Emergency Vehicle Early Warning System

- **Background**
- **Our Approach**
- **Our Models**
  - Client/Server - Cellular
  - Peer to Peer - RF
- **Benchmarks**
- **Conclusion & Future Works**
- **Demo**
Nearly 300,000 people die each year to cardiac arrest
 Nearly 300,000 people die each year to cardiac arrest

30% could be saved by faster response times
30,000 serious accidents involving fire trucks each year
When an accident happens in a tunnel in South Korea.
Background

- There are too many accidents that happen each year involving emergency vehicles
- Our current alert system (lights and sirens) has seen few advances in the past decades
- With modern wireless technology we should be able to alert drivers before they can even hear or see emergency vehicles
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server - Cellular
  - Peer to Peer - RF
- Benchmarks
- Conclusion & Future Works
- Demo
Our Approach

- Emergency Vehicles broadcast their location, and receiving vehicles can react appropriately
- How to accomplish this?
  - Client / Server - Cellular
  - Peer to Peer - RF
Early Testing - Client / Server (Cellular)

- What does mobile data performance look like?
- How does it compare to WiFi?
- Test Application
- Lead us to three possible models

Early Testing - Peer to Peer (RF)

- Hardware experimentation
Connection Test Application

**Mobile Data**

4G from Malone

Mean: 442 ms
Median: 435 ms

More variability

**WiFi**

hopkins network in Malone

Mean: 300 - 450 ms
Median: 410 ms

More consistent, but more dependant on specific URL
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server - Cellular
  - Peer to Peer - RF
- Server Benchmarks???
- Conclusion
- Demo
Peer to Peer (RF) Method

Prototyped with an Arduino
P2P between cars
Uses smartphone for GPS and mapping of emergency vehicles
Peer to Peer (RF) Overview

- Decentralize our alert system for faster warning propagation
  - Similar to 802.11p, which was designed for inter-car communications
  - P2P, with alerts originating from equipped emergency vehicles

- The emergency vehicle constantly broadcasts to surroundings
  - Easily scalable
  - Very fast communication (~10ms per hop)

- All devices forward any alerts they receive, propagating messages away from their sources
Client/Server (Cellular) Method

Deployed as a smartphone app
Client/Server Model
TCP-based protocol over 4G/LTE
Central Server Alerts Drivers
Client/Server (Cellular) overview

- Maintain a centralized list of users and their current locations
  - Server client model
- Emergency vehicles send their messages to the server which then alerts all users near the vehicle
- Takes advantage of existing infrastructure
  - Rely heavily on 4G/LTE coverage and future innovations to wireless infrastructure to handle communications
- Use Firebase to message all clients instead of server
- Potential to give warnings even earlier than RF
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server
  - RF
- Benchmarks
- Conclusion & Future Works
- Demo
Discarded Strategy: Firebase-Only Approach

- Firebase can have users join “groups”
- We would associate a “group” with a geographical bucket
- All clients join and leave groups as their location changes
- Emergency vehicles send notifications to the nearby groups
- Jeff Dalla Tezza cautioned against this:
  - Firebase is not designed to handle the churn of our users switching groups rapidly
  - Recommends we store Firebase ID’s and directly message devices
Method: Hybrid Server/Firebase

- Clients receive a unique token from Google
- Clients pass our server the token
- Clients periodically update our server with location
- On emergency, we send appropriate tokens to Google, who then manages the notifications to clients
Method: Hybrid Server/Firebase

- **Pros:**
  - Load on our servers is lower
    - Utilize Google’s resources
- **Cons:**
  - Need to store location of all users
  - Additional latency from using Firebase
Method: Server-Only Approach

- Emergency vehicles constantly send the server their location
- Server only stores the locations of emergency vehicles
- Clients send periodically their location, and ask, “is there an emergency in my area?”
- Clients can additionally send older locations for better heuristics on the server
Method: Server-Only Approach

- **Pros**
  - Lowest expected latency from server/client approaches
  - We already round trip connections to the client for each update, so we might as well take advantage of that response
  - Less information to store server side (more scalable!)
Stationary Mobile Packet Loss

Mobile Packet Loss in a Vehicle
Method: Server-Only Approach (Faster?)

- Scaling to millions of cars making millions of TCP connections is impractical
- Car interaction with the server is stateless, side-effect free, and brief
- Replace HTTP with space efficient binary system
- Emergency vehicles still communicate over TCP
- Cars can use a simple UDP with retry protocol to avoid the 3-way handshake
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server - Cellular
  - Peer to Peer - RF
- Benchmarks
- Conclusion & Future Works
- Demo
Peer to Peer (RF) approaches

- Tested with both 433MHz and nRF24l01 (2.5GHz) transmitters
- Found better range and library support for nRF24l01
Peer to Peer (RF) Hardware Details

- nRF24L01 with line of sight has range of about a football field - 120 yards.
  Loss of 5-10%
- Maximum message length of 32 bytes
RF24 Details

- Used the RF24 library by TMRh20
- Library supports reads, writes, multiple channels
- Library supports IP-like addressing & mesh network, but overhead was too high for nodes in motion
Peer to Peer (RF) Protocol

- Emergency vehicles create alerts, send them out via broadcast
- Each created message has an ID and a TTL
- Messages that are received by civilian vehicles are retransmitted with probability $1/(2^n)$, $n =$ times message has been retransmitted
  - In a busy environment, civilian vehicles are expected to retransmit each message 2 times:

$$
\sum_{n=0}^{\infty} \frac{1}{2^n} = 2
$$
Peer to Peer (RF) Protocol

● If the network is busy, transmitters will wait a random amount of time (up to 10ms) before attempting to transmit again.
  ○ Based on Carrier-sense multiple access (CSMA) protocols.
● Older versions displayed message paths, knowledge matrices
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server - Cellular
  - Peer to Peer - RF
- Benchmarks
- Conclusion & Future Works
- Demo
Peer to Peer (RF) Benchmarks

- On average, about 18ms per hop
- First few hops are faster (~10ms) due to lower message traffic
- With android app, much slower due to Serial communication (baud rate = 9600)
Client/Server (Cellular) Implementation Details

- Server running in the basement of Malone right now
  - 4 core Intel Xeon 3.20GHz, 3 gigabytes of RAM
- Android App (Java)
- Python Flask Server (behind uwsgi)/Golang Server
- Redis backend (Manage location data)
- Nginx reverse proxy
Emergency Vehicle Early Warning System

- Background
- Our Approach
- Our Models
  - Client/Server - Cellular
  - Peer to Peer - RF
- Benchmarks
- Conclusion & Future Works
- Demo
Conclusions

- We have a version of our system that works for both sides
- The RF side may need more tuning in the future to work in crowded situations
- The server the app communicates to may eventually need to become distributed to handle increased loads or failures
- Adoption of system is likely to be a large hurdle to overcome
Future Works

- Test in actual cars
- Smarter decisions using location on the server side
- Larger scale RF tests - play with dynamic wait periods, dynamic propagation probabilities, find/develop ways to send longer messages.
- Optimize communication between Android/Arduino, do more computation on Arduino to limit Serial communication
Emergency Vehicle Early Warning System

● Background
● Our Approach
● Our Models
  ○ Client/Server - Cellular
  ○ Peer to Peer - RF
● Benchmarks
● Conclusion & Future works
● Demo
Demo