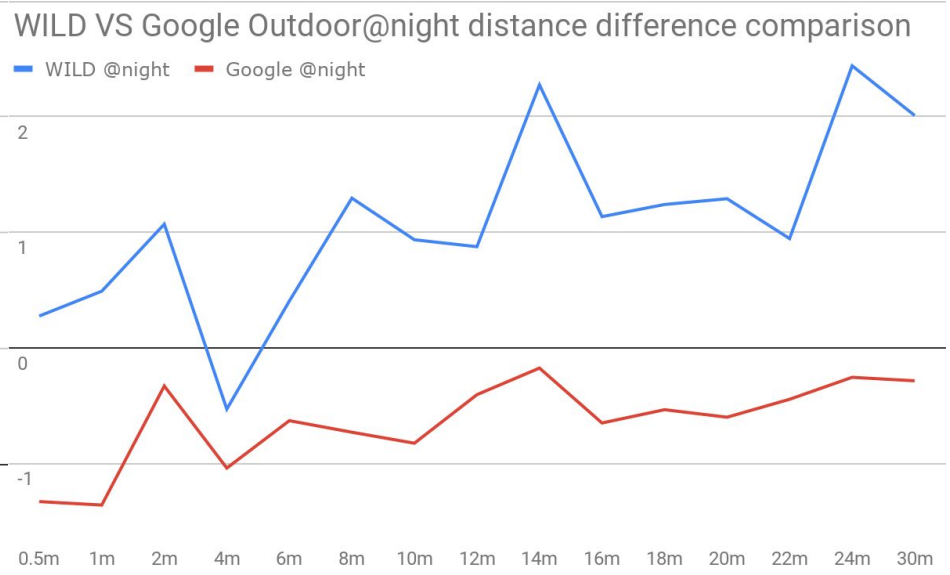
- Temperature: Outdoor @ noon vs. Outdoor @ night
- Occupancy: Indoor w/ low occupancy vs. Indoor w/ high occupancy
- Obstacles: In-room condition, tested in apartment
- 8 Antenna Orientation: tested in outdoor low occupancy condition
- Concrete Obstacle: tested vertically in parking structure

Localization:

