ENHANCING AUTONOMOUS VEHICLE SCHEDULING AND DISPATCH WITH SEAMLESS WIRELESS SMART PARKING

by

Paul R Seymer
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

________________________	Dr. Duminda Wijesekera, Dissertation Director
________________________	Dr. Cing-Dao Kan, Dissertation Co-Director
________________________	Dr. Zoran Durić, Committee Member
________________________	Dr. Jyh-Ming Lien, Committee Member
________________________	Dr. Sanjeev Setia, Department Chair
________________________	Dr. Kenneth Ball, Dean, Volgenau School of Engineering
Date: ____________________	Summer Semester 2019 George Mason University Fairfax, VA

Enhancing Autonomous Vehicle Scheduling and Dispatch With Seamless Wireless Smart Parking

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Paul R Seymer
Master of Science
George Mason University, 2012
Master of Science
Hood College, 2007
Bachelor of Science
University of Maryland at College Park, 2004

Director: Dr. Duminda Wijesekera, Professor
Department of Computer Science
Co-Director: Dr. Cing-Dao Kan, Professor
College of Science

Summer Semester 2019
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my late mother. Her unconditional confidence and support was and continues to be the foundation I build on.

Acknowledgments

There are several people I would like to acknowledge for either supporting this work or supporting me during my program. First and foremost, I would like to thank my thesis advisor, Dr Duminda Wijesekera for his patience, guidance, insight, and concern for the work product as well as the emotional well-being of students like me. I also want to thank him for helping me realign my program after changing advisors and topics very late into candidacy, and helping to work through all of the challenges that come with such a comprehensive change. Successfully completing a PhD on a part-time basis requires herculean efforts to solve problems both within and outside of the research itself, and having a supportive advisor is absolutely critical to that success. I would also like to thank Dr Cing-Dao (Steve) Kan for working with us to develop the background needs for the work, and for the level of support offered in the way of funding, office and work spaces, and extensive domain expertise. Additionally, I would like to thank Dr Zoran Durić and Dr Jyh-Ming Lien for serving on my committee and providing review and feedback to this dissertation, and Dr Hakan Aydin for his help with polishing my public defense.

I would also like to single out my direct management chain at my employer: Ingrid and Eric for their flexibility accommodating part-time students and their efforts to help me navigate and resolve issues related to our employer's public release and document control processes. Not all managers take the time to understand unique needs like mine and prioritize my personal outcomes. Having support from them was vital at reducing obstacles (and stress levels).

Lastly, no large scale effort like this can be achieved without the emotional support of peers, friends, and fellow graduate students. I would like to thank the following people for providing advice and constant encouragement: Noha, Chaitanya, Bo, Sepideh, Haris, Parnian, Chen, Spiros, Charles, Quan, Sharath, all of the other past students in Research 1, Headley, Darren, Zinoosh, Jeremy, and others for persistent kindness and friendship. Without their help and support I would not have made it through this process.

Table of Contents

List of Tables

List of Figures

Abstract

ENHANCING AUTONOMOUS VEHICLE SCHEDULING AND DISPATCH WITH SEAMLESS WIRELESS SMART PARKING

Paul R Seymer, PhD

George Mason University, 2019

Dissertation Director: Dr. Duminda Wijesekera

Integrating and incentivizing multi-modal mass transit continues to be a challenge for municipalities, particularly in the US. Our *driving culture* influences both the personal choices we make, and the funding priorities for national, state, and local governments. In some cases these priorities do not focus on technology initiatives such as automated vehicles (AV) and last-mile transit. For small, traffic dense areas such as urban centers and college campuses, competition for funding is strong. Additionally, as autonomous vehicles are emerging technology, solutions for facilitation and other use cases are currently a work in progress.

Any use of AVs and multi-modal transit must exist in tandem with or otherwise acknowledging person-vehicle driving culture. To that end, any solution involving AVs should supplement existing personal vehicle use, and incentivize the use of mass transit in the process. In this dissertation, I develop a seamless, wireless, Bluetooth-based smart parking localization solution and couple its outputs with an AV scheduling and tasking system to enable integrated multi-modal *last-mile* travel.

Chapter 1: Introduction

US urban centers continue to see low public transportation usage, often declining in most major cities (and particularly for buses [1]). There are several factors that could be the cause of this decline; such as an increase of the average transit fare at a rate higher than inflation or the recent decrease in average annualized gasoline prices [2] that may motivate would-be mass transit users to drive their own vehicles [3]. In addition, fixed bus routes have many stops causing longer transit times than in modes such as ride hailing services, taxis, or personal vehicles that do not have numerous mandatory stops. Buses themselves can become over-crowed, preventing passengers who may have already experienced a long wait time to be unable to board the bus at all. As there is no dynamic bidirectional information exchange between contemporary mass transit systems and awaiting passengers, uninformed delays and failures further frustrate users and cause them to use ride-hailing services or personal vehicles [4]. In addition, mass transit infrastructures are cost burdens for metropolises without convincing proportional cost justifications. Although mass transit stops (such as bus stops) could be infused with customer supporting technology, current spending trends appear to be limited to passenger seating, shelters and real-time train/bus information that carries a significant cost [5]. The response from the public to these factors are minimal and still favor privately owned vehicles (PoVs) that exacerbate traffic congestion during peak hours and parking solutions at commuter destinations. Some use ride-sharing services like Uber [6] to fill the gap between PoV use and public transportation which some research shows helps accelerate further declines in mass transit use [7].

Encouraging a population to replace personal vehicles with mass transit solutions continues to be a challenge, particularly in suburban spaces where *first mile* mechanisms do not exist. Autonomous vehicles (AVs) are hailed as the solution to this, and many contemporary urban transportation problems by providing flexible transportation options [8]. However, public sentiment is not as optimistic, suggesting that AVs will only make streets more congested and unsafe [9]. Without the interest and confidence from the public, deployed AVs may go unused. If not deployed effectively, AVs may fail to improve travel scenarios and create additional traffic. Cities have began pilot initiatives [10] to ensure that AVs reach their fullest potential by encouraging safe, effective, and sustainable design of AV solutions that meet their needs. Washington DC in particular, created an inter-agency working group [11] to establish and drive a set of design and deployment principles for AVs. These principles include Safety, Efficiency, Equity, and Sustainability [12]. Any solution using AVs in dense areas such as the greater Washington, DC area must embody these initiatives and collaborate with this working group's goals. These principles align with the notional that AVs should complement and integrate with mass transit. One aspect of ride-sharing that is attractive to people is the customized, on-demand nature with minimal intermediate stops between entering and leaving the vehicle [13]. AV deployments must emulate this on-demand feature to be attractive to travelers, but with a minimal amount of time the vehicle spends on the road, thereby reducing traffic congestion. To further reduce congestion, AVs must leverage and work in harmony with existing off-road mass transit solutions such as the underground Metro. In this way, an efficient multi-modal solution can be created, without replacing existing mass-transit sources while reducing overall urban vehicular traffic.

One of the most significant problems with multi-modal transportation is the burden placed on passengers to coordinate between and handle payments with the different modes. Municipal Transit Authorities, such as the Washington Metropolitan Area Transit Authority (WMATA) have mitigated a portion of the payment problem in the parking/bus/Metro use case by allowing a single electronic smartcards [14] to be used, although the coordination burden remains. Passengers are forced to perform this coordination on their own and wait at each modal transfer point. The use of these cards often create bottlenecks when passengers scan them at station ingress and egress points. Innovation with new technology must also work to reduce these bottlenecks. Additionally, passenger coordination should be removed or reduced, allowing users to focus more on their destinations, instead of the multiple paths along them. Apps such as the Trip Planner offered by WMATA [15] reduce this burden. However any changes to the travel environment such as missed transfers, traffic and other delays that develop en-route, etc., may render the pre-planned trip unusable. Thus, trip planning must be seamless to the traveler, and must adaptable to these failure and delay conditions.

As availability of consumer-grade electric vehicles and initiatives in public transportation to become more efficient and environmentally sound, autonomous vehicles emerged in these use cases. A significant constraint with multiple-occupancy electric AVs, however, is their limited duty cycles (compared to PoVs) due to short battery lifetimes requiring more than an hour to recharge per tens of miles for many multi-passenger vehicles [16]. Without careful scheduling, this could lead to queues at charging stations or vehicle unavailability during peak times. Additionally, vehicle tasking and route schedules must be constructed to ensure the minimal on-board passenger density is maintained while every charge cycle is spent transporting a maximum number of passengers.

Lastly, cultural influence and similar elements continue to encourage the maintenance of personal vehicles. Therefore, any shift towards the use of mass transit must first overcome or coexist with these notions. The work outlined in this thesis opts for the latter option, by creating a smart parking solution that integrates with and drives our autonomous vehicle customized tasking and dispatch system to create a unified multi-modal transportation option travelers can use without giving up the use of their personal vehicles.

This aspect too, has its own set of unique problems that must be solved in any collaborative system. Contemporary public parking lots use controlled access ingress and egress points to facilitate ticketing and payment. Parking lots with high traffic volumes often form bottlenecks at these ingress and egress points, particularly during rush hour or the beginning or end of nearby events. If parking lots were free to limit the use of these ticketing systems or eliminate payment infrastructure entirely, occupants could simply drive their car into the lot, park, and leave the lot without the time expense of coordinating payment and validation. The result would be a seamless parking experience with higher occupancy turnover. Additionally, an ongoing challenge in developing seamless parking experiences is the detection and identification of vehicles in parking spaces without the need for complex and expensive per-space occupancy detection technology. Available commercial solutions often deploy magnetic, infrared or ultrasonic sensors to detect the presence of a vehicle in a space [17] but provide no vehicle identification unless augments are made with technologies such as license plate readers at each entrance and exit point. Such solutions often require per-sensor wiring to facilitate data collection and supply power. Existing lot owners who do not have this infrastructure in place will face a potentially cost prohibitive decision when choosing to fully automate lot management. Other solutions that eliminate this ticketing and management infrastructure rely on *crowdsourced* information, such as users indicating their parked location to an Internet-based interface [18]. While not needing presence at ingress and egress points, these solutions rely on the driver to accurately (and honestly) indicate where they park. Additionally, mobile network-dependent solutions such as these become unusable in areas where there is no network connectivity such as underground parking garages. I address these problems by developing a smart parking solution that uses a single wireless radio technology to perform localization and transport of sensor data, independent of heavy network infrastructure or one sensor per-space requirements. The sensor platform must be low power or battery powered, and not require significant maintenance. The solution must be completely seamless to the end user, and must be resistant to adversary attack that may attempt to compromise user identity or the parking tracking infrastructure itself. Solutions like the ones presented in this thesis free drivers from the need to interface with the surrounding management system at all, outside of an initial destination indication.

My solution begins the process prior to a vehicle even entering a lot, enabling smart infrastructures such as multi-modal vehicle tasking to respond earlier to traveler's needs and in an on-demand fashion (which in turn incentivizes their use). My solution assists vehicle/traveler prediction problems in most cases by augmenting existing modal schedules with a human-specified *last mile* choices. This makes the goal of connecting humans to vehicles less predictive and more deterministic, while allowing the traveler to influence the tasking system in a meaningful way as they travel.

1.1 Problem Statement

The core problem addressed by this dissertation is the development of a low cost, low power, seamless smart parking and vehicle localization and safe privacy-supporting traveler identification system and its integration into a multi-modal autonomous vehicle scheduling and dispatch system to enable dynamic traceable multi-passenger transportation solution for high vehicle traffic and pedestrian dense areas.

1.2 Thesis Statement

It is possible to create a low cost Bluetooth BLE and mesh-based smart parking solution that provides an integrating, multi-modal mass transit enabling, technology between personal vehicle use and autonomous vehicle tasking and dispatch.

1.3 Contributions

I validate this thesis statement with the following technical contributions:

- Explored the use of Bluetooth beacon technology as a point-to-point distance measurement solution as part of a smart space computing resource fairness enforcement system, confirming its poor accuracy when used as a Euclidean distance measure, motivating early decisions to utilize multi-sensor radio maps in localization solutions.

- Design, implement, deployed, and tested a Bluetooth Low Energy (BLE)-only outdoor vehicle localization solution that predicts per-parking-space vehicle occupancy using a custom in-vehicle BLE beacon, RSSI radio map, and Random Forest machine learning classifier. Solution was deployed, tested and validated at the CCSA, GMU Fairfax Campus.

- Design, implement, deployed, and tested a Bluetooth-only self-forming, authenticated node, controlled flooding meshed sensor network that does not exclusively use BLE advertising channels for data transfer, thereby reducing interference with the localization solution. Solution was deployed tested and validated at the CCSA, GMU Fairfax Campus.

- Improved my mesh network to reduce message delay, duplication, and radio use by altering the network to form routes based off of radio-strength based links.

- Improved my localization solution accuracy by producing a parking lot *zoned* prediction model, and performed a comparison between improved and unimproved solution for several different vehicle types (of differing size/internal orientation).

- Designed, implemented, and tested a Bluetooth-based vehicle authentication and early-arrival system that enables lots to detect the approach of vehicles destined for it, and engage connected services prior to their arrival. This solution was briefly tested against and compared with a commercial object recognition camera solution.

- Designed and partially implemented an end-to-end AV scheduling and tasking solution, that combines privacy-preserved user-provided destination information and current at-location AV fleet conditions (power, assigned passenger counts and destinations, etc.) to schedule and task vehicles to transport passengers from near their vehicle within a smart parking lot to their final *last-mile* destination, effectively reducing localized traffic at that destination and reducing overall commute times, incentivizing its use. System use cases include L'Enfant Plaza, Washington DC, and the West Campus lot at GMU Fairfax. Implementation and experiments are currently underway at the West Campus lot.

- Developed a trust management system that consumes privacy-preserving traveler information such as destination, history, etc, and utilizes it to inform network defenders monitoring aircraft in-flight networks for suspicious activity. I see the traveler tracking capabilities of my multi-modal solution as potential data sources for this trust management system, equipping network defenders with additional traveler context from which they may infer patterns of trustworthiness.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows:

- Chapter 2 contains related work for each of the research areas overlapped by the work in this dissertation. I comment throughout this section where my work compares or diverges in these areas.
- Chapter 3 presents past work with Bluetooth Beacons and in-door localization that supports seamless interaction between users, mobile devices, and smart computing environments. I include this as background research I later used to inform the outdoor localization work using Bluetooth LE and Beacon technology.
- Chapter 4 outlines my first implementation of a smart parking solution, composed of a custom Bluetooth EDR mesh network of sensor nodes measuring in-vehicle Bluetooth LE beacons. I discuss the approach I took to discover an accurate machine learning model for this application, and identify some initial problems with my approach to be solved in later work. My system was deployed and tested at the Center for Collision Safety and Analysis (CCSA), at George Mason University, Fairfax VA.
- Chapter 5 presents a revised solution that solves many of the mesh density, queuing, and localization accuracy problems uncovered in past work. I also begin to solve the vehicle identification and authentication problem problem in this work, informally comparing my wireless-only solution to the use of object recognition cameras in my

particular use case. This updated solution was also deployed at the CCSA.

- Chapter 6 presents work that begins to create a comprehensive multi-modal coordination system that incentivizes the use of mass transit solutions through the use of on-demand, coordinated autonomous vehicles to provide a *last-mile* solution currently problematic for commuters. I target the Washington DC Wharf area and L'Enfant Plaza Metro station as our use case, enabling an on-demand, multi-passenger AV solution to incentivize the use of mass transit to connect passengers to the Wharf.

- Chapter 7 presents work that connects my smart parking solution to the coordination system outlined in Chapter 6. I enhance the smart parking system by developing a wireless *early-notification* system that informs the AV dispatch system of traveler arrival to the West Campus parking lot on the George Mason University's Fairfax Campus, enabling reduced traveler wait times for vehicles transporting them to the center of campus. I outline a basic scheduling algorithm that informally optimizes passenger preferences and system-dictated constraints to maximize per-AV occupancy, maximizing the effective traffic reduction around Patriot Circle, a major vehicle artery on campus currently stymied with traffic during peak usage times.

- Chapter 8 presents ongoing work that encompasses all prior capabilities developed within this dissertation, applied to a multi-site deployment located at the West Campus parking lot, on George Mason University's main campus.

- Chapter 9 includes the development of a trust management system that can enable multi-modal travel environments, such as passenger airline travel, to consume outputs from my privacy-preserving passenger tracking features to supplement in-flight network defense with passenger-trust-informed decisions.

- Chapter 10 lists the hypotheses put forth by this dissertation, discusses and validates my hypotheses, and lists limitations and future work.

- The dissertation concludes in Chapter 11.

Chapter 2: Background and Related Work

2.1 Preliminaries

This section contains a collection of background topics useful in further discussion.

2.1.1 Inverse-Square Law and Received Signal Strength Indicator

There are many relations in Physics that follow an inverse square. For example, Coulomb's Law [19] states that the electrostatic force exerted by one electrically charged body on another varies according to the inverse square of their distances. In wireless communications, signal strength when measured at a receiving antenna is referred to as the Received Signal Strength Indicator (RSSI). The higher an RSSI value, the *stronger* the wireless signal, typically indicating closer proximity to the source antenna. This value is a key component in localization systems that compute distance based on signal strength, as wireless signals also follow the inverse square relation. Many factors influence RSSI, in addition to distance, such as obstruction, transmission medium, and radio interference, however a commonly used but simplified expression [20] of an inverse square law for wireless RSSI through open air is found in 2.1.

A Bluetooth receiver can approximate the distance between it and a beacon broadcaster by comparing the advertised Received Signal Strength Indicator (RSSI) to the strength listed in the beacon. A basic formula for this computation derived from [20] is shown in Equation 2.1. Here, n represents a signal propagation constant, that varies depending on environmental factors such as radio interference, obstructions between antenna, etc.

$$RSSI_{received} = -10n[log_{10}(distance)] + RSSI_{calibration} \tag{2.1}$$

where n is a signal propagation constant based on transmission medium (commonly approximated for open air wireless transmissions as n = 2). $RSSI_{calibration}$ refers to a reference signal strength at 1 meter. Solving for distance using this equation is shown in Equation 2.2.

$$distance = 10^{[}(RSSI_{received} - RSSI_{calibration})/(-10n)] \tag{2.2}$$

2.1.2 Bluetooth Protocols and Communication

2.2 Wireless Localization

Early work with wireless localization used linear distance measures constructed using techniques like trilateration. By measuring the radio frequency strength, and using techniques based on Equation 2.2, linear distances can be approximated when the location of the antenna is known. A bulk of wireless localization work confronts the problem in an indoor environment where existing technology such as GPS may not effectively reach.

RSSI-based trilateration has also been studied for outdoor environments over IEEE 802.15.4 radios [21], where the authors improved linear distance estimates by adding nodes along with measurement tuning based on signal propagation parameters.

Additional work exists that favors radio fingerprinting over raw distance calculation methods like trilateration [22].

Faragher and Harle study signal loss and RSSI fingerprinting in detail, and propose an obstruction compensating calculation method for indoor spaces, comparing use of BLE and WiFi [23]. Additional indoor wireless positioning techniques are surveyed in [24].

2.2.1 Bluetooth-based

Prior to the release of the Bluetooth Mesh specification, Silver constructed a localization solution combined with a custom BLE mesh network using advertising channels for both solutions [25]. This work combined different filtering techniques to estimate distances and compares *discrete trilateration* and KNN fingerprinting to produce localization within an accuracy of 2.03m - 2.34m. Other work [26] focuses on RSSI fingerprinting of moving beacons, using Wasserstein distance interpolation, kNN, and Neural Networks.

2.3 Wireless Mesh Networks

2.3.1 Bluetooth-based

Similar to the Bluetooth SIG BLE mesh specification [27], my initial meshnet solution used a controlled flooding algorithm with *classic* Bluetooth instead of BLE to form the meshnet. Later iterations replaced this with a routed solution that greatly reduced message overhead and delay. Additional flooding [28] and routed Bluetooth mesh networks are surveyed in [29].

2.4 Parking Management

Research in Smart Parking systems encompasses many aspects of parking management including space occupancy detection, space availability prediction, payment management, rule violation alerts, and much of the IT infrastructure used to connect these aspects into a unified system. [30] uses GPS-based localization augmented by mobile phones and BLE beacons. This information is combined with the vehicle's On-Board Diagnostic (OBD-2) data to measure the state of the vehicle (speed, etc.). [31] provides a solution that uses BLE trilateration to develop a local parking management solution. Yee et al. [32] uses Zigbee radios. Some solutions on dead reckoning and other navigation-based solutions, such as those found within smartphone inertial sensors [33] that benefit from no requirement to modify the internal space, but controlled by the vehicle owner, making localization results untrustworthy. Other solutions such as [34] used historic data or and crowd-sourced GPS information to estimate space occupancy.

Solutions of this type benefit from no requirement to modify the internal space, but are also wholly controlled by the vehicle owner, and not the parking lot owner, which makes localization results untrustworthy (as-is). Additionally, such solutions require detailed physical maps of the parking lot, including physical anomalies (speed bumps, obstacles, etc.) that may influence vehicle movement.

Research in Smart Parking systems stretches across additional areas, including vehicle movement tracking [33], space availability detection and prediction [30, 34], autonomous payment management, space reservation and allocation systems [35] and rule violation detection [36] using a variety of wireless technologies such as RFID [37,38], image and object recognition [39], Infrared [40], Zigbee [32], etc., with an additional survey appearing in [41].

2.4.1 Space Availability and Prediction

Crowd-sourced Space Availability

These systems rely exclusively on human interaction to indicate when and where they park, and the management process typically only begins after a vehicle has parked. Other Smart Parking solutions focus on space availability prediction based on historic data, and crowd-sourced information such as GPS locations and other features to determine where a car is parked [34]. These solutions require long term data management, and carry a level of explicit trust in the providers of that information. Infrastructure controlled solutions that use local sensing technology allow for a central authority to certify where vehicles are within a space, an item of particular importance in scenarios where revenue is generated and fines are levied for rule violations.

2.4.2 Parking Reservation Systems

Commonplace Commercial Systems

While the above cited works use sensor networks to determine space occupancy, other systems rely on volunteered *crowd-sourced* information such as ParkMobile [18]. These systems also rely heavily on human interaction to indicate when and where they park, and the management process typically only begins after a vehicle has parked. Space reservation and allocation systems [35] and similar exist [42] that trust the user to provide accurate information. These systems are time consuming for the user both at the time of payment and because of the need for payment supporting gates at the ingress and egress points of parking lots.

