Location Tracking and Mapping

SOFE 4790U Distributed Systems
Dr. Qusay Mahmoud
Project Group 7

Marwa Safa 100585192
Christopher Phan 100620028
Walid Safi 100623815
Curtis Whall 100655399

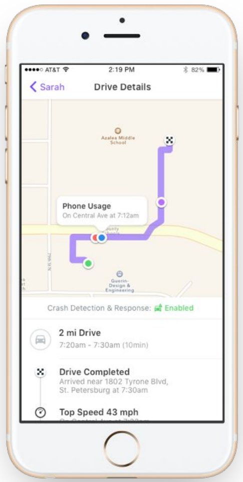
Project Objective

- Develop a distributed location tracking system
- The system should track users real-time location
- Users will be able to see the locations of the devices that are connected
- This system will allow users (Friends, and family members) to safely check on the location of specific users

Existing Solutions

There are many other location tracking, and system Tracking Applications

- **Life360**
  - Application to track users and provide device monitoring and notifications
- **Google Maps**
  - Web-based Service that provides information about the area, does not include location sharing
- **mSpy**
  - Mobile and computer parental control application to track applications activity


AT&T 2:19 PM 82%

Sarah Drive Details

Phone Usage
On Central Ave at 7:12am

Crash Detection & Response: Enabled

2 mi Drive
7:20am - 7:30am (10min)

Drive Completed
Arrived near 1802 Tyrone Blvd, St. Petersburg at 7:30am

Top Speed 43 mph

Existing Solutions Cont'd

- **Famisafe**
  - Another Parental control mobile application that enables locations tracking and device limitations
- **Find my Phone**
  - Exclusive IOS phone tracking device
- Majority of the current solutions are paid services

Image 2: FindmyPhone

Proposed Solution

- Location Tracking and Mapping Mobile System
- Use Mobile Devices as Clients
- Backend Server to handle Client requests and incoming data
- Track current/previously connected clients

Technologies Used

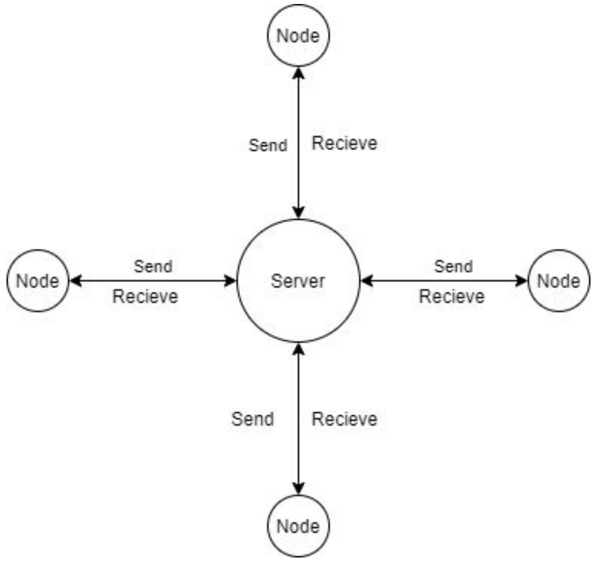
- Android Studio
  - Android Studio protocols (Location manager, Mapping Activities)
- Java based Server
- Firebase Database
- Google maps API

Design

- Front end/ Client side was developed using Android Studio
  - Multithreaded Client to continuously ping the server for updates
  - Google Maps API to create custom markers based on Longitude and Latitude
- Backend
  - Multi-threaded server responsible for scheduling and distributing data to clients
- TCP/IP Network Sockets (Object DataStream)
- Firebase Database

Centralized System Architecture

- Central Server responsible for servicing multiple client, Consists of three main components
  - Server (Master)
  - Nodes (Android Mobile Devices)
  - Communication Link (TCP Network Sockets)
- Clients do not interact with other clients, the Server will handle and Co-ordinate messaging
- All Clients will receive synced data



Pinging

- Benefits:
  - Cheap to implement
  - Concurrently runs in the background without affecting the system
  - Sent as a non-stopping in a time period and won't shut down compared to a normal ping (fault tolerance)
- Implementation:
  - We used heartbeat pinging on a multithreaded system. (Unique heartbeat for each client)
  - Using concurrency in order to ping
  - Pings are set at 10 sec intervals for this project (Testing Purposes)
  - Ideally Pings will be set as a longer interval to reduce Centralized System bottlenecking when accessing data from the server

Scheduling

- Server Receiving data must input the data into its local storage
  - Multiple Client requesting to input data may lead to corruption
- Need to avoid Race Conditions with multiple clients
- Implemented Locks using a semaphore to lock the critical section
  - Allow only one client thread to input data/ call inputData function on the server
  - Release Lock after function is completed
- Fault Tolerance Measure

Replication

- Clients are able to run concurrently and not dependant on the server
    - Server crashes are to be hidden from the user
- Clients will create a local copy of the data received from the Server every time it pings
- Clients will use the local copy to map the different users even after Server crashes
- Once the Server is running again the client updates their local copy
- Increases Fault Tolerance
- Failure-Transparency, Replication Transparency

Project Demo

Challenges and Solutions

Challenges	Solutions
Creating a Socket Connection on android Studio was challenging as a connection is not allowed to be created on the main thread	Utilized multithreading android to create a connection whenever needed
Creating a Heartbeat and refreshing the map module every ping	Unable to solve this problem as we are unable to call the mapping function outside of the main process. Worked around by using a button to manually refresh
Connectivity over a network was difficult implement using sockets	Port Forwarded and opened ports on the router and system, the opened ports and ip were able to work on the hosts system

Evaluation of out System

Future Work

- Create a broker to handle scalability (Multiple Servers dedicated to a group of devices) and security (Group of devices connect to a server using a Unique ID)
- Use Firebase Database to provide the application with real-time data analysis

- We were able to develop a functioning Location Tracking and mapping system
- Apply what we learned in class to increase usability, reliability, availability, etc.

Q&A