- Tested in indoor, low occupancy, for both WILD and Google

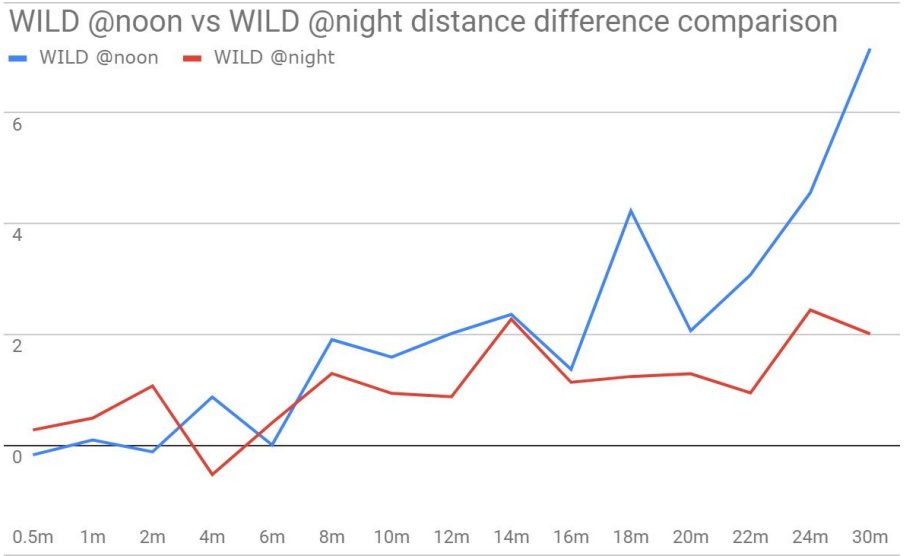
Data Analysis - Temperature

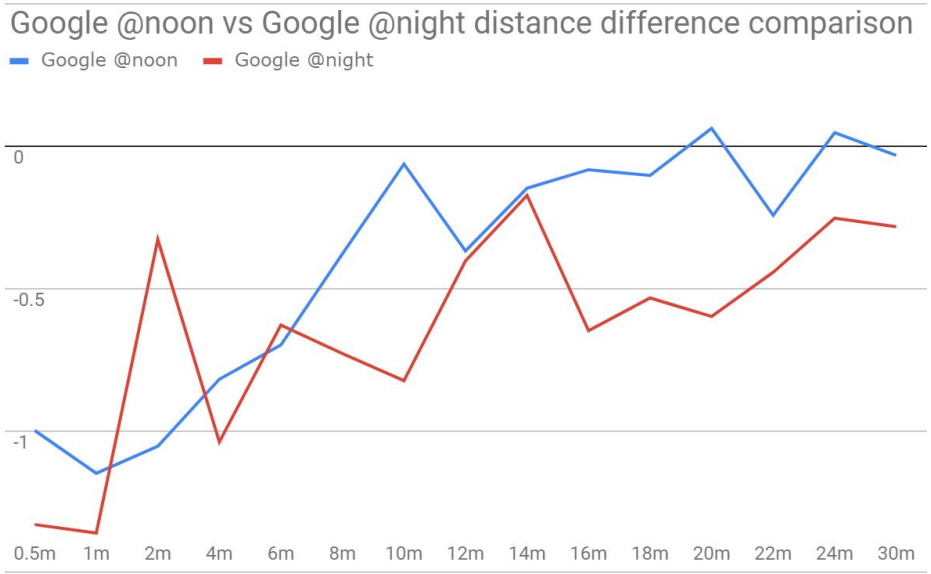




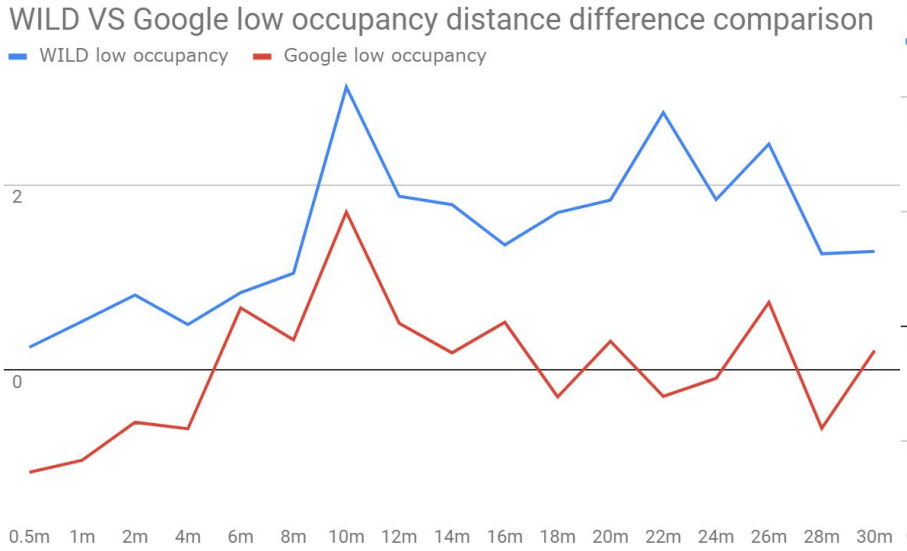
Noon: 17.5℃ (63.5℉) Night:9℃ (48.2℉)