2.4.3 Additional Systems and Miscellaneous Work

Lastly, several parking solutions have been proposed that include complex software (cloud-based, mobile phone-based) and per-space hardware stacks (Radio-frequency identification (RFID), optical or magnetic sensors, ZigBee or other RF radios, etc.), with little or no experimental validation [36, 43–45] and without details on communication protocols used, resilience features, power demands, etc. While theoretically they represent end-to-end parking solutions and may achieve some success at detecting vehicle occupancy, their viability remains to be tested.

Many complex designs have been proposed for software backends to manage user experiences, payment systems, and other IT aspects of smart parking. As our solution focuses on the creation of an accurate localization system and a parking lot spanning meshnet used to transport sensor data for that system, we do not include work that covers these IT aspects.

Solutions like ours free drivers from the need to interface with the surrounding management system at all, outside of an initial destination indication. Additionally, our solution begins the process prior to a vehicle even entering a lot, enabling smart infrastructures such as multi-modal vehicle tasking to respond earlier to traveler's needs and in an on-demand fashion (which in turn incentivizes their use).

2.5 Vehicle Scheduling/On-Demand Dispatch

Prior to planning autonomous vehicles for human usage, industrial application of robotics and autonomous vehicles was common. For example, underground mining applications were a domain space that benefited from scheduling and transport automation [46, 47]. As availability of consumer-grade electric vehicles and initiatives in public transportation to become more efficient and environmentally sound, autonomous vehicle emerged in these use cases.

Computational problems such as optimal mass transit scheduling based on prediction of travel times [48–52] have been well studied, even outside of the driver-less vehicle use-case. My work enhances vehicle/traveler prediction problems in most cases by augmenting existing modal schedules with a human-specified *last mile* choice. This makes the goal of connecting humans to vehicles less predictive and more deterministic, while allowing the traveler to influence the tasking of the vehicle as they travel.

The application of autonomous vehicles is listed as *game-changing* when applied to short distance travel [53] scenarios in *smart cities* [54, 55]. On-Demand mobility has also been incorporated into these scenarios [56]. Existing work for mobile travel applications are common, with one focusing specifically on tasking AVs [57]. This work enables travelers to request vehicles for any pickup and drop-off location in ways similar to existing ride-hailing service applications. Although the challenges of traffic management and incentivization of mass transit are not sufficiently addressed, potentially creating a solution which will increase travel time and environmental impacts due to point-to-point travel.

2.5.1 Autonomous Vehicle Specific Scheduling

Prior to more recent efforts in planning autonomous vehicles to serve public human transit, industrial applications of autonomous vehicles was common. For example, environments not always fit for human occupancy such as underground mining benefit from the use of AVs and thus mandate scheduling and transport automation research [46, 47].

2.5.2 On-Demand Mobility

On-Demand mobility has also been incorporated into these scenarios [56]. Existing work for mobile travel applications are common, with one focusing specifically on tasking AVs [57].

2.6 Diverging Work

2.6.1 Smart Computational Spaces

The notion of a computer leveraging surrounding infrastructure to augment its own was originally presented by Want et al. [58]. Their work created a mobile device from the ground up based on StrongARM and Linux, communicating with the environment using Bluetooth. Husemann et al. documents an IBM creation of a mobile hub [59] that users carry with them to facilitate inter-device communication of wearables. The proof of concept created an extensible platform for allowing heterogeneous communication technologies like Bluetooth, WiFi, and ZigBee. Users of current mobile technology, have access within their devices to communicate with most wireless communication platforms that the mobile hub provided. This makes carrying around a second device of this type no longer necessary.

Ballesteros et al. [60] produced a smart space software named Octopus, and a device operating system called Plan B [61,62], which replace most of the mobile platform. Bennaceur et al. posit that current mobile computing models have moved from a network serviced one, to a device centric one where users' devices connect to one another for data exchange. They provide a middleware named iBICOOP [63] that seeks to reduce user overhead when connecting to other devices. I believe such systems are largely incompatible with current mobile technology, and replace too many contemporary protocols with their own.

Johanson et al. [64] designed a space called iRoom, which focused on interfaces to walk-up displays within the space. Similar early work included Gaia [65] from Roman et al., which created a middleware for integrating technology within a traditional computing space. Infrared broadcast beacons were used for registration and directory services within the space.

A sample smart space similar to my use case is the work from Pering et al. [66] based upon popular tabletop collaboration space from Steelcase [67]. The authors customize the table by embedding three desktop computers within it, along with a tabletop touchscreen using wired connections. Work from Molyneaux et al. [68] constructs a room with multiple sensors and cameras that allow a central control system to sense and understand the physical locations of people and objects (chairs, etc.) within a room. We see these as being too expensive of a solution for most public spaces.

Another approach uses a thin client architecture over network protocols like RDP [69] or VNC [70]. Annamalai et al. [71] argues against this, claiming network latency dampens interactivity. Instead, they leverage an increase in mobile storage capability (e.g. a USB drive) to create a *desktop on a key chain (DoK)* containing an image of a full desktop virtual machine. Similar work is found in Soulpad [72].

Early work from Ferscha and Vogl [73] present a software system for creating access to large public display walls using a backend server communicating over GSM, 802.11b, or wired LAN. Commands and messages are relayed to service modules running on the backend server to control the content on the displays.

These solutions do not address the use of commodity hardware that users may commonly have, nor directly address security problems faced by contemporary smart spaces. I see my solution as a practical approach that can enable everyday locations to deploy KVMs to enhance the space with computing support.

2.6.2 Trust Management

Decentralized trust models [74] have been used frequently in the past to make access decisions in service oriented architectures [75], peer-to-peer (P2P) environments [76,77], Internet of Things (IoT) environments [78,79], social networks [80,81], and packet routing [82]. Al-Arayed et al. [83] combine three trust models to form one unified model that applies to general cases of Smart Spaces (called Trust in Smart Spaces; TISS). First is a P2P service provider model [84] that uses peer recommendations. Second is a trust bootstrapping solution, called *Trullo*, aimed at advertising content management of mobile users entering a specific physical location [85]. Third is a trust framework focused on fairness, that decides routing in a P2P network [86]. We use this work as a basis to compute direct experience and peer recommendation components and basic weighted sum trust calculations. Similar weighted sum trust models can be found in [87].

Using trust models for intrusion detection has been explored in past work. Trust can be used to uncover misbehaving wireless sensor nodes [76, 86] and routing in mobile ad-hoc networks [88, 89] when performance and other metrics have known ranges and behaviors. [90] applies trust management to the intrusion detection alerting process. Their system, called RepCIDN, evolves over time to identify untrustworthy alerting mechanisms so that they may be dealt with accordingly. Lastly, network artifact *reputation* plays a role in contemporary network defense, whereby external reputation providers inform a defender of what others think of an artifact [91, 92]. I contrast the use of the term ”reputation” among trust management research with these common tools within the network defense industry.

Chapter 3: Preliminary Work using Bluetooth Beacon technology in support of Smart Computational Spaces

Enabling general purpose computing in most public spaces is an expensive proposition. Many public spaces facilitate mobile users with WiFi access, but do not provide ubiquitous human computer interfaces or displays for the general public to connect to, outside of wired interfaces that undermine mobility. This largely prohibits traditional computing with display-less devices such as wearable computers.

In this Chapter I discuss preliminary prior work used to explore the security and fairness enforcement for commodity wireless Bluetooth interfaces for mobile devices in *smart spaces*. I address fairness for public interface sharing by designing a lightweight middleware solution that inserts a control layer between USB keyboard and mouse events relayed to mobile devices over a Bluetooth HCI service. I co-opt the fields within BLE beacons to identify devices and enforce security controls that protect the smart space's resources from access and availability attacks. If public spaces can leverage cost-effective, security-focused solutions such as ours, I believe they can turn into smart spaces, enabling more freedom of computer use within them.

3.1 Introduction

Many notions of a *smart space* have been proposed, but most are an extension of Weiser's notion of a seamless, maximally integrated environment where there is no explicit boundary between the computer and the user's everyday life [93]. Creating them is challenging partly due to the varying differences in how humans interact with technology, but also due to their high cost. Common public spaces, such as coffee shops, airport terminals, grocery stores, and other places where support for computer use has not been a primary goal are developing an increased interest in providing easy-to-use expenses paid services like human-computer interfaces to their customers. The expectation is that to take advantage of such services, the customers must not be forced to endure overly burdensome actions or bring an excessive amount of technology along with them.

There are many design decisions for such spaces. Users could bring a bulky laptop, or use a resource within the space such as desktop or a kiosk. One middle ground between these two extremes distributes computation off-device, a technique called *cyber foraging* [94]. Another constructs a portable device that contains all of a user's data and necessary computational power but uses resources and interfaces resident in the space, referred to as a *personal server* by Want et al. [58]. We posit that such a *personal server* solution is currently feasible among smartphone users, using off-the-shelf technology, without the need for hardware modification or assembly.

Current public spaces typically provide only a form of wireless Internet access, and no seamless integration between mobile users and interfaces within the environment, outside of wired kiosks that use a wired connection to the mobile device. This wired connection may come in the form of a wireless USB dongle [95] or other attachable hardware within this space, but such items could be stolen if the space does not have physical monitoring. Commercially available systems like these are often quite expensive, subjecting a space to an increased financial risk of providing this service to the general public.

Making better smart spaces requires product support such as a security layer. Although there are communication standards such as Bluetooth, and Miracast over WiFi, end devices like smart TVs do not offer a means to deploy a security layer around them. Similarly, Bluetooth mice and keyboards also often provide no means to insert a security and management layer between them and their users. These devices can also be easily stolen as there is no easy means to securely attach them to the space.

This lack of a security insertion point restricts defenders to resort to passive network monitoring, such as the use of a Wireless Intrusion Detection Systems (WIDS). But passive monitoring cannot be performed easily on encrypted links, such as those used in secure

Bluetooth communication (commonly found in keyboard pairings).

As a solution I proposed the deployment of a hybrid wired/wireless keyboard, video, mouse (KVM) solution with custom middleware housing a security layer. This KVM is wirelessly connected to mobile devices that coordinate positioning and fairness using active BLE beacons. I consider Android smartphones due to their popularity and developer support. I argue that a cyber adversary can exist within these spaces in the form of a bully, or one that violates fairness considerations, and thereby prevents other users from properly connecting to and using site resources. Additionally, due to the challenges in automating connections in smart spaces, I designed a thin middleware that resides on mobile devices and within the site for connection establishment, etc. I combined a physical/wireless KVM model that allows sites to use wired physical interfaces and connect wirelessly to user devices, using a middleware that enforces security and fairness policies. This arrangement also provides a network traffic inspection point both at the user's end device and at the site's service device for each established communication.

3.2 Usage Scenario

Users who enter smart spaces must quickly interface with the environment as needed without wasting too much effort in configuring their device. Given the diversity of mobile devices, this is a significant challenge. My objective is to support transient users that keep their device secured in a pocket or bag, and migrate to and from a KVM interface without losing security or worrying about being physically tethered to that particular interface. We would enable the user to connect their devices to a minimum of keyboard, mouse, and video with minimal human interface on-device, completely wirelessly. The solution must not require any hardware modification to the device (such as a hardware dongle), or change to the underlying operating system (e.g. *rooting*), or require expensive modifications to commercially available keyboards, mice, and LCD displays.

A popular communication standard supported by many Android devices is Miracast [96], that allows screen sharing between a device and a display using WiFi-Direct. Displays must provide a Miracast software receiver connected to its WiFi interface. The primary limitation with most low cost Miracast receivers [97, 98] is the lack of a published API to allow for programmatic control from an Android device. Users are required to manually scan for receivers and initiate connections from their device. Additionally, there exists no way to enforce fairness or access control to a particular receiver. The wireless connections themselves are peer-to-peer, preventing any easy means to man-in-the-middle these connections so that such a layer could be deployed. To use Miracast in the way we would like, one must use an open source software receiver [99] and modify it.

The most common means to wirelessly connect keyboards and mice to Android devices is over Bluetooth. For the most part, initiating and managing these connections can be handled programmatically. We found, however, that the *connect* function for Bluetooth within Android is no longer available without rooting the device. This then forces any middleware that doesn't require root to support this manual step through the device's display.

Additionally, most commercially available Bluetooth keyboards and mice make a poor choice smart spaces, as they require the user to physically place them into discovery mode prior to pairing. This prevents complete automation and presents a user burden. These devices can also be easily stolen from a space and have a limited battery life. Lastly, these devices create direct-to-device connections, so any management layer that enforced fairness or additional security features (like connection monitoring) can not be deployed.

WiFi connectivity can easily be automated within Android, and easily connect the device to the Internet or other connected networks once the device knows what network to join and what key material to use.

3.3 Proposed Solution

I proposed a solution that inserts a management and security layer in between the user's device and the surrounding interfaces within the smart space to provide a place to deploy security and access controls. I implement this layer as a pair of middleware components, located on the mobile device (referred to as a *Device Manager*) and on an IoT-scale computing platform such as a Raspberry Pi (referred to as a *Site Manager*) connected to each KVM deployed (See Figure 3.1).

The middleware is responsible for coordinating KVM discovery, connection setup and tear-down, maintaining and monitoring fair use, and managing disconnections based on user request or behavior. My experimental Site-Manager platform runs on a Raspberry Pi with a USB keyboard and mouse and relays device events through a published Bluetooth HCI device using on-board Bluetooth radios. The Device Manager coordinates device connections to these published Bluetooth HCI devices according to the existing Bluetooth protocol. I use a Miracast receiver (Roku 2) to provide wireless LCD display capability. I limit my interaction with the phone's display to simulate the use of a reduced interface wearable. The display shows a very minimal on-screen interface composed of simulated physical buttons. This permits us to use audio queues and voice control, Bluetooth and WiFi services, and the on-board camera within my solution.

3.3.1 Space and KVM Bootstrapping

Prior to entering the space, I assume the user has installed the Device Manager using a site-advertised method, such as an appstore. The Device Manager bootstraps to the WiFi network within the space utilizing a scanned QR code. Upon entry, users scan the code (displayed on a wall near the entrance) with the device's camera. Our code is a Base-64 encoded string containing a room identifier concatenated with the WiFi SSID, WPA2-PSK key, and IP and port of the registration component of the Site Manager. I use a commonly available Java library and scanning application from ZXing [100] to scan, read, and decode the QR code. Once the code is ingested and parsed, the Device Manager automatically establishes a connection to the local WiFi network and begins registration activities.

Registration allows the Device and Site Managers to advertise presence and exchange key material for use later during KVM reservation and to encrypt a communication channel

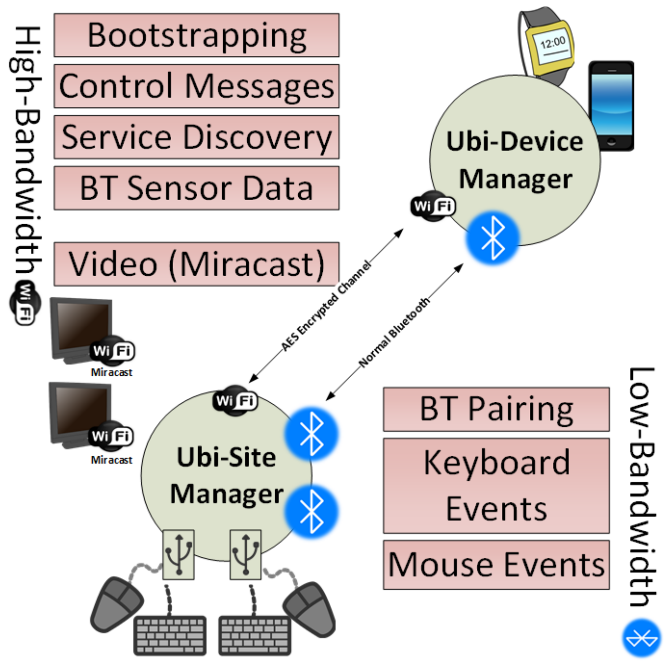

Figure 3.1: Model Smart Space

whenever they need to communicate over WiFi. If a space has more than one Site Manager, this key material and identity information is shared between them over WiFi. During the initial site installation, all Site Managers perform a similar key exchange between them to allow for the creation of inter-Site-Manager encrypted channels. This is done to protect their communications over WiFi, as the QR code is essentially public and thus an adversary can potentially recover the WiFi password and passively monitor the network unnoticed.

This initial network connection setup happens very quickly, allowing a user to continue into the space uninterrupted. An audible tone is produced from the device when this process is complete, and the device holder walks to a KVM interface. Each Device Manager then continuously emits a custom Bluetooth Low Energy (BLE) beacon [101] based on the key material originally exchanged with the site, allowing Site Managers to detect their local proximity. Location could be determined by using on-board GPS, however this is a problem as I envision most smart spaces to be indoor.

This beacon allows the Site Manager to wait until a valid registered Device Manager is nearby (consistently within 3 feet) before publishing its interfaces. This method is a variation on *port knocking* [102] and reduces exposure to potential adversary attack by publishing services on-demand. Additionally, such a structure reduces the need for a central discovery server or other service discovery infrastructure. This beacon contains a payload with an authentication token (valid for 24 hours from time of issue) that provides the device's identity to the Site Manager. The Site Manager determines if the user is valid, if the interface they requesting is available, that the user is not on any internal blacklists and is not currently using any other KVM. If the user is acceptable, the Site Manager makes the appropriate Bluetooth antenna active, puts it in discovery mode, and publishes a keyboard and mouse service with a name unique to that user. The Device Manager obtains these names over a secure WiFi channel, along with the pin needed to complete keyboard pairing, and automatically performs pairing of the keyboard and mouse. Due to our previously mentioned limitation in non-rooted Android devices, I am not able to initiate a Bluetooth *connect* to finish automating the connection without explicit user interaction. I leave mitigation of this issue to future work. Additionally, a timeout of 30 seconds exists between the initial beacon, and the final Bluetooth pairing to facilitate users leaving the space during this process, or other failure cases. More detail for my beacon scheme is found in Section 3.3.3.

3.3.2 Management Middleware

The Device Manager is a non-intrusive Android application (intended to run as a background service once deployed), that uses only a simple API or user-permitted system calls to communicate with Bluetooth and WiFi controls within the operating system. This prevents any change to the device functionality when not within the space or not using site-managed connections.

The current implementation of the Site Manager exists as a set of python scripts and services on a Raspberry Pi (running Ubuntu Mate) with at least one USB keyboard and mouse pair, and corresponding Bluetooth antennae (one for each interface). After bootstrapping, keyboard and mouse events are read from the Pi's operating system device input files (`/dev/input/*`) and are pushed out over a published Bluetooth HCI service to the mobile device. My implementation for this input device event relay is based on two public codebases for Bluetooth keyboard emulation [103] . I reuse portions of these codebases to provide the USB device reading and relay functions and Bluetooth service publishing functions.

An alternative approach for turning wired HCI devices into Bluetooth ones is to tunnel events over WiFi. However this would require a rooted device to allow for copying keyboard and mouse events to Android's local inputs. As my solution requires a non-rooted device, I could not use this option.

3.3.3 Location, Fairness, and Availability

Each user of the space can connect to no more than one KVM at any one time. This is enforced at connection bootstrap time, and by Site Managers that continually communicate and exchange connection information over WiFi. Old connections are terminated if new ones are requested. If two or more users arrive at an interface at the same time, the corresponding Site Manager grants access on a first-come-first-serve basis. The timeout procedure detailed in Section 3.3.4 covers the recycling of inactive connections.

Device Location and Tracking

A Bluetooth receiver can approximate the distance between it and a beacon broadcaster by comparing the advertised Received Signal Strength Indicator (RSSI) to the strength listed in the beacon. A basic formula for this computation [20] is shown in Equation 1. Here, n represents a signal propagation constant, that varies depending on environmental factors such as radio interference, obstructions between antenna, etc.

$$RSSI_{measured} = -10n[log_{10}(dist)] + RSSI_{reference} \tag{3.1}$$

$$dist = 10^[(RSSI_{measured} - RSSI_{reference})/(-10n)] \tag{3.2}$$

With multiple radio sources transmitters at known fixed locations, a receiver can use simple techniques such as *trilateration* to determine their position. However, in practice Bluetooth RSSI measurements have low precision. Some of this inaccuracy can be potentially overcome by combining similar techniques with a space's WiFi signals or multiple Bluetooth antennae, but using these techniques will add a level of complexity and cost to the proposed solution, which must be avoided.

My current design of device tracking and distance measurements uses only two stationary beacons for each KVM as shown in Figure 3.2. One beacon is transmitted from a Site Manager and is used to determine when a registered device is near it. The second beacon is transmitted from another Site Manager operating in a *location assist* mode, allowing devices and other Site Managers to measure which direction the KVM display is facing, to know where to expect the user and their device. This *location assist beacon* must be placed along the KVM orientation axis (the direction the display faces) 1.5 feet past the expected position for the end-user when using the interface. I choose 1.5 feet due to the expectation that it will be a comfortable distance between a user and the KVM. Prior to this final placement, a special administrator interactive script is run on the KVM-providing Site Manager that takes RSSI measurements when an installer places the location assist beacon immediately next to it, 1.5 feet along the orientation path (near where the user is expected to stand), and at its final destination, 3 feet away.

Should a user move outside of their expected location by 1.5 feet or more, an audible tone is emitted by the Device Manager, and a connection teardown timeout begins as described in Section 3.3.4.

As the connection establishment and teardown process requires a user interaction to overcome limitations in Android, my solution can be more tolerant of inaccurate distance measurements, where only local proximity (not highly accurate position) is needed.

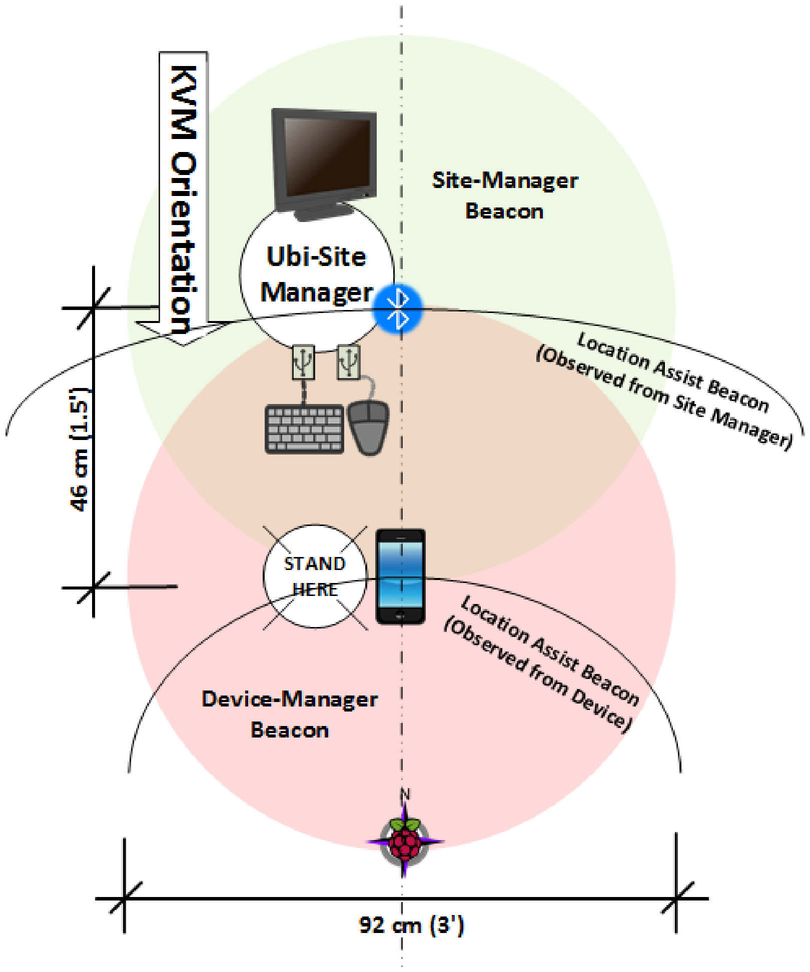

Figure 3.2: Distance and KVM Orientation Measurement System

Authenticated Beacon Design

Different beacon standards set the length of the UUID field differently. I choose the Eddystone format to ease implementation on Android. I reuse the 16-byte identifier fields to provide a rolling-code of sorts, to provide unique beacons that contain our encrypted code, leaving the remainder of the beacon as-is. My new UUID field is as follows:

$$\left\langle 64\, bit\, identifier + 32bit\, epochtime + 32bit\, random\, padding \right\rangle$$

Prior to insertion into the beacon, this field is encrypted with the sender's AES128 key so that a passive observer may not easily recover the identifier from the message. The use of epoch time and randomized fields provides uniqueness to each beacon, and a logical clock for ordering the beacons. My messaging scheme is tolerant of dropped or disrupted beacons, as long as at least one beacon is readable during the timeout period.

One problem exists with this use, as there is no additional room within this beacon to transmit an Initialization Vector (IV) along with the message. I leave the remedy of this to future work, where such information could be send out-of-band over the WiFi connection, or AES could be replaced entirely with a new cryptographic primitive that uses a smaller block size, multiple messages, or is otherwise usable within a 128 bit-per message channel.

Use of Proof-of-Life

The Site Manager follows a protocol for connecting valid devices, but there is no way within that procedure to determine if there is a human using the device. I outline a potential attack in Section 3.4.2 where an adversary clandestinely consumes nearby interfaces, preventing legitimate users from successfully connecting to them. To overcome such an attack, I use a Proof-of-Life test that the device (its user) must pass, or the connection is terminated.

If a device has not passed a Proof-of-Life test during its connection to that interface, and has not moved for a specified period of time (determined by a significant change in RSSI-based distance calculation), a Proof-of-Life test is initiated when another user requests a connection to that same interface. I further discuss the potential successes, and abuses of this arrangement in Section 3.4.2.

Because the Site Manager controls the KVM interfaces, it could divert the display and keyboard to a special device-provided terminal and present a CAPCHA or other more complex test. My design cannot support this yet, as the existing Miracast connections are not fully automated. Instead, the Site Manager implements a Proof-of-Life test by sending a special request to the Device Manager, instructing it to emit a special audible tone known to the user, to request that they either intentionally move around within the space or press the connection initiation button on their device. Either action passes the test. While the test itself is computationally simple, it requires a physical act which I believe is strong against the local adversary (I analyze this claim in 4.2.1).

User Blacklisting

Should a device perform malicious acts or develop a reputation of being untrustworthy, the Site Manager can add their Bluetooth hardware address to a blacklist that will cause refusal of future attempts to connect to a KVM.

At this time, detection of protocol attacks is isolated to the submission of invalid beacons, however my current design has weaknesses discussed in Section 3.4.2, preventing me from including its functionality of the current deployment. I see development of better detection as future work.

3.3.4 Connection Teardown

When a user wishes to no longer use an interface, they may click the same device button used earlier to initiate the connection, or walk out of range of the KVM long enough for the timeout period to occur. I use the device tracking scheme outlined in Section 3.3.3 to determine when a user has left the interface for a duration of 30 seconds, and automatically recycles the interface. I also engage the user to inform them they have moved too far away from the interface, with an audible tone every 5 seconds.

In practice, I prefer the explicit use of the button to disconnect, as it is faster than relying on the distance calculation and prevents any accidental initiation of the connection timeout period due to errors in distance calculation or intermittent connectivity issues with Bluetooth. I opt for a 1 second beacon frequency to reduce battery drain which occasionally encourages these issues, but plan to study this effect more in the future, and potentially adjust this interval.

3.4 Security Analysis

3.4.1 Adversary Model

I define the adversary as one who masquerades as a benign user within the space and works to exhaust available KVM interfaces so that no users may access them. I assume the adversary has access to all public information that normal users have access to, including an installed Device Manager, knowledge of all procedures, WiFi access (via QR code scanning), and physical access to the smart space.

3.4.2 Sample Attacks and Mitigations

I consider three classes of attacks available to the adversary in this smart space. An adversary could attempt to consume all resources, force benign users onto the site's blacklist, or increase user burden of my Proof-of-Life system.

Resource Exhaustion Attacks

There are four variations of *resource exhaustion attacks* that can be used in this environment. First, an adversary could determine a means to perform multiple simultaneous reservations during the site bootstrap process, thus preventing new benign users from reserving an interface, at least until those reservations expired and the Site Manager returned them to available status. However, my reservation system prevents this as only one KVM can be requested at one time. Second, if an adversary could determine a way to prevent a device manager from communicating a connection release to a Site Manager, no resource release would be initiated. Because my device manager and Site manager communicate reservation information over WiFi through an encrypted tunnel, the adversary would have to disrupt that wireless communication, and potentially expose themselves. Third, if an adversary could pick up another user's connection after they have left a KVM, prior to the timeout expiration, they could prevent the timeout, and consume the resource. However, this would only be possible if the adversary were able to forge legitimate beacons for that particular device. This would require breaking AES128. Lastly, a malicious user could carry around many devices with them, consume KVM connections, and simply place those devices near each KVM, but out of sight. This attack is unreasonable as my Proof-of-Life would require the adversary to move to each KVM within 30 seconds when challenged, which would most likely be noticed by the user who initiated the KVM reservation.

Forced Blacklist Attacks

Another type of attack is directed at convincing the Site Manager that one or more benign users within the space are malicious, forcing them to the connection blacklist. If a malicious user could forge fake reservation communications over WiFi (which they can passively monitor), and include a known device identifier, this would cause the Site Manager to include that user on the blacklist. An evolution of this attack could be to pre-populate the list with all identifiers, blocking any future user. However, to perform either action, the encrypted tunnel would need to be broken so that both the device identifiers and their matching AES keys could be recovered.

Creating a Proof-of-Life burden

If an adversary walked around a space requesting interfaces that are currently in use, they may trigger a Proof-of-Life test for the KVM user. If the user fails the test, their connection is recycled. This forces users who wish to legitimately use the interface to perform a test outside of a reasonable need. If I use the blacklist such that users who fail to solve the proof are added to it, this scenario becomes an extension of the Forced Blacklist Attack listed above. I assume each user of the space understands the Proof-of-Life test. However, I could replace the audible tone with a specific recorded messages communicating instructions directly to the user. I choose not to explicitly place users on a blacklist for failing the test, and simply recycle their connection. This limits the adversary to only becoming capable of terminating user connections, and only in cases where the user chooses not to respond to the test.

3.5 Implementation Status and Future Work

I was able to automate some, but not all of the necessary steps required to seamlessly connect my devices to their nearby KVM without user involvement. As a result, my implementation remains an early work-in-progress. Android operating system security features prevent me from programatically controlling all required *connect* Bluetooth functions, as well as connecting video through Miracast. Users must initiate these connections manually, but I plan to resolve in future work.

It is unclear what effects using public interfaces and software will have, precisely, on a user's level of trust. It may be the case that a user would prefer to simply use some aspects of the site-integration and not others. For example, a user may decide to use the video display in a space, but not the keyboard and mouse and would opt to bring their own. Such an arrangement complicates my autonomous *walk-up* use case, but may need to be planned for so that the site's resources are more often used. I list these scenarios as future work.

3.6 Conclusions

In this chapter I present past work with Bluetooth-based fairness enforcement for commodity wireless interfaces for mobile devices. I proposed a design for a lightweight middleware solution that converts USB interfaces into Bluetooth HCI interfaces that can be managed fairly within a smart space. I employ the use of BLE beacons to track users and estimate proximity, and combine this functionality with my fairness techniques. My adversary and security analysis identifies potential attacks to my design, and I discussed how that design addresses them. I believe that smart spaces that deploy solutions like mine can do so easily, and in a cost effective manner to enable transient users within the space to perform more complex computing activities, securely, and without the need to carry around their own interfaces.

The work presented in this chapter was published in *Implementing Fair Wireless Interfaces with Off-The-Shelf Hardware in Smart Spaces,* Paul Seymer, Duminda Wijesekera, In Proceedings of the 18th International Conference on Internet Computing and Internet of Things (ICOMP '17), July 17-20, 2017, Las Vegas, Nevada, USA [104].

Chapter 4: Secure Outdoor Smart Parking using Dual Mode Bluetooth Mesh Networks

In this chapter, I present my first iteration of work creating, deploying, and testing a Bluetooth-only based smart parking solution that performs vehicle localization in an outdoor parking lot. Section 4.1 provides an introduction to the problem space and discusses additional motivations. Section 4.2 presents the overall solution I employed, with Section 4.2.1 specifying localization and Section 4.2.2 outlining the self-forming controlled flooding mesh network. Experiments and results are found in Section 4.3. The chapter concludes with Section 4.4.

4.1 Introduction

Efficient parking lot automation continues to be a focal point of many smart city initiatives. Most existing unattended parking lots suffer from a lack of seamless automation, instead deploying ticketing and payment at ingress and egress points or other systems with heavy user involvement that often form bottlenecks. Similarly, many lots use per-space sensing with expensive networking and power requirements simply to determine space occupancy. Parking solutions that are free from the delay caused by this user burden and infrastructure could experience faster occupancy turnover with lower cost. An ongoing challenge in developing seamless parking experiences is the detection and identification of vehicles in parking spaces without the need for complex and expensive per-space occupancy detection technology.

I developed a smart parking solution that uses a single low-power wireless radio technology to seamlessly perform parked vehicle localization and transport of sensor data for use by a central management system. My solution uses a sparse, self-forming network of dual-mode Bluetooth sensors within a parking area to observe the presence of customized authenticated Bluetooth Low-Energy (BLE) beacons placed in vehicles parked in the lot. My localization technique is based on radio fingerprinting using Received Signal Strength Indication (RSSI) values from the beacon, and a random forest machine learning classifier that predicts where the vehicle is parked based on its fingerprint.

I implemented my solution in Python using commodity Internet of Things (IoT) hardware and deployed it to a 105 space outdoor parking lot. There, I conducted fingerprinting and prediction experiments. My results show that my exact-space prediction model evaluates with a high accuracy using radio training data (90.7% correctly identified), and my in-vehicle tests show a promising result (69.17% accurate up to and including 3 spaces away), even without employing tuning and data filtering techniques. This encouraging result shows that localization using Bluetooth is a viable means of managing parked vehicles, with great promise for a variety of future parking management applications.

4.2 Proposed Solution

I developed a smart parking solution based exclusively on Bluetooth radio. Small sensor platforms (nodes) form an authenticated network, distributed to the physical footprint of a parking lot. These nodes listen for broadcasts from a custom Bluetooth Low Energy (BLE) beacon, found in each vehicle that parks in the lot. Received Signal Strength Indication (RSSI) values from broadcasts from the beacon are picked up by the nodes within range, validated, encrypted, and sent back to a designated central node where space prediction occurs. Space prediction uses an optimized Random Forest machine learning model trained from previously recorded RSSI values for each space in the lot. I constructed and deployed this solution using Raspberry Pi 3s, to a small outdoor parking structure on the George Mason University (GMU) campus in Fairfax, VA. Figure 4.1 shows a high-level view of my solution with a high level view of each node's location within the Bluetooth software stack in Figure 4.2. I describe this solution in more detail in the following subsections.

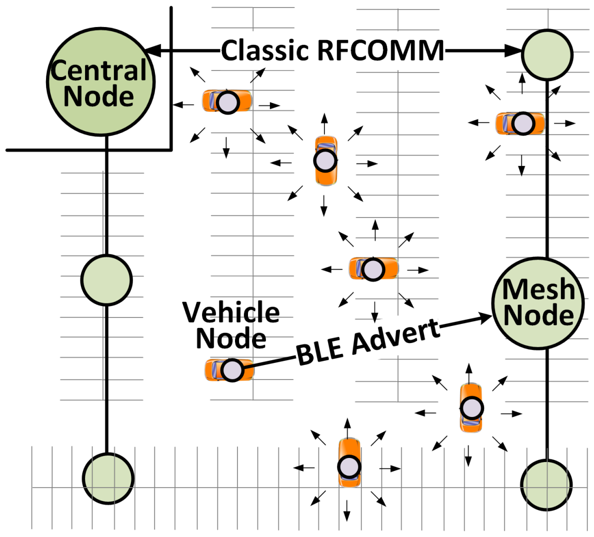

Figure 4.1: Mesh Network Components

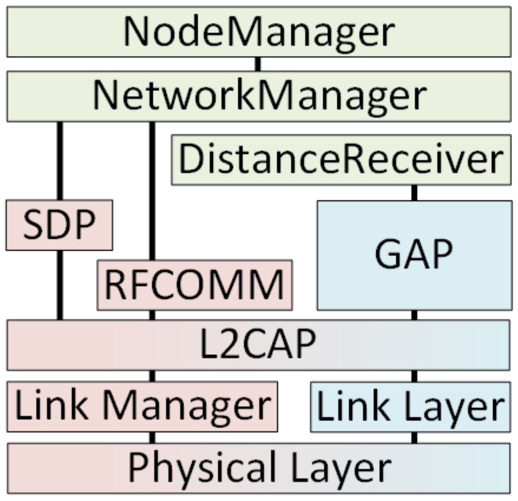

Figure 4.2: Integration w/Bluetooth Stack

4.2.1 Bluetooth Localization

My vehicle localization technique is based on radio fingerprinting each space, and uses a supervised learning algorithm to make a final determination of space occupancy. When the nodes were initially installed I deployed a continuously broadcasting BLE beacon to each space in the parking lot and collected RSSI values for the beacon over a short period of time. These values were used as training data for the supervised learning algorithm, obtaining a unique RSSI profile for each space. I summarize this process in Algorithm 1.

As nodes are located at different distances from each space, individual RSSI values at each node are expected to differ along with this distance. While Euclidean distance calculations with Bluetooth beacons has been found to be imprecise and unreliable [20], radio fingerprinting solutions show potential for use [105]. Details of my experiments appear in Section 4.3.

4.2.2 ”Smart” Mesh Network

Contemporary mesh network options, including the BLE Mesh Specification [27], use both flooding and routing based solutions [29] and span radio broadcasts over both advertising

Algorithm 1 Occupancy Detection Procedures

```
1: procedure space_fingerprinting(recording interval $i$)
2:     Enable continuous BLE advertising inquiry on all nodes.
3:     for each parking space do
4:         Move broadcasting beacon to center of parking space.
5:         for $i$ seconds interval do
6:             for each node do
7:                 for each observed valid beacon in inqury scan do
8:                     record (rssi, node_id, timestamp)
9:     Recover data from each node, and store in single dataset $d$.
10: procedure prediction_model_train(dataset $d$)
11:     for each parking space $p$ in dataset $d$ do
12:         for time interval $i$ do
13:             extract features for $p$
14:             create feature vector for extract features
15:             store feature vector in labeled training set $t$
16:     Create trained model $m$ with supervised learning algorithm
17:     Test and Store $m$
18: procedure vehicle_location_predict(model $m$, interval $i$)
19:     Enable continuous BLE advertising inquiry on all nodes.
20:     for $i$ seconds interval do
21:         for each node do
22:             for each observed valid beacon in inqury scan do
23:                 record rssi value, node id, timestamp
24:         extract features $f$ for $p$
25:         create feature vector $v$, for $f$
26:         predict parking space $s$ using $v$ input into $m$
27:         store $s$
```

and data channels. BLE mesh solutions that use flooding over advertising channels for data transfer are problematic for my localization solution, as their use causes radio interference, negatively affecting occupancy prediction. I choose to implement a form of controlled flooding to avoid the computational, storage, and network overheads commonly associated with routing-based solutions, but instead of exclusively using advertising channels for data transfer I implement a managed flooding solution over Bluetooth *Classic* Enhanced Data Rate (EDR) which spreads transmission over the entire Bluetooth spectrum. This allows one to execute network functions at higher transfer rates, while simultaneously reducing interference with our localization solution by reducing channel overlap and taking advantage of Bluetooth's adaptive frequency hopping feature [106]. This is of particular benefit in environments with a high potential of environmental interference (ambient WiFi signals, etc.)

Pre-Deployment Configurations

Prior to deployment and out of band of the Bluetooth network, each node exchanges two 256-bit keys with the designated central node, later used to encrypt messages using the Advanced Encryption Standard (AES) algorithm and sign messages using a hash-based message authentication code (HMAC). The design uses the Secure Hash Algorithm 256 (SHA-256) as the HMAC (HMAC-SHA256). I encrypt each message upon creation to ensure secrecy and integrity but also forgo the use of individually encrypted channels between nodes that would require additional power and computation. Additionally, each node receives a second pair of AES/HMAC keys for use in broadcast messages that contain node management instructions from the central node. Because my meshnet uses managed flooding, all nodes see at least one copy of every message, making message broadcasting require no additional implementation. In addition to keys, the central node assigns a unique identifier and the parking network's Bluetooth Universally Unique Identifier (UUID) to each node.

Network Construction

Each node that joins the meshnet must be authenticated by the central node. Since Bluetooth has a limited range, new nodes may not be able to communicate directly with the central node when they wish to join the network. As mesh networks extend the effective range of any single node by relaying its messages through other nodes, my mesh network is formed from the central node outward allowing any new nodes to have a network path back to the central node for authentication. New nodes, once powered on, go into a *staging* mode where they broadcast a service with the pre-exchanged Bluetooth UUID. During network construction each networked node scans for near-by Bluetooth services that match this UUID, using Bluetooth's built-in Service Discovery Protocol (SDP) until all nodes are joined. To ensure that only nodes close enough to establish a reliable connection are joined to the network, any discovered node with a consistent RSSI measurement below -80 decibel -miliwatts (dBm) is ignored. Nodes that have RSSI values above -80 dBm are authenticated using the protocol shown in Figure 4.3.

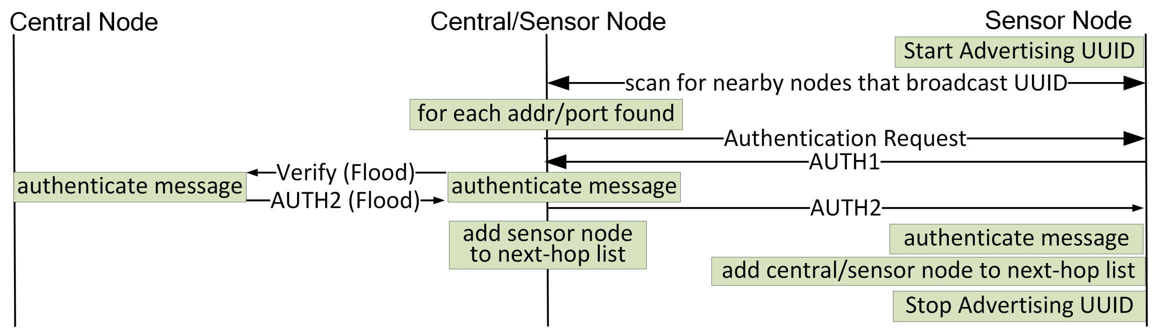

Figure 4.3: New Node Rendezvous Protocol

Once a node is authenticated, its connection information is stored within the node it authenticated through so that future messages can be flooded to it. In the event a node loses contact with the network, or changes its hardware address, nodes within the network that communicate with it will timeout, and inform the central node of the failure. This triggers a repeat of the rendezvous protocol, re-joining the missing node once its service begins broadcasting again and re-authenticates. Performing rendezvous based on UUID combined with pre-exchanged key material allows the network to support privacy and security features that utilize random hardware addressing, a feature already in BLE. Functional primitives for network construction are shown in Algorithm 2. Messages could be of any size, however to reduce network overhead all messages are fit into a single block of AES in Cypher Block Chaining (CBC) mode appended with the HMAC-SHA256 signature to provide authentication and integrity checking, due to ease of implementation and wide acceptance. The message format appears in Section 4.2.2.

The number of total nodes is known to the central node (due to pre-deployment key exchange). Once all nodes on the network have authenticated, the central node instructs the network to stop the rendezvous protocol, and all nodes terminate their SDP services to save power and reduce advertising traffic that would cause interference with distance calculations.

Message Flooding

The system use a controlled flooding algorithm to send messages throughout the network. When a node has a message to send, it floods the message to all of its neighbor nodes. Each node stores a buffer of recently seen messages. When a node receives a message, it checks this buffer and discards a message it has already seen. Otherwise the node then decides if the message is for them, or for another node (e.g. re-flooded) by examining a predefined message field. Re-flooded messages are sent to all neighbor nodes except the node that originally sent the message. This results in every node seeing every message, but never using duplicate messages. All messages are unique, within a reasonable time period, as each is encrypted and signed resulting in a pseudo-random set of bits for each message.

Message Format

To allow for efficient message decoding, I prepend each message with a 4-bit message type, and a 16-bit node identifier for the node that originally sent the message. This is followed by a 108-bit payload containing authentication information, RSSI measurements, or other data that needs to traverse the network in the message. This forms the 128-bit AES block to be encrypted and signed. To enable efficient decryption, once this message is received by its intended recipient, a 2-bit *mode* is pre-pended: 00 for messages destined to the central node, 01 for messages destined for one of the nodes, 10 for broadcast messages and 11 for non-flooded messages such as for those used during exchange of tokens in the authentication process. Additionally, there is a 16-bit node identifier also pre-pended and corresponds to either the key that the central node should use to decrypt the message, or the node that the message applies to. This forms a total of 530 bits for each message sent over the radio.