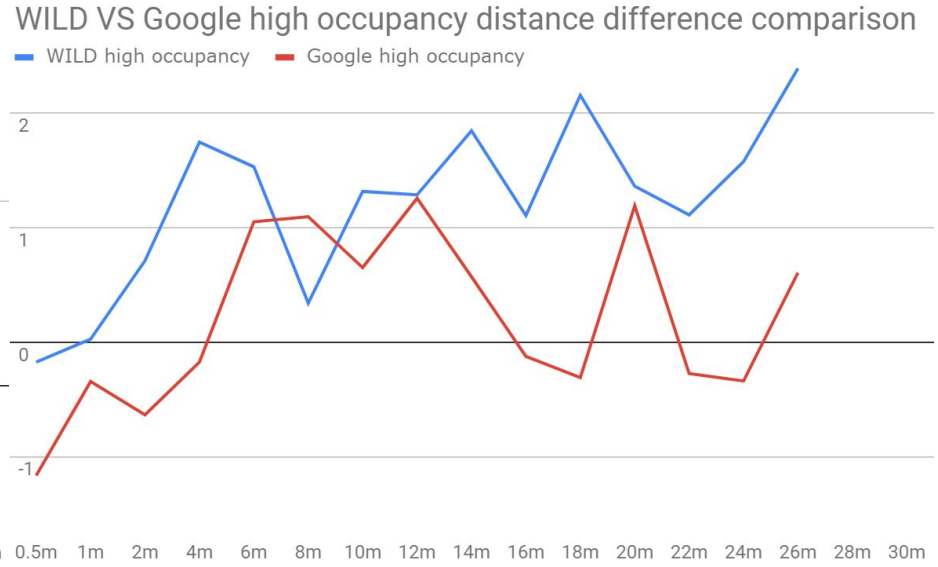
Data Analysis - Temperature



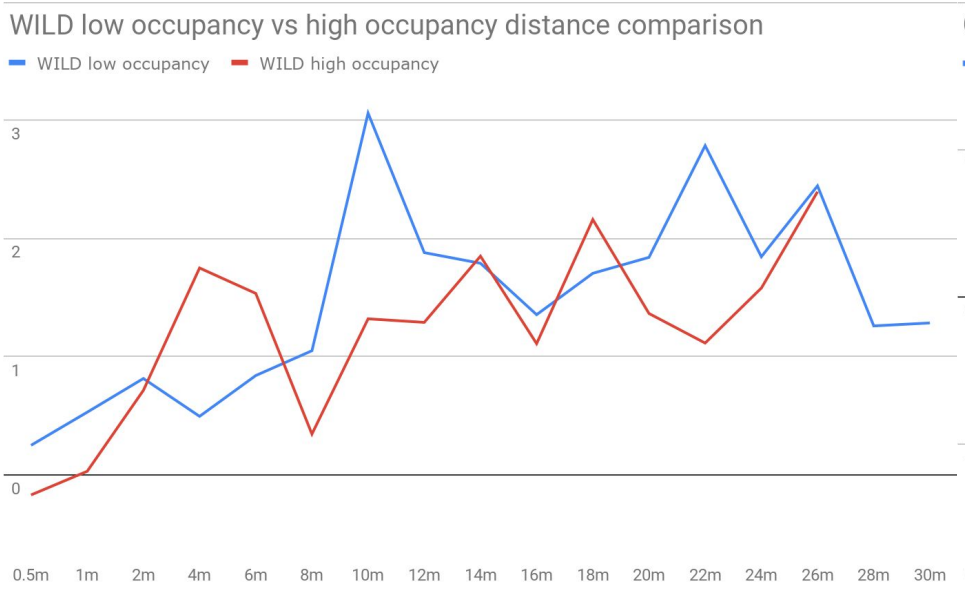


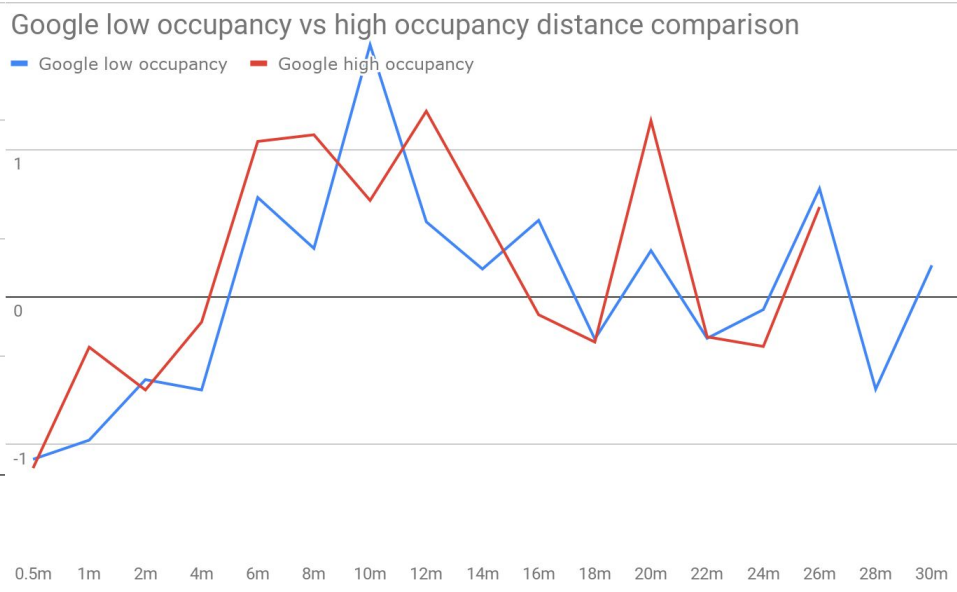
Data Analysis - Occupancy



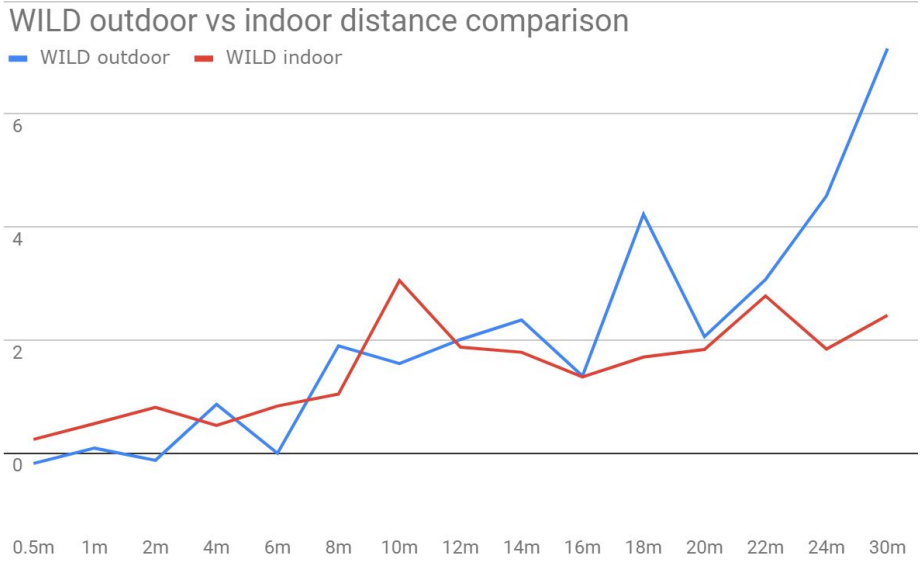


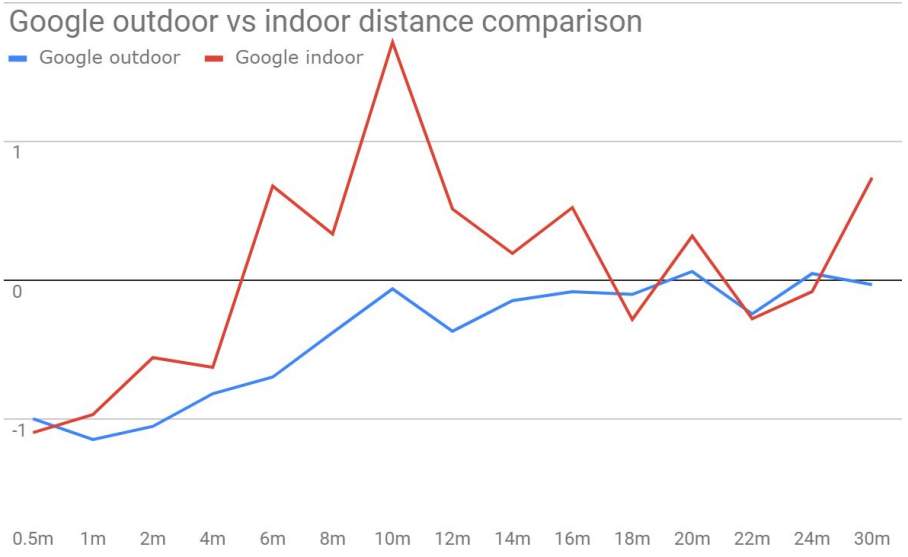
Data Analysis - Occupancy



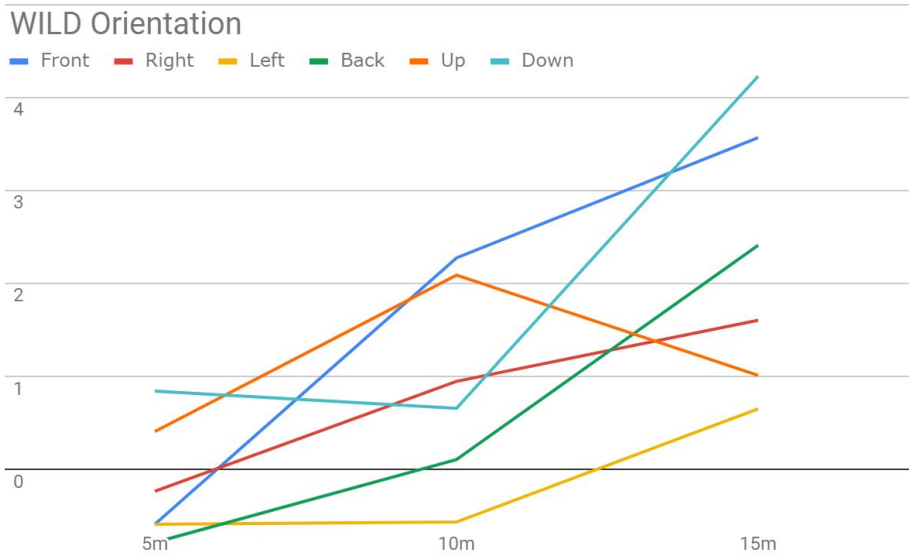


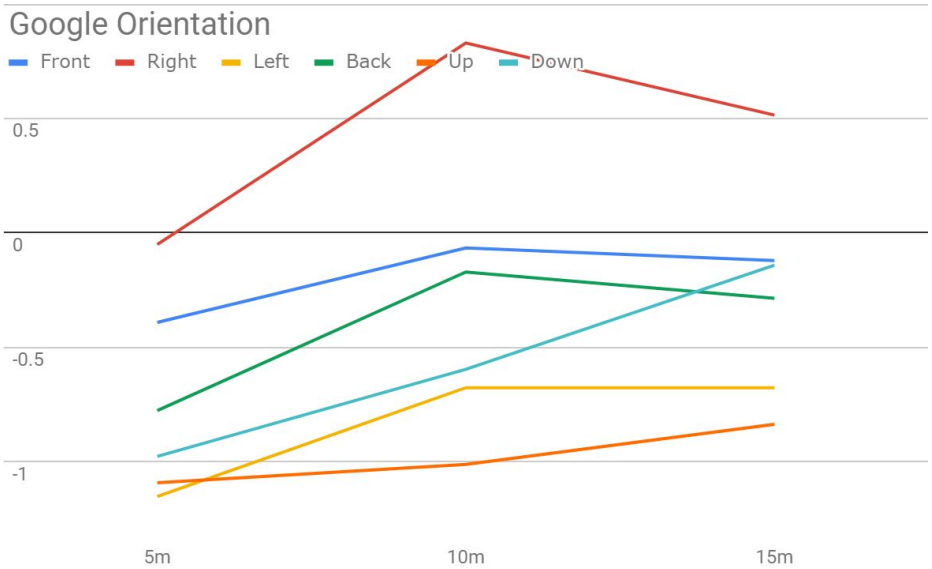