4.3 Experiments and Results

The experimental platform is composed of several Raspberry Pi 3s (with built-in dual mode Bluetooth) running Ubuntu Mate 16.4.1, equipped with 10kmAh portable battery pack to enable uninterrupted outdoor deployment. All code is written in Python, using the pybluez library [107] and the Bluez Linux Bluetooth stack [108]. I chose a computationally overpowered and open platform to facilitate evolving the system and conduct functional experiments without hardware or software performance limitations. However, I discovered throughout the course of my work, that the Bluetooth capabilities within the System On Chip (SoC) used in the Raspberry Pi became unstable when running my code, resulting in the need to power-cycle the node within minutes of use to maintain proper Bluetooth functionality. I addressed this by replacing the use of the on-board Bluetooth with a commodity Bluetooth USB adapter from StarTech [109] (which uses a different chipset). Initial localization experiments were conducted within an Antioch Chamber, but the parking experiments were conducted in the outdoor lot shown in Figure 4.5.

4.3.1 Selecting a Localization Technique

I conducted a preliminary set of distance calibration experiments for a sampling of my nodes to gain insight into feasibility for trilateration-based localization. RSSI measurements were taken inside the Antioch chamber at fixed distances of 1m, 1.5m, 2m, 2.5m, and 3.0m, from a laptop equipped with an Ubertooth One (for consistent measurements) [110]. There was a large variance between 1m RSSI measurements, ranging from -19 dBm (in two nodes) and between -20 and -30 dBm (for all other nodes). Additionally, RSSI measurements did not consistently decrease in value as nodes were moved farther away from our measurement antenna. This behavior and related literature showed RSSI-based trilateration should be limited to proximity-only applications.

A popular alternative used in indoor localization is to use RSSI measurements to form radio fingerprints for known locations within the space [105]. Claims have been made that suggest Bluetooth can perform data transfers between radios at least 350 meters apart [111]. However to confirm that this is the case for my target environment, I obtained RSSI measurements for a BLE broadcast from a node and the most distant parts of the parking lot while maintaining line of sight. I observed a successful beacon observation every 2-30 seconds with an RSSI value ranging between -80 to -90 dBm. This experiment confirmed that a vehicle could park anywhere and its beacon would be observable, provided we had line of site.

4.3.2 Choosing Physical Locations for Measurement Nodes

Under ideal conditions, nodes would be placed at equal distances from one another at a range that balances lot coverage and network connectivity. However, I expect that creating an accurate radio profile for each space will require more sensor nodes than network nodes needed to transfer data. Additionally, physical parameters of the lot, such as placement of mounting locations, obstructions due to trees or structures, or elevation changes must be taken into account to produce a solution that is both functional and practical to install and maintain. The target parking lot shown in Figure 4.5 has four tall lampposts approximately 20 feet apart that make obvious locations for our initial node deployment as they provide line of sight to most spaces, and do not require that they further obstruct the parking lot. I deployed nodes 2, 4, 6, and 7 to each of the posts. These locations, however are far from the parking spaces near the entrance of the lot, so I deployed node 5 to the entrance of the lot and a central node to an office space on the second floor of the building the lot serves (that contains a window facing node 5). However, this arrangement resulted in no line of site between the lamp post nodes and the entrance and central nodes. However there was no practical man-made structure to mount an additional node to, so I hung a final node on a tree along the south edge of the lot (node 1).

Measuring Inter-node Signal Strength

To study the feasibility of my deployment and confirm that my nodes are placed at usable locations, I used BLE advertising broadcasts from each node to observe how strong a node's radio was to other nodes in the network. I consider signals above -70 to -80 dBm as sufficient to form out the mesh network. For this experiment, I configured each node to continuously broadcast a BLE beacon and record all observable BLE beacons. I ran this activity for 5 minutes, and summarize the result in Figure 4.4. I noticed that some node pairings resulted in a low volume of observed advertising counts and low RSSI values, while others had higher values and counts. This appeared consistent with my assumptions about distance and obstruction. For example, mounting location for node 2 was immediately behind a large aluminum sign, blocking line of sight between it and node 5. The resulting measurements confirm that this is indeed an obstruction, however I left the deployment as-is to enable me to study this special case where obstructions influence how information passes around the network. In the next solution iteration, my ongoing work automates this measurement process, but for consistency during the remainder of this experiments, I configured each node to reconstruct the paths shown in green in Figure 4.5.

4.3.3 Machine Learning-based Parking Fingerprinting

Feature Selection

One challenge I faced is that for any given moment of time, one or more advertisements may not be observable from any given node. In order to collect observations from all nodes that span a period of time, I averaged all RSSI value heard by a beacon for each node, within each 10 second period over the 5 minute fingerprinting period for each parking space. If no RSSI values exist during this time window, a value far off of the scale was used (-10,000). To expand this initial feature space, additional features were added such as the average and median RSSI value for the entire 5 minutes for that node/space combination, variance, standard deviation, as the number of beacons observed per second. This resulted in feature vectors of 70 values, for 30 total time slices within the 5 minute fingerprinting period for

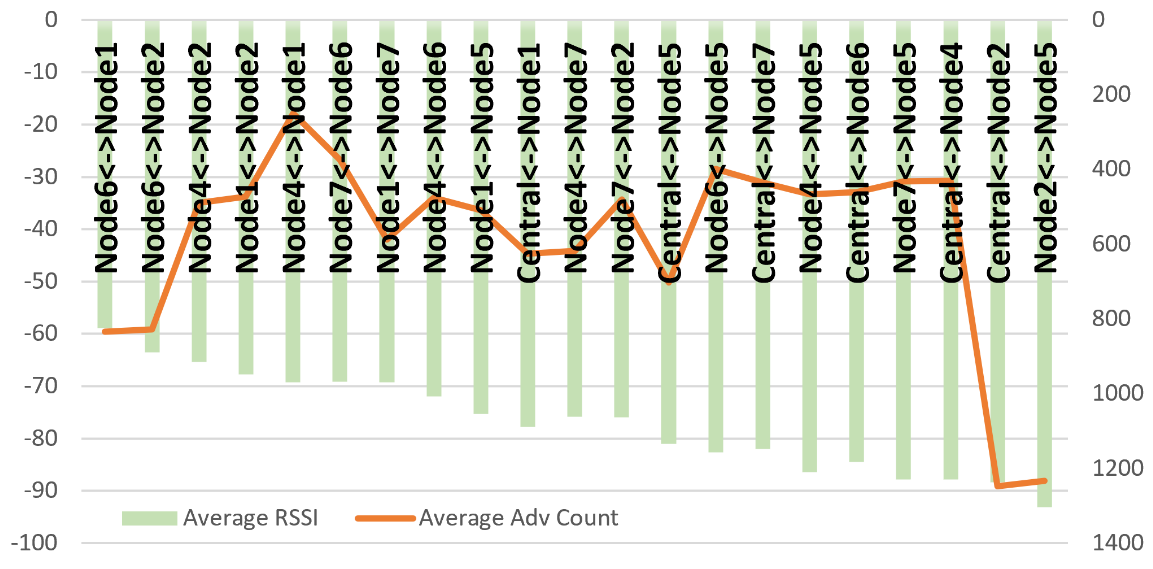

Figure 4.4: Inter-Node RSSI, Advertisement Counts

each space.

Model Training and Evaluation

Although unable to fingerprint all of the 105 spaces in the parking lot, due to a consistently parked vehicle present (in three spaces of the lot) during all of my experiments, I did ensure that none of my in-vehicle experiments used these non-fingerprinted spaces. Once I collected my RSSI data and created the feature set, I trained and tested several popular supervised learning classifiers use for prediction: Decision Tree (J48), Random Forest, Naive Bayes, Support Vector Machines (SMO), and k-Nearest Neighbors (k=1). I used *Weka* [112] as my machine learning toolkit, with default settings and 10-fold cross-validation. I summarize our training results in Table 4.1, showing average percentage of correctly identified spaces for each algorithm used and the average Receiver Operating Characteristic (ROC) area across all spaces and across only High Value (H.V.) spaces. High Value spaces carry rules for usage, such as those requiring handicapped placards and short-term (60 minute) spaces near

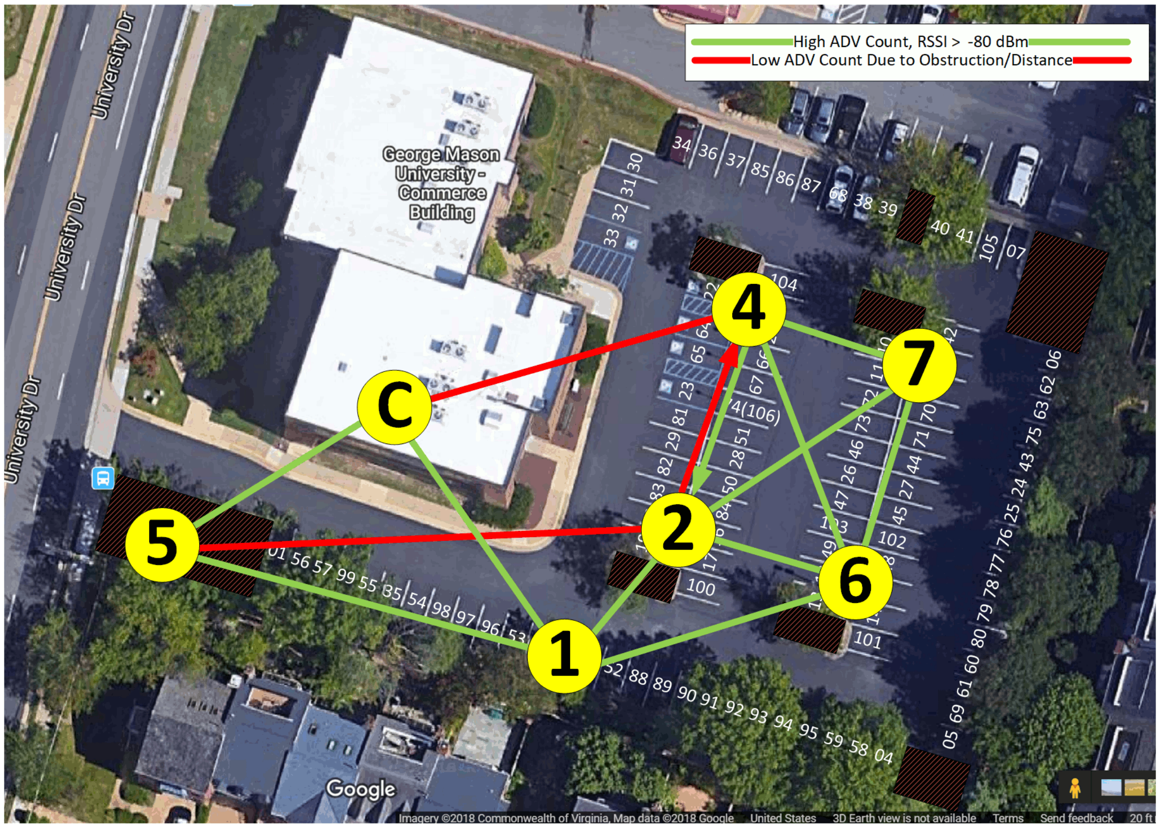
High ADV Count, RSSI > -80 dBm
Low ADV Count Due to Obstruction/Distance
George Mason University - Commerce Building
University Dr
C
1
2
4
5
6
7

Figure 4.5: Node, Space Layout, w/ Average BLE Advert RSSI

the building entrance. These results show that the Random Forest classifier performs best, correctly predicting the parking space occupied an average of 89.1111% across all spaces, and 92.9250% for H.V. spaces. This was a significant result as I had not yet performed model tuning, data filtering, or detailed investigation into the selected feature set. Further examination of per-space accuracy results showed that for some spaces we achieved 100% or near-100% prediction accuracy, while others such as those that lined the east end of the parking lot near a residential community that contained between 25 to 35 WiFi hotspots providing uncontrolled interference, sometimes performed poorly. I then performed a parameter tuning exercise on the Random Forest model and produced an optimized model with a slightly more accurate average classification of 90.6984% across all parking spaces, and 93.3500%t across H.V. spaces. I summarize the per-space results in Figure 4.6 showing the True Positive (TP), Precision, and ROC Area metrics for each space.

Table 4.1: Trained Classifier Results (10-fold CV)

Algorithm	All Training Data		H.V. Space Samples	
	%Correct	ROC Area	%Correct	ROC Area
kNN(1)	48.0635	0.736	43.7750	0.7138
SMO	25.6508	0.805	20.4125	0.8109
J48	77.6825	0.909	80.4250	0.9073
NB	21.873	0.910	18.3375	0.9165
RF	89.1111	0.997	92.9250	0.9993
RF (i=2000)	90.6984	0.998	93.3500	0.9996

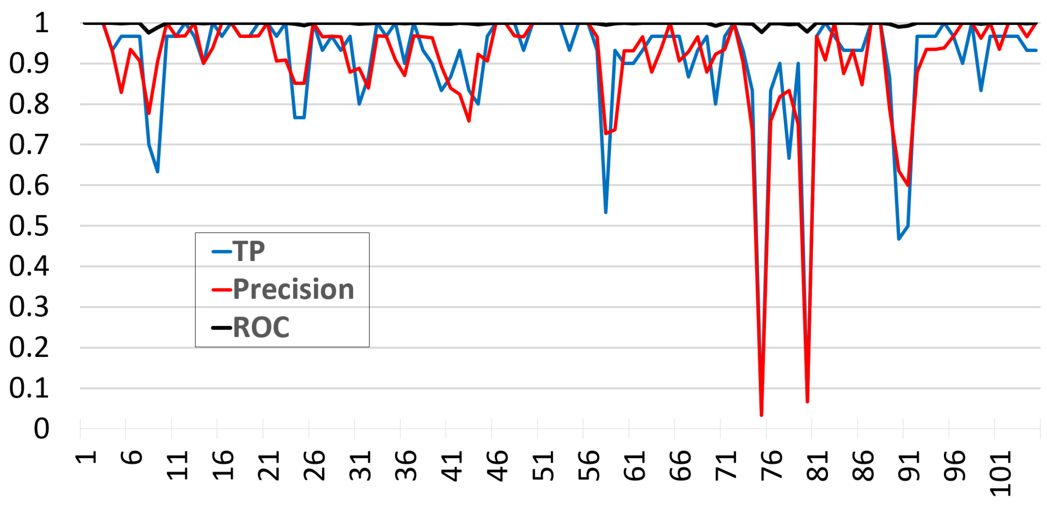

Figure 4.6: Per space True Positive, Precision, and ROC Area

”In-Car” testing

I further tested the accuracy of my model by placing the beacon in a two passenger car, after parking the car in randomly selected spaces. Each measurement of the in-car beacon was taken in an identical way to space fingerprinting: 5 minute sampling resulting in 30 total measurements per space. Then I used my model to classify these 30 measures for each space. As expected, results shown in Table 4.2 had a lower prediction accuracy than my fingerprinting experiment. To quantify the inaccurate of prediction with respect to the distance between predicted space and true space, I examined the map in Figure 5 and drew a straight line between the approximate center of each space (predicted and actual), and counted the number of spaces the line recognizably overlapped. I chose this method due to the fact that the parking lot was not a uniform grid, and I did not fingerprint locations in the lot that were not marked as parking spaces. Although this evaluation method lacks rigor, it does provide me with an approximation of how inaccurate our prediction errors were. Outcomes of exact matches, 1, 2, 3, or more than 3 space differences are shown in Table 4.2. Some spaces such as 12, 17, and 31 had many exact matches. Others were less accurate resulting in predictions of 3 or more spaces between predicted and true location. The percentage of predictions for the un-optimized model was as follows: exact spaces 5.33%, 1 space away 19.00%, 2 spaces away 15.67%, 3 spaces away 24.17%, and more than 3 spaces away 35.83%. Repeating this exercise using the optimized model is shown in Table 4.3 with similar results: exact spaces 7.00%, 1 space away 19.17%, 2 spaces away 15.67%, 3 spaces away 24.00%, and more than 3 spaces away 34.17%.

Table 4.2: Predicted "In-Car" Results. Default Model

Space	Error(spaces)					Space	Error(spaces)					Space	Error(spaces)				
	0	1	2	3	>3		0	1	2	3	>3		0	1	2	3	>3
12	9	15	0	6	0	32	2	14	0	6	8	64	1	0	1	24	4
17	8	21	1	0	0	33	0	0	0	0	30	65	0	6	0	8	16
22	1	0	1	23	5	34	0	11	7	10	2	69	0	0	5	2	23
23	0	0	1	7	22	37	1	23	4	0	2	81	0	3	6	0	21
25	1	2	13	10	4	40	0	2	8	6	19	95	0	1	9	6	14
30	0	4	3	20	3	44	0	2	14	5	9	98	0	1	9	7	13
31	9	9	7	5	0	52	0	0	10	0	20						

Table 4.3: Predicted "In-Car" Results using optimized model

Space	Error(spaces)					Space	Error(spaces)					Space	Error(spaces)				
	0	1	2	3	>3		0	1	2	3	>3		0	1	2	3	>3
12	11	14	0	5	0	32	2	15	1	8	4	64	2	0	1	24	3
17	9	18	3	0	0	33	0	0	0	0	30	65	0	3	0	15	12
22	0	0	3	23	4	34	0	9	7	12	2	69	0	0	6	2	22
23	0	0	1	6	23	37	1	24	3	1	1	81	0	3	4	0	23
25	2	1	10	12	5	40	2	2	4	6	16	95	0	1	8	3	18
30	0	6	8	15	1	44	0	7	13	0	10	98	0	1	8	8	13
31	13	11	2	4	0	52	0	0	12	0	18						

4.3.4 Measurements Over Mesh Network

I tested the combined mesh network solution and localization solution. We used the same parameters from the *in-car* experiment, but collected the RSSI measurements at the central node after they were transmitted over the mesh network. Over the course of this work, I began to experience significant performance and reliability issues with the built-in dual mode Bluetooth controller on the Raspberry Pi. I was not able to maintain a stable mesh network until I ran my experiments with replacement Bluetooth adapters (discussed above).

Table 4.4: Predicted "In-Car" Results, Mesh Collected

Space	Error(spaces)					Space	Error(spaces)					Space	Error(spaces)				
	0	1	2	3	>3		0	1	2	3	>3		0	1	2	3	>3
12	0	5	2	23	0	32	22	3	0	5	0	64	0	0	16	5	9
17	0	0	0	30	0	33	0	25	0	0	5	65	0	0	0	6	24
22	0	20	0	0	10	34	0	0	0	30	0	4	0	0	2	0	28
23	0	0	0	0	30	37	0	0	0	30	0	81	0	0	0	6	24
25	0	0	0	24	6	40	0	4	1	0	25	95	0	0	0	0	30
30	1	0	23	0	6	44	0	15	0	11	4	98	0	0	28	0	2
31	0	27	0	3	0	52	0	0	1	0	29						

4.3.5 Compensating for Attenuation

I learned from my in-car experiments that placement of the beacon inside a vehicle resulted in less accurate prediction, due to attenuation and variance in physical location of the beacon. I studied a simple mitigation for this attenuation by increasing each measured RSSI value to compensate. I took my in-car testing data (both through the mesh network, and without the network) and added 3, 6, and 9 dBm to each RSSI value and repeated my prediction using the optimized random forest model. I show these results in Table 4.5.

Table 4.5: Predicted "In-Car" Results, Attenuation Corrected

+dBm	Error(spaces) % (via mesh)				
	0	1	2	3	>3
0	7.00(3.83)	19.17(16.50)	15.67(12.17)	24.00(28.83)	34.17(38.67)
3	12.67(2.17)	20.33(19.00)	16.67(9.67)	19.50(31.83)	30.83(37.33)
6	10.33(1.00)	19.50(25.33)	15.83(11.50)	20.50(30.00)	33.83(32.17)
9	5.17(5.67)	10.83(24.00)	12.50(17.50)	29.17(22.00)	42.33(30.83)

4.4 Conclusion

My solution addresses the parking lot occupancy detection problem by providing a low power and low cost localization and sensor data transport solution using Bluetooth. My results show that this solution has promise at providing per-space occupancy detection, allowing for a variety of parking applications. When combined with payment, registration, and other infrastructures, parking lot owners will be able to free themselves from costly sensor network and ticketing systems, enabling a seamless parking experience for its users.

The work presented in this chapter was published in *Secure Outdoor Smart Parking using Dual Mode Bluetooth Mesh Networks,* Paul Seymer, Duminda Wijesekera, Cing-Dao Kan. In Proceedings of the 89th IEEE Vehicular Technology Conference (VTC'19-Spring), April 28 May 1, 2019, Kuala Lumpur, Malaysia [113].

Algorithm 2 Mesh Network Operations

```
procedure join_network(service_UUID)
    Enable ”piscan” on bluetooth adapter
    Start service advertisements for service_UUID
    Start locator beacon
    while not_joined_to_network do
        for each Incoming RFCOMM connection do
            Create, encrypt, and sign new authentication message
            Send message over RFCOMM connection
            Wait for network authentication reply
            if network authentication message is valid then
                add connection details to neighbor_node_list
            Terminate connection
procedure discover_new_nodes(service_UUID)
    while network_incomplete do
        observable_nodes[] ← find˙services(service_UUID)
        for each node in observable_nodes[] do
            node˙RSSI = scan˙for˙RSSI˙value(node)
            if node˙RSSI ¿ -70dBm then
                Establish RFCOMM connection
                Wait to receive authentication message $a$
                flood_message($a$)
                Receive central auth. decision $d$ and token $t$
                if $d$ = authentic then
                    Send $t$ to node
                    Add connection details to neighbor_node_list
                Terminate connection
procedure flood_to_neighbors(message $m$)
    for each node in neighbor_node_list[] do
        Append $m$ to outgoing queue for node
    seen_message[hash(m)] = True
procedure send_messages(message $m$, fairness_limit $n$)
    for each queue in outgoing_queues do
        Connect to queue.neighbor_node_name
        Pop and send first $n$ messages in queue
        Terminate connection
    Start sleep timer
procedure receive_message(message $m$)
    if $m$ in seen_message[] then
        discard $m$
    else
        if $m$ is for me then
            $p$ ← decrypt_and_validate($m$)
            if $p$ != null then
                process $p$
            else
                discard $m$
        else
            flood_to_neighbors($m$)
```

Chapter 5: Smart Parking Zones using Dual Mode Routed Bluetooth Fogged Meshes

Contemporary parking solutions are often limited by the need for complex sensor infrastructures to perform space occupancy detection, and costly to maintain ingress and egress parking systems. For outdoor lots, network infrastructure and computational requirements often limit the availability of innovative technology. I propose the use of Bluetooth Low Energy (BLE) beacon technology and low power sensor nodes, coupled with sensible placement of computational support and data storage near to the sensor network (using a *Fog* computing paradigm) to provide a seamless parking solution capable of providing parking maintainers with accurate determinations of where vehicles are parked within the lot. My solution is easy to install, easy to maintain, and does require significant alterations to the existing parking structures.

In this chapter, I present an improvement on the work outlined in the previous chapter, by enhancing the design and operation of both my localization solution and the mesh network that implements it, and laying a foundation for integration of this work with a multi-modal scheduler (found later in this dissertation). I augment the problem space and motivations in Section 5.1. I introduce routing, zoned-based occupancy detection, vehicle ingress/egress detection, and other features in Section 5.2. A lengthy series of experiments with results that motivated my final system design is outlined in Section 5.3. Conclusions drawn from the experiments of this chapter is included Section 5.4.

5.1 Introduction

Modern *smart* parking solutions have many problems. First, many suffer from the need to install per-space sensors to provide accurate occupancy detection. Others require too much user interaction, such as using smartphone apps to scan space identifiers. Other solutions use isolated ingress and egress payment interaction points, entailing costly automation. While the original intent of technology deployments to parking lots was to make management cost effective and usable, contemporary commercial solutions continue to use old paradigms and fail to achieve seamless, low cost, low resource intensive but service rich parking management solutions.

Parking lots, particularly outdoors often reside outside wireless (or even wired) network boundaries. Additionally, these spaces have minimal power to support lighting and other basic infrastructure. Suggesting a complicated technology stack for deployment to these lots will replace one problem with another and become cost prohibitive. Shifting to *cloud* computing technology offer the ability to take advantage of high computational ability without the overhead and expense of maintaining it, nor paying for its use when idle. One major drawback of this technology is the dependence on a reliable, and often large, network connection, a rarity in parking lots. *Fog* computing, counters such dependencies by shifting computation and storage closer to the parts of a network that need them. In the case of sensor networks, like the ones that are deployed to smart parking lots for occupancy detection, *Fog* computing suggests that storage and computational capability be located as close to the network's sensor capability as possible. This is a challenge, as the devices that perform sensor activities are often low power, and as a result have extremely limited storage and computational capability. What is needed is a low cost, low power solution with minimal user interaction to function. Doing so requires eliminating traditional ingress and egress payment and tracking support and replacing it with something completely seamless that does not interfere with vehicle ingress or egress or to/from the lot. Occupancy detection must not have a one-sensor-per-space requirement. Deployment of the solution should not require significant alteration of the existing lot. I propose using low cost, low power, Bluetooth Low Energy (BLE) sensors and BLE beacon technology to create a fully automated and seamless parking experience. In this chapter, I improve upon the work reported in the previous chapter and my publication [113] in several ways:

Improved localization, simplified machine learning model (with a smaller feature vector), significantly lower mesh network overhead during operation, and adaptation of Fog/Cloud concepts to provide on-demand and off-network computation (thereby simplifying on-lot computation demand). I compare this to a solution that use an *Edge Computing* object recognition camera as a BLE vehicle attestation to the solution presented in this chapter.

5.2 Proposed Solution

An overview of my solution is shown in Figure 5.1 where the target parking lot is equipped with Bluetooth sensor nodes that measure radio strength from vehicles equipped with BLE beacons. These sensor nodes form an authenticated mesh network, based off of real-time radio signal strength between them to allow the network to self-form and adapt to changing conditions like radio interference and node failure. When vehicles enter the lot and park, the sensor network records Received Signal Strength Indicator (RSSI) values and relays these values over Bluetooth Classic (RFCOMM connections) to a central node. This central node assembles the sensor data together and performs a space prediction activity using a Random Forest machine learning model, trained against a radio map created at installation time. This model was trained offline to simulate an activity we see as occurring in a real-life deployment in the cloud, however the size of parking lots and the low-density of sensor nodes is such that training this model could have been performed on a host within the fog network. My solution takes advantage of *edge computing* and *fog computing* by configuring the sensor nodes with sample rates and other data pre-processing steps aimed at reducing network demand, performing execution of the space prediction activity on a power unrestricted fog network located within a nearby office building, and offloading lot-agnostic or global activities such as trending and payment processing to the cloud. Here, all computations needed to make a space determination is located on the fog network, sparing the connection to the cloud from this additional demand. External services that can benefit from this data can interact directly with the cloud, sparing the fog and sensor networks

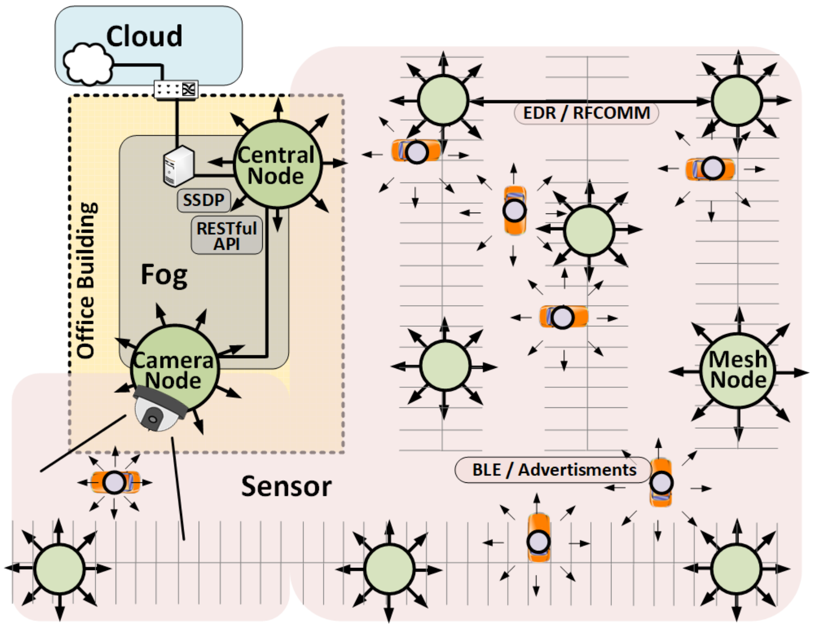

Figure 5.1: Solution Architectural Overview

from supporting those processes. This arrangement is shown in Figure 5.2. In addition, our system detects vehicles that enter and exit the lot, and authenticates them. I do so with two solutions, one based purely on Bluetooth radio, and another that combines the radio with an object recognizing camera deployed to an ingress point at the parking lot. I experimentally contrasted both solutions.

5.2.1 Parking Space Occupancy Detection

My space occupancy prediction model is an evolution of prior work [113], where I used RSSI values from Bluetooth beacons, measured at multiple locations within a parking lot to train a Random Forest machine learning classifier. This section outlines significant changes and improvements to that work, and new features provided by our current system.

Zone Based Occupancy Detection

Preliminary experiments were performed with a per-space occupancy detection goal. Results were accurate above 90% for my training set for all spaces in the lot, with less accurate spaces

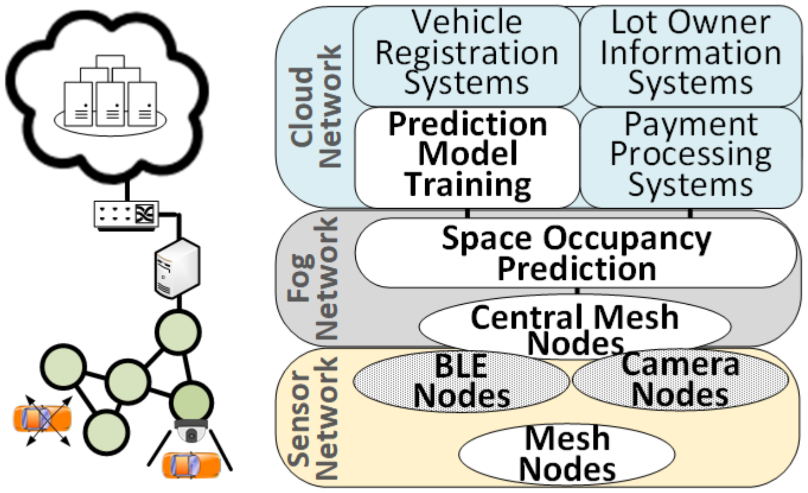
Cloud Network
Vehicle Registration Systems
Lot Owner Information Systems
Prediction Model Training
Payment Processing Systems
Fog Network
Space Occupancy Prediction
Central Mesh Nodes
Sensor Network
BLE Nodes
Camera Nodes
Mesh Nodes

Figure 5.2: Cloud, Fog, and Sensor Network Computational Model

along the south and south east corners of the lot (marked as orange circles in Figure 5.3). Suspecting that this was due to signal interference in the 2.4GHz range from the surrounding residential area, I considered zoned-based occupancy detection instead of individual spaces as a mitigation technique, as parking lot owners will most likely not want a pricing model that differs between spaces that border one another, except for regions such as near a building entrance, or on a level of a parking garage. Consequently, I use zones consisting of contiguous spaces as shown in Figure 5.3.

Sensor Node Placement

To avoid deploying structures that impede traffic flow or compromise existing spaces, I used existing lamp posts to mount nodes at heights of 8 feet or more. Two sensor nodes are located within the nearby building that belong to both the BLE sensor network and the fog network, to act as a network bridge to allow for sensor data to be used off-network for payment, etc., as shown in Figure 5.3.

Preliminary experiments in prior work with a 7 node network resulting in incorrect classifications primarily involving spaces in zones 2 and 3 in both the training set and the in-vehicle testing. Additionally, the formed mesh routed all traffic through node 1, creating

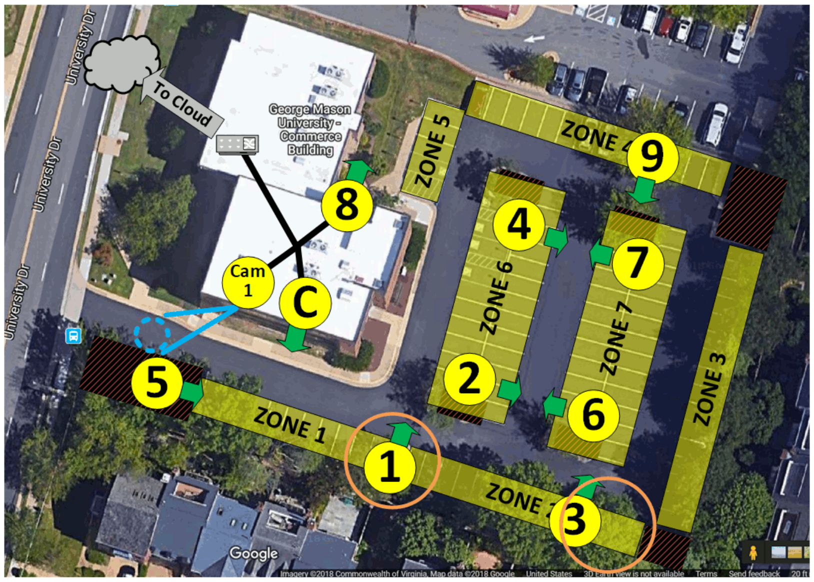

Figure 5.3: Parking Space Zone Map.

a single point of failure and a potential bottleneck, should the network become saturated. As a mitigation, I deployed two additional sensor nodes to address inaccurate predictions for nearby spaces in zones 2, 3 and 4, and a third node on the other side of the building (near node 4, not shown in the figure) to allow for multiple mesh paths to be formed with the nodes inside the building.

Radio Fingerprint Feature Selection

My first features included many statistical measures of RSSI values and their changes over time. After consequent analysis, I removed features with minimal positive contributions to the eventual determination. An example is the median count of RSSI beacons seen over discrete time intervals. In practice, this metric varied greatly once a beacon was placed inside a vehicle, creating higher errors in prediction. Such issues are addressed in Section 5.3.1. Temporary physical obstructions cause signal attenuation or in some cases prevent a beacon measurement entirely. To mitigate this effect, I used the highest observed RSSI value per node per space, as this gave the best result. My experiments described in Section 5.3 support this choice.

Bluetooth-Only Vehicle Identification and Tracking

Due to the range of Bluetooth, each space is observable from several sensors in our network. When a vehicle enters the lot and parks, its unique beacon broadcast will be detected as part of the localization process. If that beacon has not been seen for a period of time, I assume that the vehicle has exited the lot. The process for vehicle detection and identification is shown in Algorithm 3. Initialization is run when the network is started, shown in lines 1-7. This algorithm maintains a data structure of observed beacons (lines 8-12), launches a thread (line 7) to observe this data structure (line 16) and determine when a vehicle has exited the lot (line 18). If an exited vehicle is detected, a notification is sent (line 19) so that external functions could be run to initiate payment and other business functions based on lot occupancy.

Algorithm 3 BLE Only Vehicle Identification Procedures

```
1: procedure initialize()
2:     parked_records ← {}                                        ▷ init history data structure
3:     veh_entered ← {}                                                  ▷ init enter records
4:     veh_exited ← {}                                                    ▷ init exit records
5:     detect_veh_kill_flag = False
6:     beacon_measure_intvl = 5min
7:     start thread detect_vehicle_exit()
8: procedure detect_veh_enter(each received beacon b)           ▷ called for each beacon central
   receives
9:     if b.veh_id not in parked_records then
10:        create entry for b.veh_id in parked_records
11:        notify mgr new vehicle parked (b.veh_id)
12:    parked_records[b.veh_id].last_seen ← b.time
13: procedure detect_veh_exit()
14:    while detect_veh_kill_flag = False do
15:        n ← time.now()
16:        for each veh_rec ∈ parked_records do
17:            r ← parked_records[b.veh_id].last_seen
18:            if (n − r) > beacon_measure_intvl then
19:                notify mgr new vehicle exited (b.veh_id)
20:        sleep (beacon_measure_intvl)
```

One potential gap in my solution is the assumption that a beacon found within a lot is indeed inside a vehicle, and that all vehicle entering the lot have an active beacon. While my smart parking solution will only be deployed to lots that have 100% participation by all vehicles, I am sensitive to the potential for a malicious vehicle operator entering the lot without powering their beacon. To my solution, this would not appear as a space occupying vehicle at all, but I explore a solution to this in the next section.

Camera-based Vehicle Detection and BLE Beacon Attestation

To remedy the potential for vehicles to evade detection by disabling their beacons, I explored the use of a low cost, low power object recognition camera located at the entrance/exit points of the lot. I selected the Jevois-A33 Smart Camera [114], an all-in-one camera and CPU with a USB interface, chosen primarily due to its low price point (under $60 USD), and the ability to recognize objects out-of-the-box with little configuration. I configured the camera to recognize vehicles using the Jevois Darknet YOLO module [115] which employs YOLOv3 [116], a neural network that quickly detects objects based on a single pass across an image obtained from the camera. I connected the camera using a USB to one mesh node and used the camera's USB interface to sent serial messages from its classifier to its built-in logging module. The node connected to the camera was also equipped with a BLE receiver and observe nearby beacons and discover if the vehicle viewed by the camera has a valid active beacon - assuring that the specific beacon observed does indeed correspond to a vehicle. While a simple solution, I view this as a preliminary step for augmenting my BLE-only solution with additional low cost low development attestations. I conducted feasibility experiments for this implementation described in Section 5.3.3, and contrast this solution with the BLE only solution in Section 5.2.1. I outline how this arrangement functions in Algorithm 4. Algorithm initialization occurs (lines 1-6), creating a data structure, populated by two watcher threads that observe and record BLE beacons and camera-based object detections. Additionally, two thresholds are set specifying the RSSI value required by a beacon to indicate it is near to the camera node (line 5), and the maximum time difference between a camera recognition event and a matching BLE beacon (line 6). When a vehicle passes near the camera, it is recognized by the camera's object recognition system and logs a detection event through the serial connection to its node. Additionally, the vehicle's beacon is observed by the BLE receiver on the node. As BLE beacon detection and camera-based object recognition are independent processes, a step is required (lines 11-14) to match them based on similar time-stamps defined by my threshold (line 6). Section 5.3 experimentally shows that camera recognition takes a few extra seconds to process when compared to the BLE detection. This occurs because the radio broadcast of slow moving vehicles reaches the camera node's BLE sensor before the vehicle comes into the camera's view. Once the match occurs, this attestation event is passed back to the central node for use (line 16) or an error is sent if a beacon or camera event are missing (line 18). Lastly, my experiments show that the camera does not function at night when there is insufficient light to recognize objects correctly, so this algorithm is only run during daylight hours (lines 7-8).

Algorithm 4 Hybrid BLE/Camera Vehicle Identification Procedures

```
1: procedure initialize
2:     Initialize datastructure $d$
3:     Start thread to record BLE beacons
4:     Start thread to record events from camera
5:     $bt \leftarrow rssi\_threshold$ (set to -70 dBm)
6:     $td \leftarrow event\_time\_delay$ (set to 5 seconds)
7:     if is nighttime then
8:         Exit, as camera does not function at night
9: procedure attest_veh_thread(datastructure $d$)
10:    if $d$ has new event $e$ then
11:        if $e$ is a new Beacon event $b$ then
12:            Find matching Camera event $c$
13:        if $e$ is a new Camera event $c$ then
14:            Find matching Beacon event $b$
15:        if both $c$ and $b$ do not exist then
16:            Send error (-,$b$ or $c$,-) to central node
17:        else
18:            Send attestation ($c$,$b$) to central node
```

5.2.2 Fogged BLE Sensor Meshnet

My solution is composed of software-identical nodes, assigned and configured to a specific roles: mesh network, sensor, in-vehicle, or camera. Any node can perform any role provided it is equipped with the necessary hardware, allowing deployment activities to place sensor nodes where they are needed for detection, and if needed, turn them into network nodes at

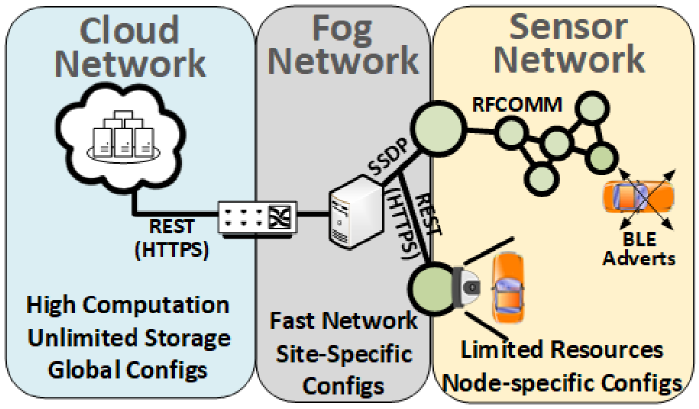

Figure 5.4: Fog Characteristics and Network Protocols

a later time. The mesh network is formed over Bluetooth *Classic* (EDR Mode), partially insulating the BLE-based occupancy prediction functionality from radio interference. Nodes that must communicate inside buildings and to upper network layers (i.e. Cloud) do so over the building's existing wired network to reduce the in-building mesh network density required to compensate for signal attenuation from walls and floors. As one of the nodes on this fog network was the central node, this wired network forms the basis for the internal boundary of our *fog* network. Once I determined the parameters would be used for the space prediction algorithm, I configured the sensor nodes to sample data and perform initial processing to reduce network traffic and computational requirements for the central node. This configuration at the sensor nodes forms the external boundary of our fog network, as computation and storage have been pushed out into what would traditionally be the sensor network, as shown in Figure 5.4. I found that the high degree of mesh density due to clusters of sensor nodes began to compromise the integrity of the BLE radio, in addition to being extremely wasteful with a high volume of redundant message transfers (a characteristic of message flooding). I explore this issue more in Section 5.3.2. As a solution, I deployed a simple route selection algorithm to maintain at most one outgoing connection per mesh node. To support node authentication, the network is formed from the central authentication authority outward, detailed in Section 5.2.2.

Self-Forming Authenticated Routed Mesh

Each node contains a configuration (using a *service* list) that specifies permissions to connect to EDR (Bluetooth) or Fog (wired Ethernet) networks. Node discovery and network construction operate differently for these networks, as each contains different protocols and independent physical layers. All configured nodes enter a network formation phase upon boot up and are required to authenticate with the central node prior to being allowed to join the network and send messages. To ensure that a path exists on the existing network back to the central node to support authentication, the network is formed outward from the central node, until every node in the network has a valid link. Major functions in the network construction process are shown in Algorithm 5. When a node boots up, *join_network* (line 1) is called, initiating discovery steps for a fog network (lines 2-8) or Bluetooth mesh network (lines 9-18). Initially, only the central node advertises its services (both fog and EDR Bluetooth mesh). Fog network advertisements use a stripped down SSDP service [113] implemented in Python using multicast group address 239.255.255.250, port 1900 (line 3). When services are found, the new node and the existing networked node are mutually authenticated (lines 5-6, procedure outlined in lines 19-27) over the networked node's RESTful API (lines 21-27). If authentication is successful, the new node launches its RESTful API (line 8) written in Python and Flask (over HTTPS on port 443) to enable bidirectional communication. For the Bluetooth Mesh network, we use a BLE beacon to facilitate node discovery (line 10), rather than the SDP service protocol I used previously [113], as current software support for RSSI measurements is limited to BLE. For each node discovered (line 11) that has a sufficiently strong signal (line 12), authentication (line 13, procedure outlined in lines 18-36) is performed. If authentication is successful, node connection information is stored (line 14) so connections can be made to send future messages.

I solve both the central path dependency for authentication and mesh density reduction (Section 5.3) problems for the Bluetooth based network with the introduction of a simple route generation algorithm (in procedure *join_network*) for the *edr* network. Only nodes with an RSSI value on their advertisements above -75dBm are considered for authentication. After nodes are authenticated, the new node creates only an outgoing message queue for the node with the best connection (e.g. the largest RSSI advertisement value) (lines 15-18). Additional nodes are stored (line 14), in case this node fails so that the network is maintained. This forces all nodes to have at most one outgoing message queue, and only send messages to a single networked node, thereby reducing the overall network density and creating forward routes for each node. Experiments that lead me to this design are described in Section 5.3. Fog connections are made over Ethernet, and do not have such a density reduction requirement. However I assume that the central node is in the same broadcast domain as all other fog nodes (preventing the central path dependency problem, as every node is observable from one another during the SSDP protocol). I realize many possible expansion of robustness for these solutions, but leave them as future work in this dissertation.