Data Analysis - Outdoor/Indoor



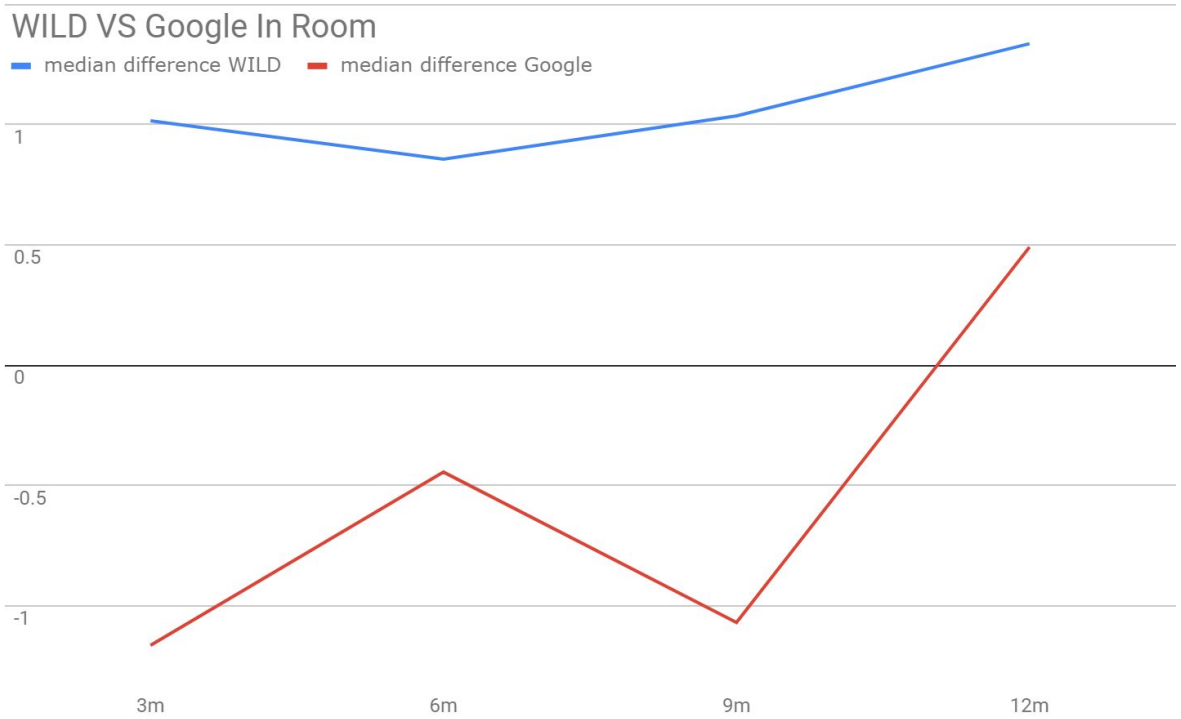


Data Analysis - Antenna Orientation





Data Analysis - In-Room



Data Analysis - Concrete Obstacles

Distance median difference	first floor = 3.3m	stdev	second floor = 3.3+2.95 = 6.25m	stdev
WILD	1.44	0.92	4.065	0.1
Google	-0.425	0.72	0.235	0.19

Data Analysis - Google Localization

Actual Location	Estimated x	Stdev x	Estimated y	Stdev y	distance difference
(10,10)	9.18	0.19	9.22	0.28	1.13
(20,10)	20.63	0.087	7.9	0.4	2.19
(15,20)	13.71	17.74	20.91	15.19	1.79
(15,15)	15.65	0.19	15.3	0.11	0.71
(15,5)	15.34	0.6	6.32	4.35	1.36
(20,15)	21.22	0.04	14.21	0.16	1.45
(15,25)	19.33	2.28	26.955	1.15	4.75
(15,30)	17.68	0.43	30.05	0.28	2.68