Reliably Encrypted Message Transfer

Each node pre-exchanges an AES and HMAC keys with the central node to support message encryption (AES) and signature based integrity checking (HMAC-SHA256), stored locally within each node. Messages are of two main types, authentication messages and parking system messages. Authentication messages are used to authenticate new nodes to the mesh network, and parking system messages support the camera, BLE space occupancy, and network housekeeping. Each message follows the same format (Figure 5.5) but differs in fields based on the type of message being sent. This format is limited to a single AES block size (128 bit), and begins with a 4 bit field indicating the message type (0000 for heartbeats, 0001 for BLE RSSI messages, 0010 for camera object recognition events, 0011 for message acknowledgements, 0100 for authentication related messages, and 0101 - 1111 reserved for future use.). This message type helps to ensure the message is properly parsed on the receiving end. The next field is a 16 bit node identifier, indicating which node sent the message. The remaining 108 bits contain the message text being delivered. This entire

Algorithm 5 RSSI based authenticated meshnet formation

```
1: procedure join_network
2:     if node contains "fog" service then
3:         do SSDP on Ethernet network (for 20 mins)
4:         for each fog node fn found do
5:             if auth_to_fog_network(fn) then
6:                 initialize message queues for fn
7:         if at least one fog node found then
8:             launch RESTful API listener (flask)
9:     if node contains "edr" service then
10:        perform BLE scan (for 20 mins)
11:        for each advert bn with matching UUID do
12:            if bn avg RSSI $\leq$ -75 dBm then
13:                if auth_to_ble_network(bn) then
14:                    known_nodes.append(bn)
15:        if at least one node bn found then
16:            broadcast BLE advertisements (for 20 mins)
17:            gw $\leftarrow$ bn with largest RSSI value
18:            Initialize message queues for gw
19: procedure auth_to_fog_network(node_info fn)
20:                                                  ▷ this is run by the new node
21:     authmsg $\leftarrow$ construct_authentication_message
22:     Open RESTful HTTPS connection to fn
23:     POST authmsg
24:     authreply $\leftarrow$ HTTP reply from POST
25:     if authreply is valid then
26:         return True
27:     return False
28: procedure auth_to_ble_network(node_info bn)
29:                                                  ▷ this is run by the new node
30:     authmsg $\leftarrow$ construct_authentication_message
31:     Open RFCOMM connection to bn
32:     Send authmsg
33:     authreply $\leftarrow$ Receive from bn
34:     if authreply is valid then
35:         return True
36:     return False
```

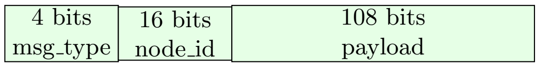

Figure 5.5: 128 bit Message (plaintext).

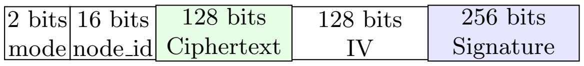

Figure 5.6: 530 bit Encrypted and Signed Message.

128 bit message is encrypted and signed using the originator's shared AES and HMAC keys. This grows the message to 530 bits (Figure 5.6). When authenticating new nodes to the network, non-central nodes will need to relay this message to the central node, and provide it's own encryption and signing to ensure secure message delivery. This additional pair of operations grows these messages to 802 bits as shown in Figure 5.7. To assist the central node in choosing the appropriate keys when validating encrypted and signed message, these message types are pre-pended with a plaintext node ID corresponding to the correct key identifier in its keystore (known to each node). Additionally, to assist with routing without requiring each node to set up its own inter-node channels, these message are also pre-pended with a 2 bit mode identifier, indicating if the message is for the central node, or for one of the sensor/relay nodes. Each of these pre-pended fields is included in the inputs to the signature hash function. While these fields make each message slightly larger, the use of these shortcuts greatly improved ease of message handling and routing implementation, and reduces encryption/decryption related computational demand on the low-power sensor nodes. Parking system messages are always routed to the central node. When a valid message is sent to the central node, an encrypted and signed reply message is sent back to the originating node, containing an identical sequence number for the original message. If a node does not receive an acknowledgement within a timeout period, it resubmits the same message. This arrangement effectively doubles the message load on the network, and I leave optimization of this to future work.

2 bit mode	16 bit node id 1	128 bit ciphertext	128 bit IV	256 bit sig. 1	16 bit node id 2	256 bit sig. 2

Figure 5.7: 802 bit Relayed Node Auth Message (encrypted).

5.3 Experiments

My experimental platform consists entirely of Raspberry Pi 3s, running Ubuntu Mate. Due to performance problems in using Pi's built-in Bluetooth adapter in dual-mode, I replaced it with an after market Bluetooth USB dongle StarTech [109] to regain expected Bluetooth performance. My code is written in Python using the pybluez library [107] and Bluez Linux Bluetooth stack [108]. For localization experiments, I constructed several datasets for training and testing. Initial space fingerprinting used a tripod mounted beacon (at the approximate height of a vehicle's rear view mirror) to create vehicle model independent fingerprints, but later found that attenuation difference when the beacon was placed in a vehicle resulted in a poor prediction model. I show these results in Section 5.3.1 and discuss my study of in-vehicle attenuation in more detail in Section 5.3.1. The complete lot was re-fingerprinted with the beacon inside a Nissan 370Z (*370_o* dataset). Additional vehicles (Nissan 350Z, Acura TL, and Nissan Rogue) were used to make partial-lot test datasets collected locally at each node, and over the mesh network. We refer to data collected from nodes and transmitted over the mesh network to the central node, as *mesh* datasets, and confined the size of the dataset to randomly selected spaces from each zone. To study the effects of the mesh network on localization, I also constructed datasets collected directly at each node (offline, after the experiment) in the absence of the mesh network. I refer to these as *offline* datasets, and are datasets are summarized in Table 5.1, with sample rates explained later in Section 5.3.2. *set1* spaces include 8, 20, 25, 27, 34, 36, 44, 53, 58, 64, 75, and 83. *set2* spaces includes 25, 27, 29, 34, 36, 38, 39, 56, 58, 60, 62, 64, 67, 70, 71, 75-77, 79-81, and 87-89. *set3* includes 1-2, 4, 6, 10-13, 18, 20, 24, 31-33, 35, 37, 39-45, 49-50, 53-55, 64, 66, 68, 72, 75, 77-78, 80, 82, and 90. *set4* spaces includes 25, 27, 29, 34, 36, 56, 58, 60, 71, 72, and 75-76.

Table 5.1: Dataset Space Composition (of 90 total spaces)

Name	Source	Spaces	Collect	Rate
Tripod	Tripod	all but 84-88	Offline	Constant
350_o	350Z	set2	Offline	Constant
370_o	370Z	all (1-90)	Offline	Constant
TL_o	Acura	set3	Offline	Constant
350_m1	350Z	set4	Mesh	Constant
350_m2	350Z	set1	Mesh	Constant
350_m3	350Z	set1	Mesh	60s sample
370_m	370Z	set1	Mesh	60s sample
Rogue_m	Rogue	set1	Mesh	60s sample

Prior experiments used the entire target parking lot [113]. The current experiments removed Zone 3 in Figure 5.3 due to un-relocatable physical obstacles in those spaces. I also sought the opportunity to deploy additional beacons compared to past work, allowing me to study the effect of additional RSSI data points on prediction accuracy, described in the next section.

5.3.1 Improved Prediction Model

In prior work we found that a Random Forest algorithm produced the most accurate predictions compared to similar classifiers, and I continue to use that algorithm in this work. However I made several improvements to our model based on what I learned in my experiments. I outline each of these improvements, along with my reasoning, in this subsection.

Initial Improvements to Model and Feature-set

I reduced the feature-set after experiments determined that only the maximum RSSI value within a time window consistently produced accurate results. This resulted in my feature vector shrinking from 38 in prior work, to 10 total features per time interval. After taking RSSI measurements for each space in the lot, I constructed a random forest model based on these 10 features, and summarize results in Table 5.2. Here I see three models trained with the tripod (tri) dataset, and used with TL_o, 350_o, and 370_o *in-vehicle* datasets. Columns *TP* and *R* show True Positive percentage and ROC area, respectively. The first model is a Random Forest model with default settings (no random tie breakers, iteration total of 100), with optimized models that are configured to randomly selected ties found by the algorithm, with total iterations set to 1000 and 1500 (respectively). All models use 10-fold cross-validation (CV). I use this same model evaluation and tuning strategy throughout this work, so that I can study the effect of model tuning on prediction accuracy. The results of this experiment show that while the model evaluates to 100% accuracy (in the second optimized case), it is only successful at predicting other vehicles' space occupancy by between approximately 8% and 14%, decreasing with my model optimization strategy. This is similar to the results from my previous experiments, and clearly requires improvement to make this a viable solution, even after introducing additional sensor nodes as I have done in this work.

Table 5.2: Initial Per-Space Tripod (tri) Model Results

Model (Dataset)	Default		Optimized 1		Optimized 2	
	TP	R	TP	R	TP	R
tri	99.88%	1	99.96%	1	100.0%	1
tri(TL_o)	11.23%	0.80	10.96%	0.87	10.44%	0.87
tri(350_o)	8.94%	0.79	8.94%	0.80	8.18%	0.83
tri(370_o)	14.05%	0.74	10.99%	0.81	10.91%	0.82

Zone Based Occupancy Detection

After closely analyzing the specific error cases in our last experiment, I surmised that I could improve my solution accuracy if I migrated from a per-space to a zoned based space occupancy strategy. In this case, the parking provider cares more about the area of the lot a vehicle is in than the individual space. In theory, this should improve my results in cases where the predicted space is near the true space. Using the same data, I divided up the lot into zones as defined in Section 5.2.1, and re-labeled my dataset accordingly and trained a new *zone-based* model. Experimental results are shown in Table 5.3. While this significantly improved the results, for some vehicles there was little improvement, and overall still below acceptable accuracy.

Table 5.3: Initial Zone Based Tripod (tri) Model Results

Model (Dataset)	Default		Optimized 1		Optimized 2	
	TP	R	TP	R	TP	R
tri	99.68%	1	99.68%	1	99.68%	1
tri(TL_o)	42.46%	0.83	38.77%	0.83	38.95%	0.82
tri(350_o)	16.06%	0.71	18.33%	0.77	18.48%	0.77
tri(370_o)	43.33%	0.81	43.93%	0.82	43.93%	0.82

In-Vehicle Effects on Beacon Attenuation

After re-examining my testing results, I noticed that there was an inconsistent decrease in RSSI values when comparing the tripod's signature with the in-vehicle signature. When the beacon is located on a tripod, it physically approximates the location of the beacon if it were mounted in a vehicle, however the effect of the vehicle's surrounding structure appeared to differ depending on the direction of the vehicle's orientation with respect to sensor nodes. For example, in every case I examined, the nodes facing the rear of the vehicle had the most significant effect on RSSI value, followed by nodes that faced to the left or right of the vehicle. Nodes in front of the vehicle had the least attenuation effect, approximately 1 RSSI in most cases. I quickly realized that the materials in the vehicle produced this effect, due to the orientation of my beacon (placed behind the vehicle's rear-view mirror). To the front and sides of the vehicle, transmission was most often through a single pane of autoglass. To the rear of the vehicle, it was often through seats, metal, and in most cases the rear-view mirror itself, resulting in as much as a 6 RSSI decrease. In prior work, I globally increased all recorded RSSI values to compensate for some of this attenuation, however, improvements were limited and inconsistent. current experiments and analysis add clarity to this result. I used a tripod for fingerprinting so that I didn't introduce a vehicle-specific bias into the model, however the consequence of inconsistent in-vehicle beacon attenuation became my limiting factor. My solution was to replace the tripod dataset with a model constructed from beacon measurements inside a vehicle. I selected the 370Z, due to convenience and availability of the vehicle, but acknowledge that the physical placement of the beacon may produce positive results only for similarly oriented vehicles (i.e. larger vehicles may need to have their own radio map). I discuss experiments with the *in-vehicle* model in the next subsection.

In-Vehicle Fingerprinting

After re-fingerprinting the lot with the 370Z, I trained new models, and repeated my per-space tests as they provide better insight into the cause of failures. Results are shown in Table 5.4. Here I see a similarly valid trained model with accuracy above 99% in all cases. Testing results for the Acura and 350Z also improved, in some cases significantly, although the need for zone-based detection was clearly still needed.

Table 5.4: Per-Space In-Vehicle (370) Fingerprinting Results.

Model (Dataset)	Default		Optimized 1		Optimized 2	
	TP	R	TP	R	TP	R
370	99.63%	1	99.56%	1	99.56%	1
370(TL_o)	42.11%	0.92	40.79%	0.95	40.79%	0.95
370(350_o)	12.78%	0.85	13.75%	0.89	13.19%	0.89

I also recomputed these results using the *zoned* scheme described in Section 5.3.1, and show results in Table 5.5. Here one see marked improvements overall, with the default model achieving the highest accuracy. All tests produced results with ROC area above 0.99. This degree of accuracy, while not perfect in all cases, is high enough that I proceeded with constructing the remainder of the solution around them. I proceeded with *mesh* experiments in the next subsection.

Table 5.5: Zoned In-Vehicle (370) Fingerprinting Results.

Model (Dataset)	Default		Optimized 1		Optimized 2	
	TP	R	TP	R	TP	R
370	99.78%	1	99.67%	1	99.67%	1
370(TL_o)	89.65%	0.99	89.39%	0.99	89.39%	0.99
370(350_o)	85.28%	1.0	79.86%	1	79.86%	1

5.3.2 Over-Mesh Experiments

My initial mesh experiments used a managed flooding algorithm without my routing feature, and used parameters that were extremely permissive in both node link creation and message volume which resulted in some areas of the mesh becoming very dense. This section outlines experiments of our self-forming mesh network, and improvements that lead to the sampled routing outlined in Section 5.2.

Self Forming Experiment

Recall the node deployment in Figure 5.3. When activated, mesh network nodes and the network formed with initial set of parameters, I noticed that the mesh was not well formed. The network became extremely dense and highly dependent on node 1. Additionally, no *EDR* nodes were observable by node 8, resulting in the fog network and *EDR* network being very isolated, routing exclusively through node 1. My solution was to deploy an *EDR*-only (no vehicle sensing configuration) node (called node 10) to a midpoint between node 8 and it's nearest (but not BLE observable) neighbor, node 4. This allowed me to provide a redundant network path, lessening the risk from the single path through node 1. I was also able to more easily manipulate connection thresholds due to this additional path, and discuss this further in Sections 5.3.2 and 5.3.2.

Prediction Over-Mesh Experiments

Once my mesh network was established, I performed beacon collection with the 370Z as my test vehicles (to temporarily remove error due to vehicle differences), and ran prediction tests using our current model while collecting data from the central node (since the network now existed to route data back to it). My preliminary results are shown in Table 5.6. I examined the data used in this result and message collection metrics and other instrumentation, and noticed large effects on the observability of beacons in the mesh configuration. As both *EDR* network and the beacon scanning shared the same radio, one could not function while the other was in use, resulting in fewer beacons being observed due to the load placed on the radio by the mesh network. There was also a very long delay in transmission of all beacons, an effect that increased over time due to the inefficiently managed flooding algorithm that were used. I also show the *zoned* version of our result in Table 5.6. While improvements were seen, this result was worse than data collected locally. After examining the data, I noticed that most of the data collected had gaps in observations from some nodes, causing the model to perform poorly.

I concluded from these results that I have to find a way to reduce the number of messages routed through the network. This would result in less message flooding, providing more on-radio time for the beacon receiver. I reprocessed my data to take the maximum RSSI value for the entire observation window (of 5 minutes), and use that as a single prediction case, instead of the 10-second time intervals I had been using, as such windows were too short for the beacon to be seen by any given node. I show this in the same table, as *Single Max.* This resulted in some improvement, but still below my non-mesh result. This suggested, however, that sampling beacons to reduce message volume, may result in a viable solution that would improve performance.

Table 5.6: Initial Over-Mesh Predication Results.

Model (Dataset)	TP	R
370 Per-Space (350_m1)	1.98%	0.60
370 Per-Space (350_m2)	9.72%	0.82
370 Per-Space (350_m1 Single Max)	8.33%	0.85
370 Per-Space (350_m2 Single Max)	8.33%	0.91
370 Zoned (350_m1)	46.33%	0.82
370 Zoned (350_m2)	61.94%	0.85
370 Zoned (350_m1 Single Max)	66.67%	0.68
370 Zoned (350_m2 Single Max)	75%	0.75

Message Down-sampling Experiments, effects on prediction

To gain insight into the effects of down-sampling without re-implementing my code, I took the in-vehicle fingerprinting set and compiled several training sets, each with different sample rates. I chose to favor heavily sampled sets, and chose 30, 60, 120, and 300 seconds as sample rates. Results are shown in Table 5.7. Here one sees that all sampling rates produced a highly accurate model, however with diminishing returns when passed the 60 second sampling rate. Then I re-implemented the mesh network and beacon sampling solution with this rate, and repeated experiments with additional test vehicles. The results are described in the next subsection.

Table 5.7: 370 Down-sampled Per-Space Results.

Model (Dataset)	TP	R	Model (Dataset)	TP	R
370 (370 30s)	99.78%	1	370 (370 60s)	99.78%	1
370 (370 120s)	98.33%	1	370 (370 300s)	97.78%	1

Sampled and Routed Mesh Results

Prior to implementing my sampling solution in my mesh network code, I attempted to resolve the radio use problem by introducing sender invocation delays in our mesh network, so that messages could queue up on a node, then *burst* across to other nodes when the sender established a connection. I performed an experiment with this parameter in place with the Nissan 350 (constructing the *350_m3* dataset). I show this results in the first row of Table 5.8. This resulted in fewer invocations of *EDR* connections, further reducing demand on the radio. I also noticed that the mesh network was introducing a large delay in the time the central node received all of the messages needed to make a prediction decision. While these were acceptable, they were still not ideal, as these delays will need to be increased as load increases. Such an arrangement will not scale with more beaconing vehicles entering the lot. Instead, I introduced a node selection and routing algorithm, Algorithm 5 in Section 5.2.2, implemented the sampled message scheme previously determined, and repeated mesh experiments with the Nissan 370, and Nissan Rogue (to add diversity with a larger vehicle than the smaller ones we had been using). I used the data from the *350_o* and assembled a sampled dataset with it's values, without repeating the over-mesh experiment with that vehicle. With these three mesh datasets I tested the previously constructed Nissan 370 Zoned model shown in the first three lines of Table 5.8.

I noticed that during my analysis that there was a slight loss in accuracy when I trained a model based on zoned labels, when compared to a per-space model that was relabeled after prediction. My final modification to the prediction scheme was to use the per-space model for prediction, but match each space with its correct zone label after prediction. Then I took the majority result as the zone our system indicates the vehicle to be located in. These results are shown in the bottom 6 lines of Table 5.8. Here one sees that the Nissan 350z was predicted in the correct zone for all of our test data, while the Nissan 370 and Nissan Rogue were correct 83.33% and 75% of the time, respectively. After examining results more closely, I determined that the lack of perfect prediction in Nissan 370 and Rogue were the result of 3 problematic spaces in the test set, in some cases resulting in a 50% (but not a majority) prediction. I seek to tune our model in the future to solve the problem with those particular spaces.

Table 5.8: 60s Sampled Mesh Results (Zoned only).

Model (Dataset)	Default	
	TP	R
370 Zoned(350_m3)	83.33%	0.90
370 Zoned(370_m)	77.08%	0.88
370 Zoned(Rogue_m)	77.08%	0.90
370 60s Post Zoned (350_m3)	95.83%	-
370 60s Post Zoned (370_m)	85.41%	-
370 60s Post Zoned (Rogue_m)	75%	-
370 60s Post Zoned Majority (350_m3)	100%	-
370 60s Post Zoned Majority (370_m)	83.33%	-
370 60s Post Zoned Majority (Rogue_m)	75%	-

I also examined the message delay, to further validate our choice to migrate from previous managed flooding solution to the lower-demand routing one. Figure 5.8 shows the links established in two of our dense mesh implementations used to create the *Nissan 350_m3* dataset (left) with the routed implementation used for the *Nissan 370_m* and *Nissan Rogue_m* datasets (right). The figure show that the network itself is much less dense with the routed solution. During my experiments, I also recorded the timestamp for when a message was created, and a timestamp for when the central node received that message. Then I measured the difference between these timestamps for both the dense mesh and routed mesh implementations, shown in Figure 5.9 sorted by message delay. The routed mesh shown included the 60 second beacon sampling, causing the lower bound of all messages to be 60 seconds. While the dense mesh produced message delivery for some messages much faster than the sampled and routed solution (during times of low load), its upper bound was unacceptably large (more than an hour in some cases). Consequently I selected the routed, sampled, implementation as our end solution, due to our results and the expectation that as this solution scales well to multiple vehicles on a lot.

5.3.3 Camera vs. BLE based vehicle Entrance/Exit Detection

Our last set of experiments compares the use of the object recognition camera based vehicle entrance and exit detection outlined in Section 5.2.2 with a BLE only solution. I conducted

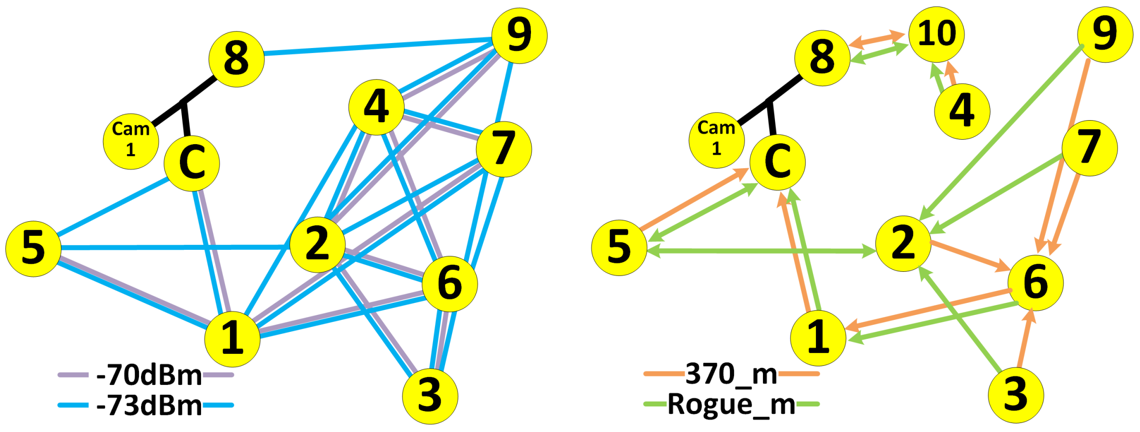

Figure 5.8: Dense Mesh vs. Routed Mesh.

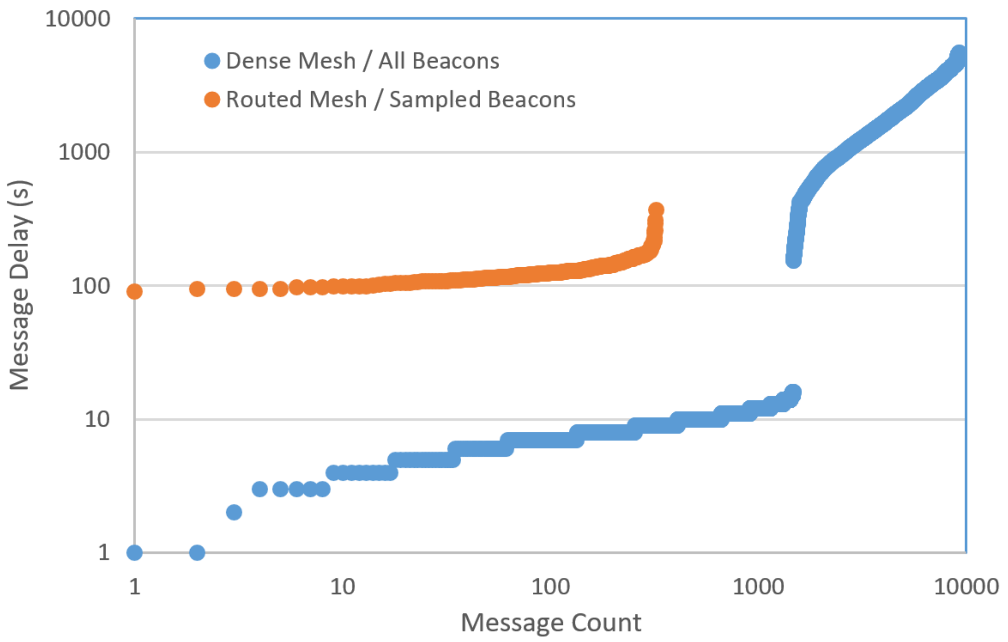

Figure 5.9: Message Delay Comparison.

a series of 10 instances where the Nissan 370Z was driven into and out of the lot at a constant speed of 10 mph, making one traversal in the camera's field of view, and its BLE beacon receiver range. I found that -70 dBm was a consistent value to use as a threshold for determining when the vehicle passed by this node. I summarize the number of beacons observed, the number of times the object recognition camera recognized the vehicle as a *car*, and the range of delay between when the beacon was seen, and when the camera produced a detection event over its serial connection in Table 5.9. This delay was computed by attempting to match a beacon event with a camera event, and measuring the time difference. There is a consistent delay of several seconds from when the vehicle passes in front of the camera, and a detection decision is made. I also see that both the camera and BLE beacon receiver observe a single vehicle multiple times. However I also acknowledge that if a vehicle travels too quickly it may not be detected by the camera, so duplicates are welcomed.

Table 5.9: Daytime Camera Recognition vs BLE ($\leq$ 70 dBm)

Direction	Beacons	Recog. Objects	delta t
Enter (10)	2-3	1-2 ”car”	0-3 sec
Exit (10)	1-4	1-3 ”car”	2-11 sec

We repeated this enter/exit experiment at night (with no light), and found that the camera produced no detection results, rendering it not usable in these conditions. I coupled this experiment with other mesh related parking experiments, including parking spaces very near to the camera node's beacon receiver. I show these results in Figure 5.10. On the left, one see now the camera node observes beacons for spaces far from it, while on the right one see observations for spaces that are close to the node. Although entrance and exit are consistently detectable by this node, there are false positives produced by beacon observations of the vehicle while it's parked, simply because those spaces are physically close to the node. I conclude from these experiments that the BLE solution is the only solution that will function 24 hours a day, and that this node was located too close to existing parking spaces to make it a good discriminator for vehicle entrance. For the chosen parking lot, the optimal solution for vehicle detection would use an aggregate from results from all nodes, rather than this particular ingress point, reducing the impact of false positives at any one node.

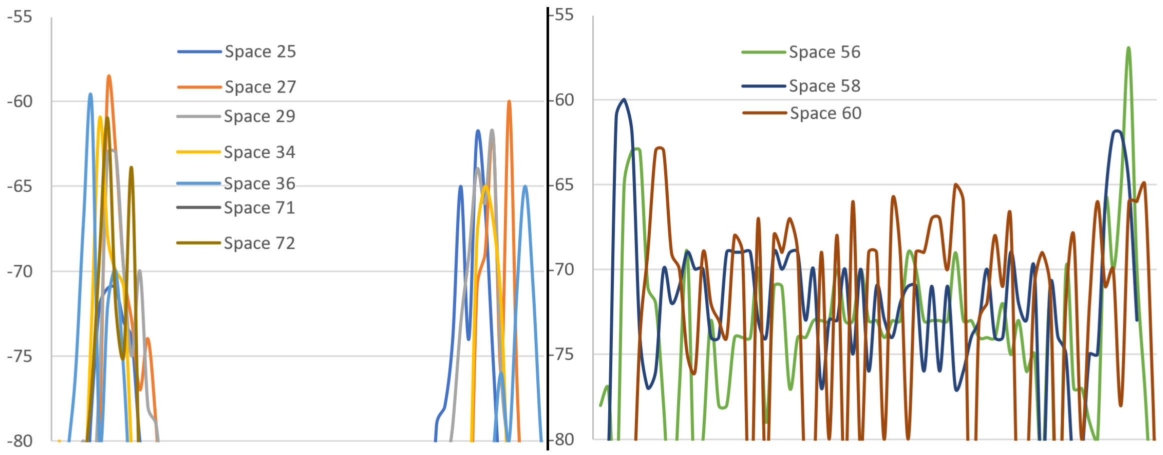

Figure 5.10: Night Entrance / Exit Detection.

5.4 Conclusion

I provide a reliable and accurate system to determine *zone-based* parked vehicle occupancy and convey that information to external networks using only Bluetooth radio. My *zone-based* solution is shown to be more accurate than per-space determinations. I designed a method to integrate camera-based object recognition into the system to provide additional vehicle attestation options, and performed an experiment-based comparison of this implementation to our BLE-only solution. Outdoor parking lot owners can use solutions like ours to cheaply and easily deploy seamless smart parking solutions.

The work presented in this chapter was published in *[Smart Parking Zones using Dual Mode Routed Bluetooth Fogged Meshes,* Paul Seymer, Duminda Wijesekera, Cing-Dao Kan. In Proceedings of the 5th International Conference on Vehicle Technology (VEHITS'19), May 3-5, 2019, Heraklion, Crete, Greece [117].

Chapter 6: Privacy preserving traveler tracking and on-demand multi-modal traffic informed tasking for autonomous vehicles near L'Enfant Plaza Metro

In this chapter, we present the outline of a comprehensive solution that builds on our past work with features that support a desired seamless multi-modal solution. We posit that the use of passenger provided destinations can be used to provide an on-demand AV staging when fixed routes are no longer efficient. We leverage Bluetooth beacon technology along with our custom beacons to authenticate passengers during ingress and egress of AVs, making boarding and disembarking seamless and fast. Additionally, this system allows for fast travel-relevant data transfer between passenger and the local vehicle, reducing external network overheads, as information rendezvous at the vehicle. We use simulations in SUMO [118] to make a preliminary study of the effects of these features and their parameters on traffic, allowing us to gain insight into when fixed routes are no longer efficient, and on-demand vehicle tasking is required. Additionally, we allow our tasking system to adapt to real-time road conditions and suggest alternative but walk-able pickup or drop-off locations to the traveler to better integrate with the surrounding traffic environment. When AVs are optimized in this way, travelers are encouraged to avoid using congestion-causing POVs to travel, avoiding parking and saving money, and cities experience less risk of negative traffic patterns due to AVs. Additionally, when vehicle tasking can be made dynamic, this solution can adapt during times of traffic spikes such as rush-hour, emergency conditions, or nearby on-site special events to mitigate spikes in expected traffic volume. The remainder of this chapter is outlined as follows: Section includes additional motivations and problem space descriptions. Section 6.2 describes the components of our solution. Section 6.3 describes our simulations. This Chapter concludes in Section 6.4.

6.1 Introduction

Contemporary public urban transit suffer from low ridership where urbanites favor more convenient options. Several factors influence these choices: long travel delays for subway and bus routes, low gasoline prices and minimal ride-hailing costs, complexity and lack of flexibility in multi-modal transit, etc. Although Autonomous Vehicles (AVs) have been suggested as mitigation for these trends, care must exercised so these vehicles do not contribute to traffic burden while simultaneously making travel easier, safer, and more cost effective to use. Public sentiment is a key factor in the success of a potentially disrupting introduction of technology. While these vehicles afford urban planners with options in introducing AV technology into mass transit ecosystems, such options much be focused on serving the needs of travelers and urban communities, while making travel easier, safer, and most cost effective to use.

I propose a system that connects AVs to travelers in a way that frees both from an expensive coordination burden. Vehicles are dispatched *on demand* emulating the ease of existing ride hailing services without adding to traffic-causing queuing by deploying AVs to the *last mile* of existing mass transit systems, using real-time traffic and road condition data to influence traveler-selectable pickup and dropoff locations while en-route. My system tracks passengers securely and anonymously during this journey to engage AVs only when and where needed, and in a way that cooperates with the surrounding environment. I replace the payment and ticketing infrastructure with two collaborative solutions: a smartphone and a software token, to reduce human burden and queuing that forms around payment interfaces. I use a motivating use case that connects the L'Enfant Plaza metro in Washington, DC to the 7^{th} Street Park near the DC Wharf. Lastly, I conduct simulations to evaluate aspects of my proposal and establish preliminary tasking and other parameter boundary conditions.

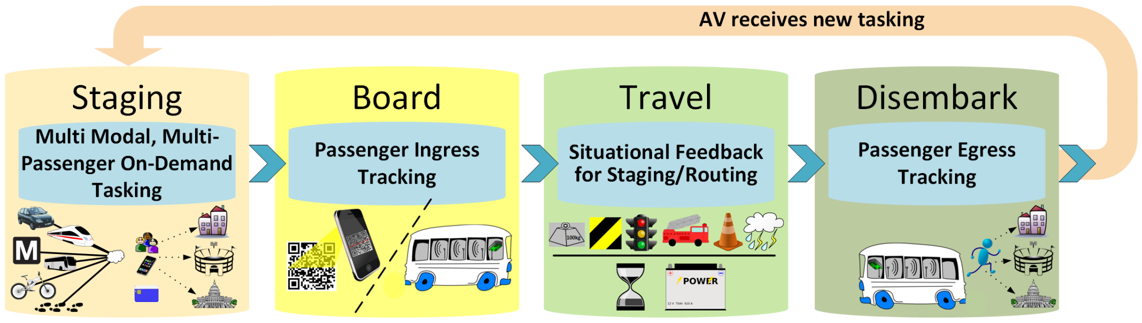
AV receives new tasking
Staging
Multi Modal, Multi-Passenger On-Demand Tasking
Board
Passenger Ingress Tracking
Travel
Situational Feedback for Staging/Routing
POWER
Disembark
Passenger Egress Tracking

Figure 6.1: Multi-Modal Travel/Passenger Tasking and Tracking (Overview)

6.2 Proposed Solution

I show an overview of my solution in Figure 6.1 grouped by the applicable phases of transportation management. The first phase is focused mostly on the travelers providing their destination information, and my system's efforts to task the AVs to serve them. The second phase covers the process of passenger rendezvous with AVs and the boarding process. The third phase describes a traveler's interactions with the AV during transit and incorporating AV's state (such as the power reserves, etc.) that assesses its fitness to continue to serve. The fourth and final phase covers passengers exiting the AV, and the vehicle receiving new tasking instructions from the staging system.

Figure 6.2 shows the deployed sensors, required network connectivity, computing backend services used for each of these stages, and their locations. I employ a hybrid of *fog* and *cloud* networks to reduce network overhead and computational delays commonly found in cloud-only approaches, without requiring a heavy technology footprint inside the AV.

6.2.1 Traveler Interface Application

I provide all traveler-dependent interfaces using a smartphone application to reduce the number of items carried by a traveler. These devices provide multiple radios, sensors and human controlled interfaces (such as screen, keyboard, speaker, microphone) making them the ideal platform for provisioning services such as the one described in this chapter.

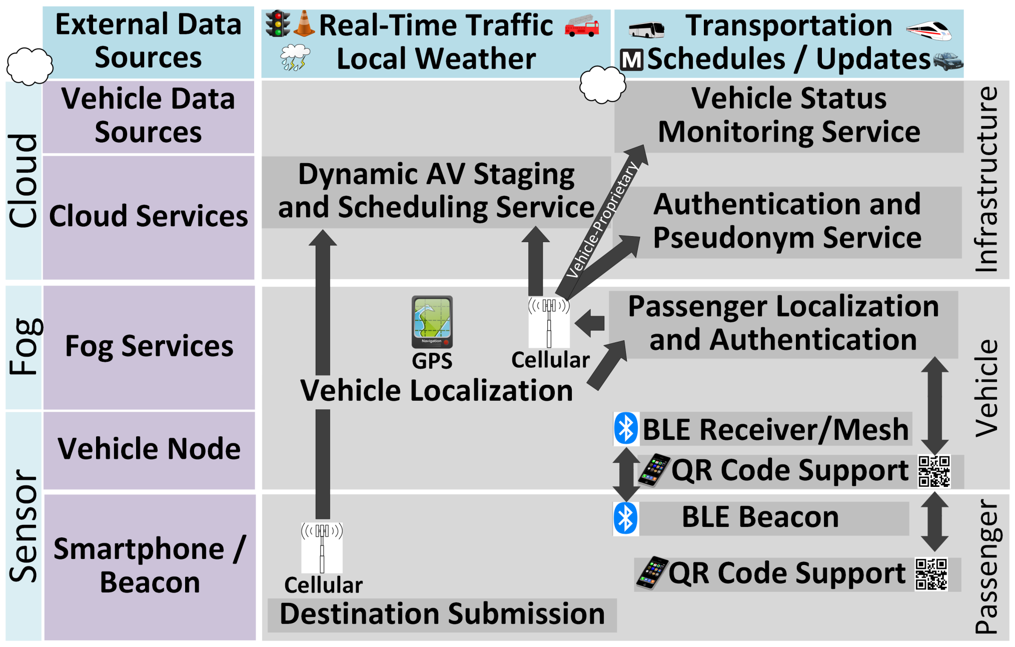

Figure 6.2: Sensor/Network/Service Relationships

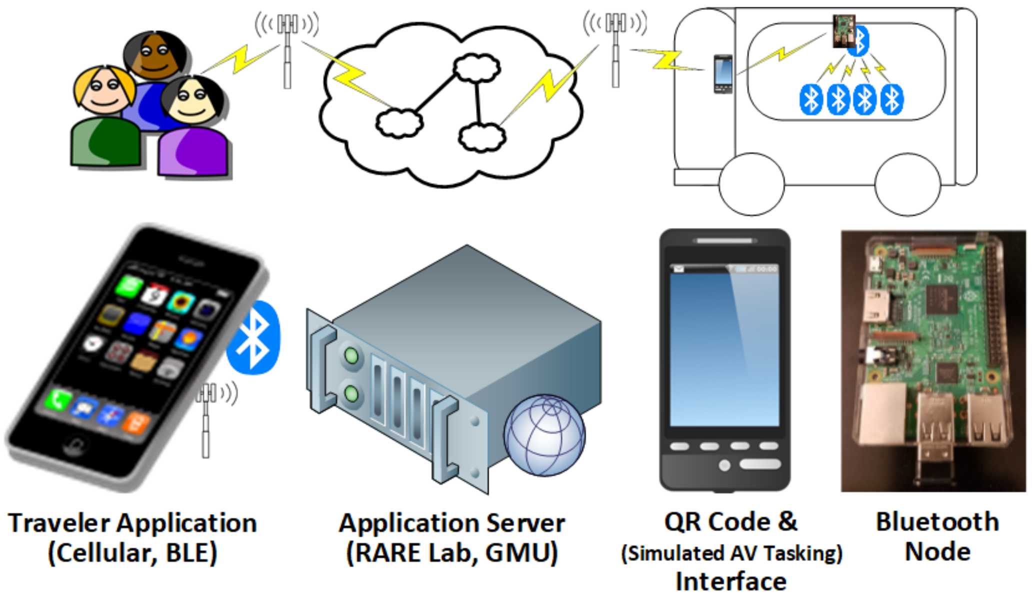

Figure 6.3: Physical Deployment Components

2-Factor Authenticated Disposable Passenger Identifiers

Any coordinated multi-modal solution requires knowing a traveler's source and destination locations and scheduling an AV while maintaining traveler privacy. This can be a challenge, particularly when that traveler moves between transport modes. One option currently available in mass transit uses multiple smartcard swipes per each vehicle within the travel scenario. Existing token solutions, such as MIFARE cards [119] use NFC technology to allow travelers to swipe a smart card at the entrance and exit of rail stations, buses and parking facilities to perform access control and payment. While the MIFARE system implementation currently popular in urban mass transit has communication security features to protect information transmitted between card and reader, there are three concerns (two security, one practical) with the use of these physical smartcards to facilitate passenger identification in my multi-modal AV scenario. First, the card's potential use as an authentication source has only one *factor*. In the current operational use case, a user's identity is established and confirmed simply by possessing the card. Should the card be stolen, an unauthorized party may use the card without issue at least until the legitimate owner becomes aware of the theft, and deactivates the card. Without replacing the card with something much more complicated, or inserting a password entry requirement at locations where the card is used (a non-starter in most scenarios) adding a second factor to this authentication scheme is impractical. Second, cards like these contain both the key material and authentication primitives as firmware within electronics inside the card. Part of the security features of smartcards prevents modification of this data without replacing the entire card. While implementations of smart cards like MIFARE use contemporary encryption, it may not be possible or convenient to either replace these algorithms or key material should they become compromised or upgrade them should they become out of date without issuing an entirely new card. Third is a practical concern: travelers must carry around the card, which has no other purpose than to authorize payment at travel mode changing points. Dedicated tokens like this one that require human interaction to use can delay transfers and travel simply by their use, and the requirement for additional travel items may encourage the potential for them to be lost while traveling.

Our practical concerns can be reduced if we combine the intent of these smartcards with devices that travelers most likely carry already. If one wants to migrate the key material and authentication primitives to software and place it onto traveler's smartphone application, one prevent the need for a traveler to carry an additional item. Consequently this key material and authentication primitives can be replaced or updated with a simple software update, *future-proofing* this solution and preventing the user or payment system from the cost and inconvenience of replacing physical cards. Lastly, contemporary smartphones are essentially mobile sensor platforms, complete with a fully fledged user interfaces and multiple radios for cellular, Bluetooth, NFC, and WiFi. This affords developers with options for adding additional authentication factors, unlocking with biometrics, or even wrapping their entire application around a commercially available multi-factor authentication mechanism, such as Duo Mobile [120].

In prior work [104], I constructed a prototype authentication client that uses a voice-based password to unlock functions within an Android application as part of a user bootstrap process to a smart computing space. According to NIST guidance [121], this is an example of a single factor authentication mechanism, however adding a second factor is relatively simple using a combination of the Google Authenticator [122] software and this voice-based password. Google Authenticator provides single use, out-of-band (from our application's point of view) password over the phone's carrier connection. Both the voice password and Google Authenticator token are used to authenticate the traveler to our AV tasking application server. As a final step in the authentication process, a pseudonym is created for the traveler, and stored within our service until the traveler's trip is complete, payment is processed, and the pseudonym is no longer needed.

Travel Destination and Special Needs Submission

Once authenticated, my solution accepts a final travel destination, pairs it with the traveler's pseudonym, and submits this pair to our solution's AV staging infrastructure through a

RESTful HTTPS connection. AVs tasked in this way pick up the *last mile* of multi-modal traveler. I consider earlier modes of travel out of scope, and outline a concrete motivating example illustrating this later in this paper. Additionally, this interface allows travelers to indicate special needs for the assigned AV, such as handicapped accessible vehicles or human-facilitated boarding/disembarking assistance. After submission, this information is used by our AV Staging and Tasking service, outlined in the next subsection.

6.2.2 Smart AV Tasking and On-Demand Scheduling

The goals of our *smart* tasking service is optimizing occupancy of vehicles while minimizing curb-side vehicle and passenger wait times. As these vehicles have an electric drivetrain, their duty cycles will be limited, requiring that this system consider the vehicle's battery life when deciding if it is to pick up passengers, or seek out it's charging station. Lastly, the outcome of this tasking solution must reduce the number of idle vehicles waiting for passengers, as urban pickup-location can themselves become traffic blocks if they are deployed curbside. Optimizing these factors balances usefulness of the AV to the traveler while reducing excess traffic caused by under-occupied vehicles.

On-Demand vs Fixed Scheduling

My solution is inherently *on-demand* focused, as travelers indicate their individual destinations and are supplied with a vehicle, when and where they are needed. This same solution can easily task vehicles that emulate a fixed schedule, simply by choosing a vehicle within a fixed period. My solution takes existing conditions and load demands and makes a choice that optimizes occupancy. If passenger demand outpaces vehicle availability, tasking that rotates vehicles through the route with short queuing intervals as quickly as possible emulates a fixed schedule route (similar to contemporary urban buses), with one significant difference: My solution enables fleet operators the capability of ensuring empty vehicles never traverse the route.

Price-Controlled Multi-Passenger Manifest Construction

Using pseudonyms, travelers are assigned to a vehicle based on common or close-by destinations, in a FIFO queue based on their expected arrival time. Destination-informed arrangements like this eliminate or minimize the need for multiple stops along a route, making the use case more desirable. I assume travel modes prior to the AV enable the system with real-time information that can be used to adjust assignments based on changes such as unexpected delays for certain travelers. Similarly, users can *cancel* their travel plan through the exposed interface, providing similar feedback to the tasking service.

Without constraints, average wait times for passengers assigned to a vehicle would be half of the difference between the expected arrival times of the first and last occupant. For some passengers and some scenarios, this may be an unacceptably long amount of time. One control that has been found to successfully motivate travelers to use one transportation option over another is occupancy-based pricing. In a study co-published by The Boston Consulting Group and World Economic Forum [13] simulated results show that pricing schemes that encouraged travelers to use multiple-occupancy on-demand vehicles has the most significant reduction in total travel time, even when compared to dedicated pick-up/drop off zones for AVs or dedicated AV travel lanes. We base our solution from this notion, by reducing the cost of using the AV when wait times are long, and increasing it when AV wait times are short per pre-specified AV pickup and drop off zone. To optimize vehicle occupancy, this model follow a similar scheme by increasing the cost for vehicle use when passengers request fewer co-passengers, and less when no such restriction is required. Lastly, destinations that are near-by to other destinations can be consolidated into a single stop by incentivizing passengers to choose a near-by destination using the same cost control.