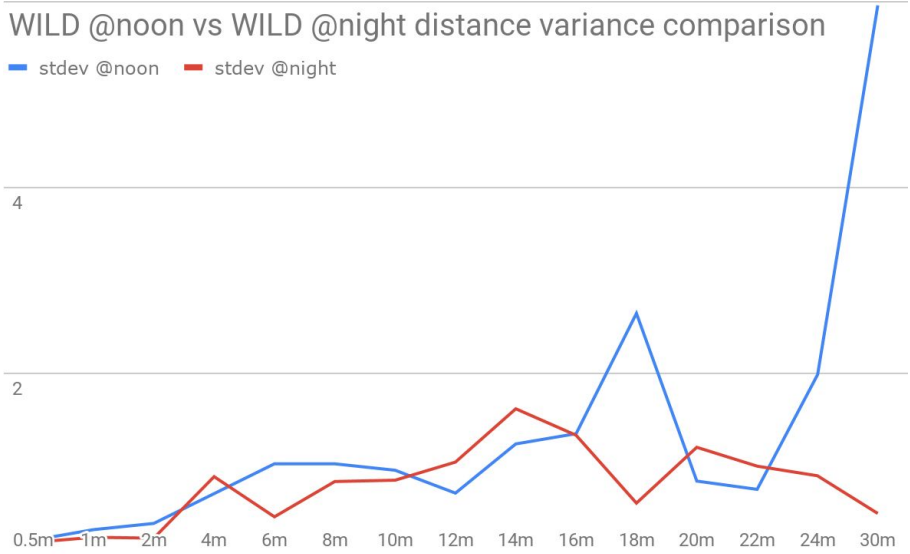
Average distance difference: 1.85

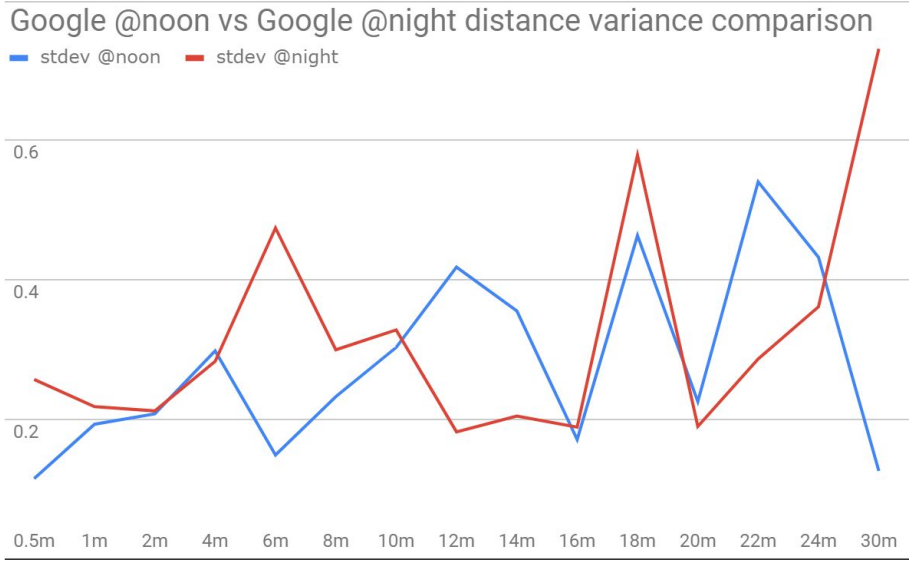
Data Analysis - WILD Localization

Actual Location	Estimated x	Stdev x	Estimated y	Stdev y	distance difference
(10,10)	9.21	2.65	11.57	0.8	1.75
(20,10)	21.18	1.55	13.48	1.28	3.67
(15,20)	15.17	5.12	21.04	2.16	1.05
(15,15)	16.5	2.38	18.17	0.38	3.5
(15,5)	15.51	0.38	6.82	0.71	1.89
(20,15)	22.71	1.06	17.57	2.31	3.73
(15,25)	14.41	5.57	26.21	1.46	1.34
(15,30)	13.05	1.8	26.79	0.94	3.75

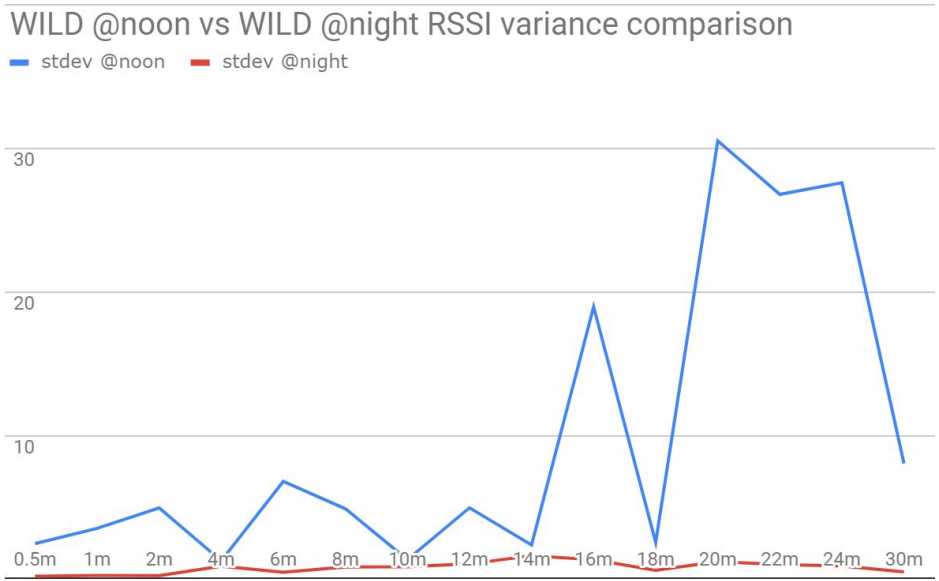
Average distance difference: 2.585

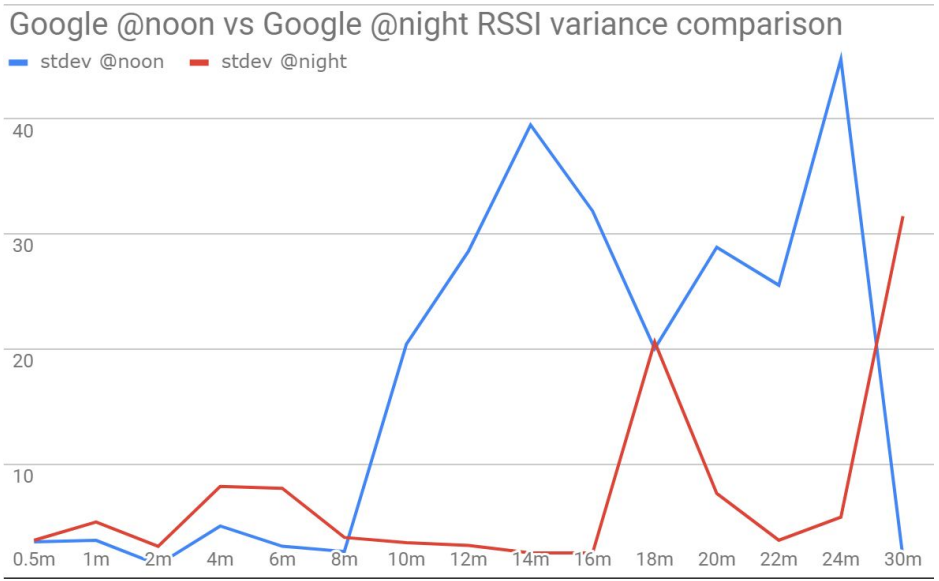
Data Analysis - Distance stdev





Data Analysis - RSSI stdev





Application - Findurcar

Motivations:

- In a large scale parking lot, it is difficult to memorize the location of parked car and people are having trouble finding their cars.
- GPS signals won't work when the parking structure is closed rooftop.
- Relative Accurate localization method needed to provide this service.
- Current solutions (such as Westfield, Santa Monica Parking Structure used cameras over all the parking spots) require lots of modifications to the parking structure and still failed to give directions based on user position.

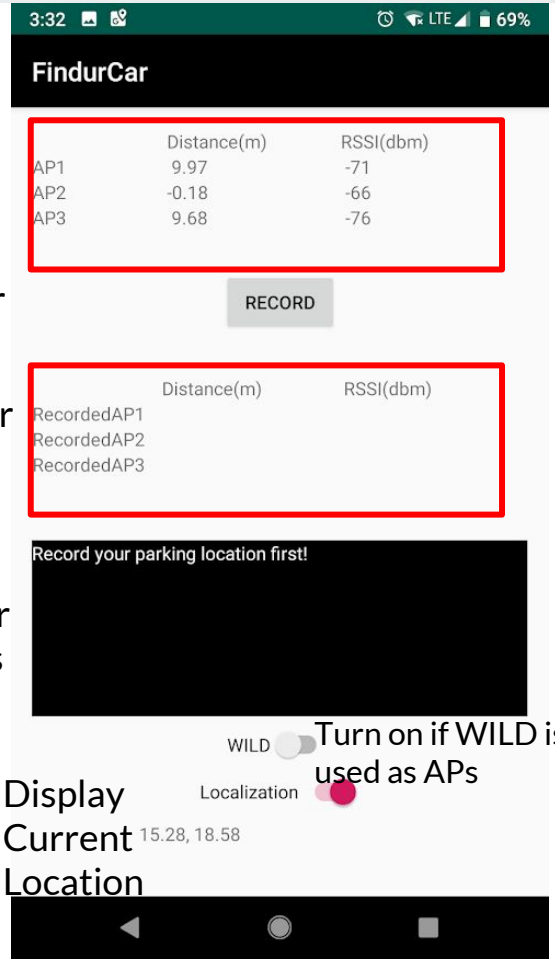
Our App Features:

- Requires minimal changes to the parking lots
- Gives real-time directions as user walks
- High Accuracy

Our PoC Application

Basics:

- Findurcar can display and record locations of your car when all three Google APs are presented
- Findurcar will display the direction to find your car based on your orientation (`Sensor.TYPE_ROTATION_VECTOR` used)
- Directions will include the distance and orientation of your car



Display Current Location

Update Car Location

Display Car Location

Main Display for Directions

Turn on if WILD is used as APs

Display Current Location

Localization Algorithm

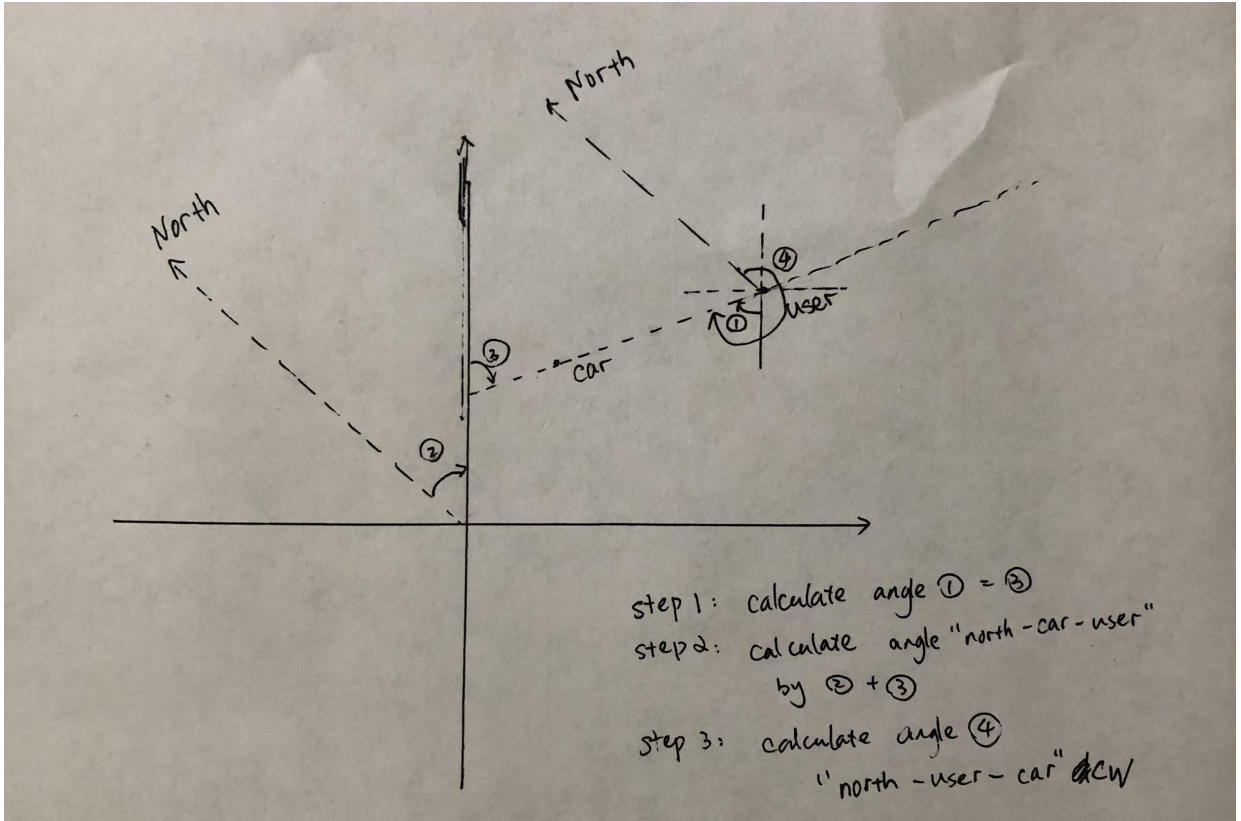
- 3 Google APs @ (10,10), (15,20), (20,10):

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$
$$(x - x_2)^2 + (y - y_2)^2 = r_2^2$$
$$(x - x_3)^2 + (y - y_3)^2 = r_3^2$$

- 2 WILD APs @ (0,10), (0,20):

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$
$$(x - x_2)^2 + (y - y_2)^2 = r_2^2$$

Orientation Algorithm



Show Time!

Future Work

Theoretical:

- Improve localization precision by introducing sensor fusion
- Develop different algorithms to optimize the precision
- Collect more data with more accurate environment setup

Findurcar App:

- Improve UI
- Add user-friendly functions like auto-recording
- Fuse the direction in a map view for given parking lot

Q&A