One challenge with this kind of scheme is that the conditions of travel may change as more passengers are tasked to a vehicle, but travelers indicate their travel requirements at the beginning of their journey. If conditions change, such that additional pricing options are created for a traveler, the proposed system use travel application to supply travelers with new information along with the option of changing their AV tasking choices to take advantage of additional options. Additionally, passengers are supplied with the expected walking times for their *last mile*, enabling them the choice to simply cancel the AV reservation entirely, and walk to their final destination. This is particularly useful in the event a traveler's delay or other real-world condition has changed their reservation cost. I outline the structure of this complete system in Figure 6.2.

Travelers without smartphones or those unwilling to use the provided application would be incompatible with this arrangement. A comprehensive solution to this is out of scope of this paper, however some cases can be mitigated. For example, travelers with small children or those traveling as a group, can indicate the size of their party when they initially submit a travel destination. In this case, a minimum of a single passenger would require the application, and that single passenger's phone would provide all of the tracking information to our system.

Passenger/Vehicle Rendezvous

When indicating a travel destination, the application provides the expected pickup location for the *last mile* AV, and travelers head there when making their last travel mode change. Once a vehicle's manifest is finalized by our service, the appropriate vehicle is tasked and dispatched to the pickup location so that it arrives just before the expected time when then last passenger is available for boarding. This minimizes the amount of time the vehicle is occupying space at the pick up location. If the manifest construction is optimal, the wait time for the first arrived passenger should be minimized (or is otherwise within the traveler-specific agreed to wait time). A timeout is employed to dispatch a vehicle in the event the last passenger is delayed, so that the vehicle has a minimum idle time once dispatched. Once the last passenger boards, or the timeout is reached, the vehicle begins moving.

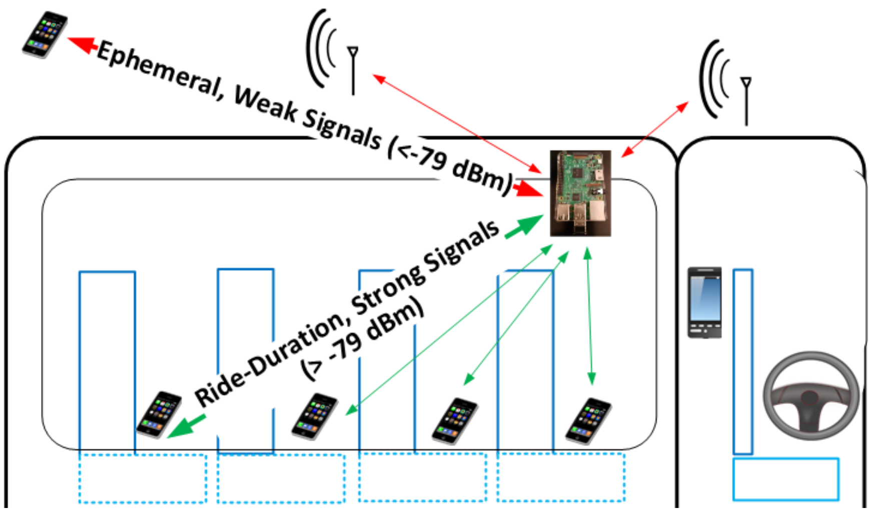
Ephemeral, Weak Signals (<-79 dBm)
Ride-Duration, Strong Signals
(> -79 dBm)

Figure 6.4: Passenger Ingress/Egress Tracking

6.2.3 Traffic-informed Vehicle Boarding and Embarkation with dynamic zones

One limitation to fully autonomous vehicle deployments is the safety and regulatory requirement of a human operator's presence on the vehicle while it's in use. It is unclear if this requirement will ever be relaxed, thus this model take advantage of their presence to provide security and on-boarding assurance so that only authorized passengers board the vehicle. Passengers who do not arrive at their reserved vehicle within a fixed timeout period would have their reservation invalidated. This would enable a vehicle to *time-out* and continue on its way without inconveniencing other travelers assigned to the vehicle. To enable automation of this boarding process, I outline a passenger ingress tracking system in the next subsection.

One preliminary simulations discovered, that fixed embarkation area near the L'Enfant Plaza Metro station became inaccessible to our AVs during periods of high vehicular traffic. This is shown as a labeled snapshot of one of the SUMO simulations in Figure 6.5. This drove the choice to enable our tasking system with the ability to dynamically assign pick-up and drop-off locations (shown as blue ovals) when traffic conditions make the original zones inefficient or inaccessible. Pricing modifications are applied (if appropriate) and passengers

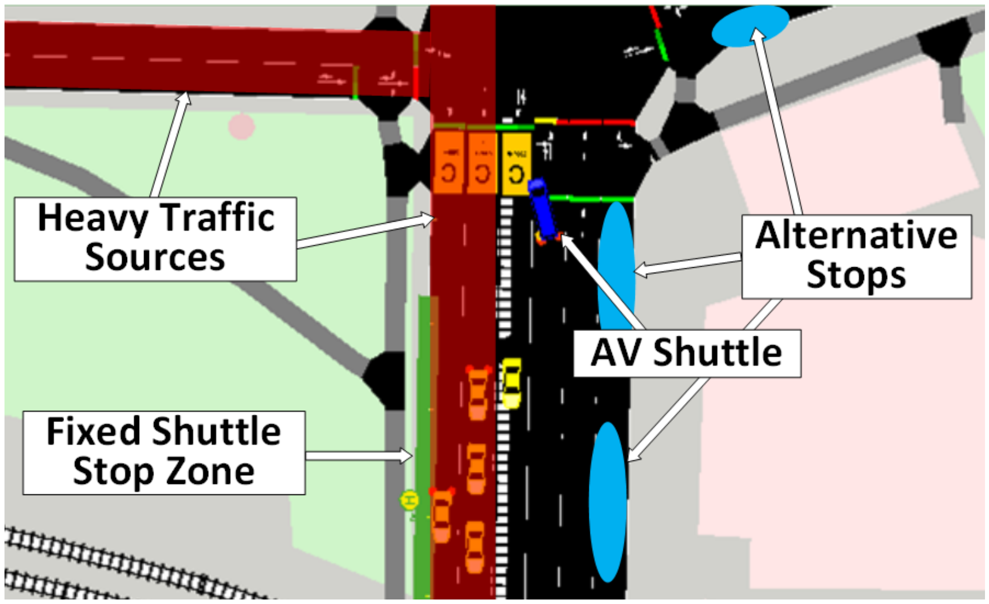

Figure 6.5: Labeled Simulation Snapshot

are notified through the traveler application where they can choose among the alternatives (or cancel the AV is desired). This solution is similar to a popular one among ride-hailing services used in dense urban environments, to suggest alternative pickup locations that reduce traffic impacts to wait times and travel duration [123].

Privacy Preserving Automated Passenger Boarding and Ingress Tracking

I improve on smartcard-based boarding by combining travelers' assigned pseudonyms with two passenger authentication solutions. First is a mutual scanning of single-use QR codes. Each vehicle is equipped with a small display and front-facing camera (the form-factor of a smartphone) just inside the boarding door (similar to modern buses) that displays a QR code that each passenger scans with the provided travel application. At the same time, the travel application displays its own QR code for the vehicle's camera to scan. These codes communicate information that is used to mutually authenticate both parties, based on key material exchanged when the travel application originally received the traveler's pseudonym. The codes themselves are capable of storing orders of magnitude more bits than is necessary for even the most advanced contemporary encryption used today [124]. Audible beeps are used to confirm/deny the outcomes of this authentication step. One drawback to this method is the potential bottleneck and burden created by performing the code scanning, similar to what is seen at modern airplane boarding scenarios. It is also the case that user presence on-board the vehicle cannot be verified after the scans, creating uncertainty that a passenger remains on-board.

An alternative proposal is similar to a solution implemented in past work to certify and track human operated vehicles as they enter a smart parking lot using Bluetooth beacon technology [117]. By placing a Bluetooth receiver on-board the AV, and enabling the phone to emit a BLE beacon in software, the system can determine when the passenger boards the vehicle both by the presence of the beacon and the signal strength of the beacon broadcast. This is shown in Figure 6.4. Weaker signals that exist for a short period of time, such as those from beacons from passengers on other vehicles or elsewhere within the ambient environment are considered non-passengers, while beacons that are observed continuously throughout the vehicle's trip and contain a strong signal are considered passengers on-board the AV. The threshold for the signal strength (RSSI, measured in dBm) would need to be experimentally determined for the type of AV, as the vehicle's exterior and internal obstructions play a role in this value. However, I expect the threshold value to be approximately -79 dBm, as it is common for BLE beacon broadcasts to measure to -59 dBm at 1 meter from source [125], and attenuation from signals traveling through a human body (and similar objects) have been shown to result in a 20 dBm loss [23]. In practice, I expect signal strength to vary, making continuous observation of the beacon a dominant metric passenger detection.

The Bluetooth solution is capable of localizing and tracking passengers. However authentication over this medium is complicated by the limited number of bits permitted in a single BLE beacon broadcast. One can combine these two solutions so that the beacon itself (or more specifically, the device's hardware address) can be attested as belonging to the traveler indicated in the QR code. This is particularly useful when the BLE beacon is transmitted from a device that enables the Bluetooth hardware address randomization feature of contemporary Bluetooth standards [126]. In this case, the hardware address of the phone periodically randomizes its hardware address so that observers may not link a particular person to their Bluetooth hardware address and use that knowledge in the future to track that same person using their future Bluetooth broadcasts. Since the hardware address is randomized, the time interval of this link being useful is minimized. When a passenger boards the AV, and performs the QR code exchange, the values within the code can indicate the phone's current BLE hardware address, or other identifying information found within the beacon broadcast. Then, this information is discarded by the phone after the passenger disembarks as it will be randomly generated in the future when needed by another AV trip. Additionally, this arrangement allows for acceptance of local broadcasts of traveler information by our traveler application (in the event of back-end network failure), such as the need for additional boarding time, wheelchair access, or other special needs outlined above.

6.2.4 AV Route Completion and Re-Tasking

As the proposed solution is employed at the last leg of a travel scenario, travelers are free from the burden of disembarking though the system. As the Bluetooth based tracking system can easily detect the absence of a previously seen beacon after the vehicle stops at the end of it's route, indicating that the passenger has disembarked. Once all passengers have disembarked, payment is processed on the back-end and the traveler's pseudonym is destroyed, then the vehicle is re-tasked with either another route or returns to its charging station.

6.3 Experiments and Simulations

6.3.1 Motivating Example

I created an example AV route by selected an area of Washington DC that was of interest to the city and other stakeholders, where mass transit is located nearby a significant destination for passengers. One such example is an AV pilot project recently proposed by the Washington DC government in collaborations with the Southwest Business Improvement District (SWBID), which specifies an AV route between L'Enfant Plaza and 10^{th} St SW as part of the city's *SW Ecodistrict Plan* [127] for that part of the city. This model deviate from this plan slightly and instead apply the route between the L'Enfant Plaza metro and The Wharf/Waterfront traversing 7^{th} street to the 7^{th} Street Park as shown in Figure 6.6. This route enables multi-modal transportation that leverages and incentivizes an existing metro station permitting faster ingress/egress of travelers to the waterfront without the potential added surface traffic of non-Metro transportation modes to L'Enfant Plaza (both ends of the route indicated with white outlines). While the proposed solution being used for any portion of a multi-modal travel scenario, for the purposes of this chapter, I confine the use of an AV for this *last mile* of travel, and complementary to the District's existing Eco-district plan and AV pilot route.

The *Be Healthy* autonomous vehicle principle [128] includes the empowerment of passengers to use *active* forms of transportation through AV use. My route selection drops off travelers at a location in the Wharf near to a Capital Bikeshare station [129], and several sidewalks that feed the Wharf with pedestrian traffic as shown in Figure 6.7. Additionally, this route is also near a WMATA bus stop, enabling additional travel options for AV passengers. The Washington DC Metro system is fully handicapped accessible, so any solution that serves Metro customers must also be handicapped accessible. For the purposes of simplifying my solution, I consider two types of passengers: those that are within a wheelchair, and those that are not. Each passenger in a wheelchair is assumed to use up 4 passenger

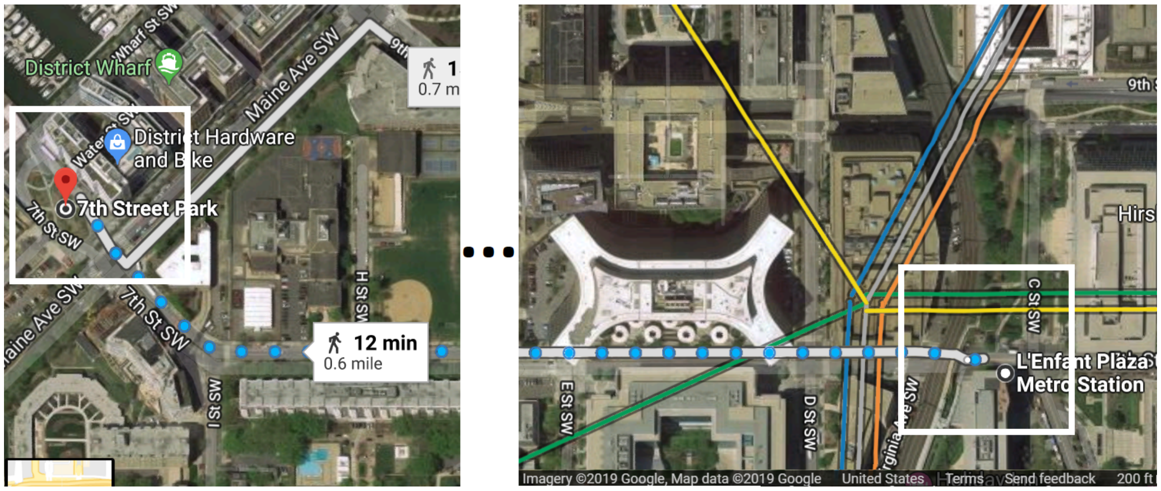
District Wharf
District Hardware and Bike
7th Street Park
Maine Ave SW
7th St SW
12 min
0.6 mile
L'Enfant Plaza Metro Station
Imagery ©2019 Google, Map data ©2019 Google United States Terms Send feedback 200 ft

Figure 6.6: L'Enfant Plaza Metro to Waterfront via 7th St.

spaces within the AV. Our route assumes AVs are electric (battery) powered and are capable of serving up to 12 passengers. This constrains a single AV to transport permutations of passengers combinations ranging from a maximum of 12 non-wheelchair passengers to a maximum of 3 wheelchair passengers.

6.3.2 Determining Vehicle and Traveler Wait Time Bounds

I developed a SUMO [118] based simulation to study the effect of AV fleet size, vehicle travel times in various traffic conditions, and traveler wait times between the L'Enfant Metro and 7^{th} Street Park. Travelers who exit the metro train with a goal of reaching the 7^{th} Street Park and Wharf business area are considered. I used the public metro ridership data at [130] to generate continuous pedestrian demand for the shuttle. During the AM peak hour (Figure. 6.10) an average of 1200 people exit the station every 15 min where as in the evening off peak hours it drops to about 50 people for 15 min. I assume that about 10% of the travelers exiting the L'Enfant metro have the 7^{th} Street Park as their destination to create the demand for AV. Factors that influence the traveler wait time bounds are modeled in my experiments are: AV shuttle fleet size, passenger demand for shuttle based on hour of the day, neighborhood traffic for the modeled hour. Table. 6.1 summarizes these factors

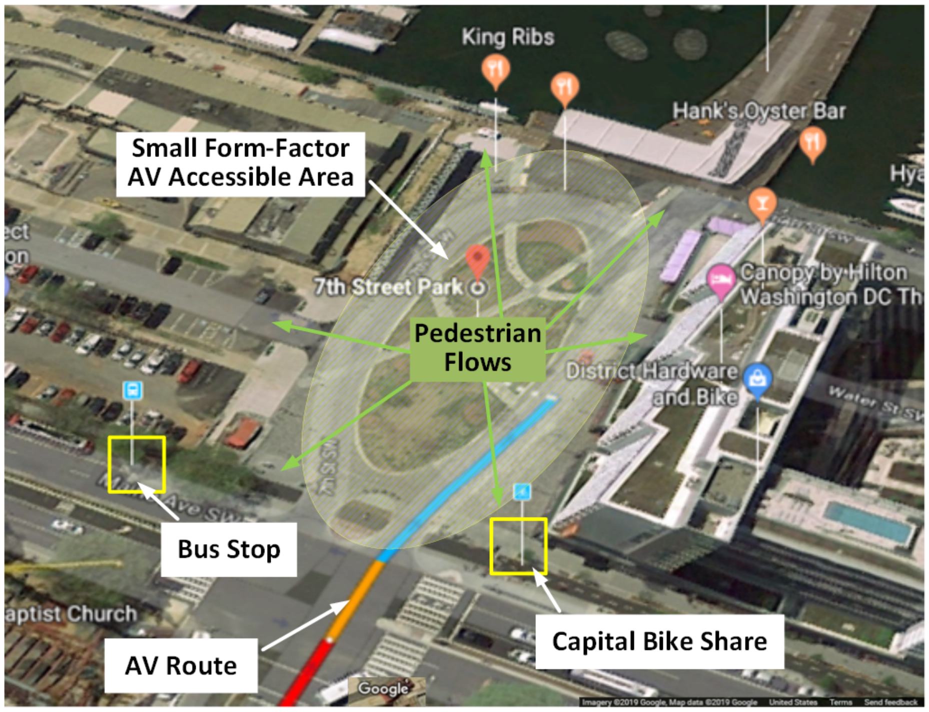

Figure 6.7: Park Embarkation Zone

and the corresponding results in terms of total number of un-serviced passengers and the average wait times of those that are serviced.

- Fleet Size vs No of passengers serviced: As Table. 6.1 shows, for metro peak hour exits increasing fleet sizes helps reduce the number of un-serviced passengers as well as lower the average wait times at the stop. During the off-peak hour all the passengers are serviced irrespective of fleet size and the average passenger wait time is close to 1 min approximately. This suggests that fleet-size could be minimized and AV can be dispatched on-demand for low passenger demand hours.

- AM Peak hour demand vs Evening off-peak hour: As both the passenger demand and surrounding traffic along the AV shuttle's route reduce during the off-peak hours compared to the AM peak hour, the average waiting time per passenger is reduced.

- Affect of other traffic on AV travel times: The traffic caused by other vehicles along the route of the shuttle affect the round trip times and therefore increase passenger wait times. In this simulation I considered the peak traffic scenario where no of vehicles per hour is as high as 1560 which is 12% of Annual Average Daily Traffic (AADT)=13006 for this route [131] along the route. In the off peak scenario, the hourly traffic volumes are much lower. For simulation purpose, I set this to 3% of AADT which is 390. The corresponding round trip times for these peak and off-peak these two traffic conditions are 1300, 801 seconds respectively. In peak case, I noticed the delays are primarily due to queuing at traffic lights. In the off-peak case the delay was due to multiple shuttles queuing at the 7^{th} street park drop-off area(Figure 6.11). For comparison purposes both scenarios were run with a fleet size of 10 and passenger demand of 480 per hour.

Table 6.1: Results for different demand scenarios

Fleet Size	Scenario	Total Passenger Demand	Unserviced Passengers (With, W/O) Traffic	Avg. wait time (sec) (With, W/O) Traffic
3	AM Peak	480/hr	(413, 163)	(722, 738)
5	AM Peak	480/hr	(364, 47)	(650, 325)
10	AM Peak	480/hr	(265, 38)	(517, 262)
3	PM Off-Peak	24/hr	(0, 0)	(88, 63)
5	PM Off-Peak	24/hr	(0, 0)	(40, 39)
10	PM Off-Peak	24/hr	(0, 0)	(41, 37)

6.4 Conclusion

In this Chapter I outline my preliminary solution to providing efficient, usable, on-demand tasking for Autonomous Vehicles in a *last mile*, multi-modal use case. We have assembled together technologies and techniques necessary to enable seamless interactions between traveler and the surrounding transit environment, to enable anonymous tracking that supports our tasking solution. With careful deployment of solutions like mine, cities can deploy AVs

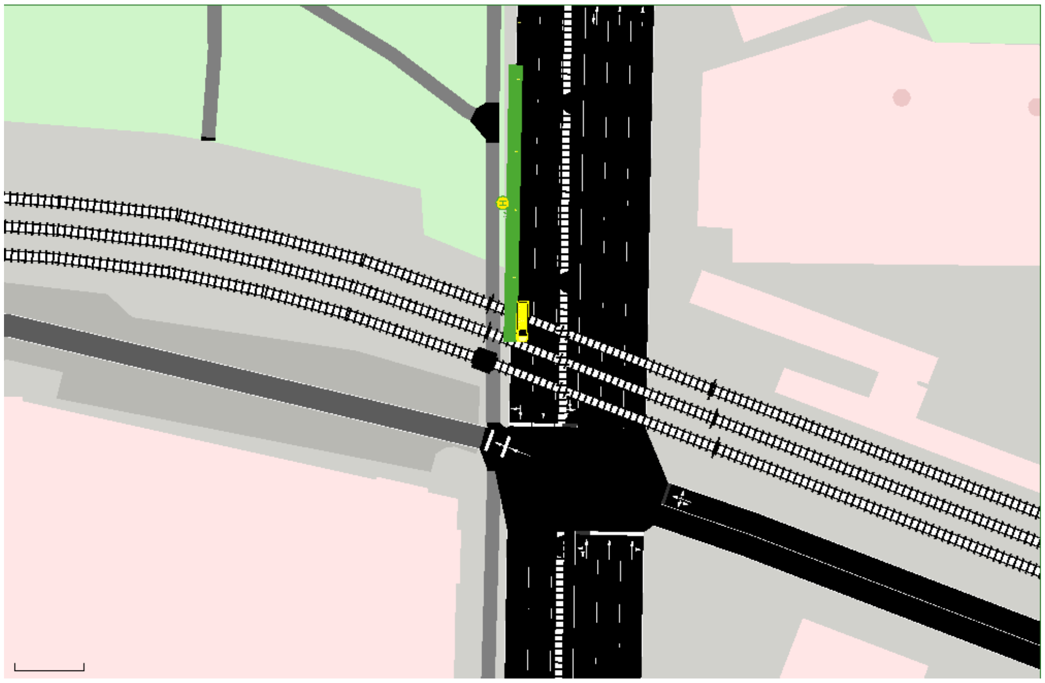
Figure 6.8: Single AV on route

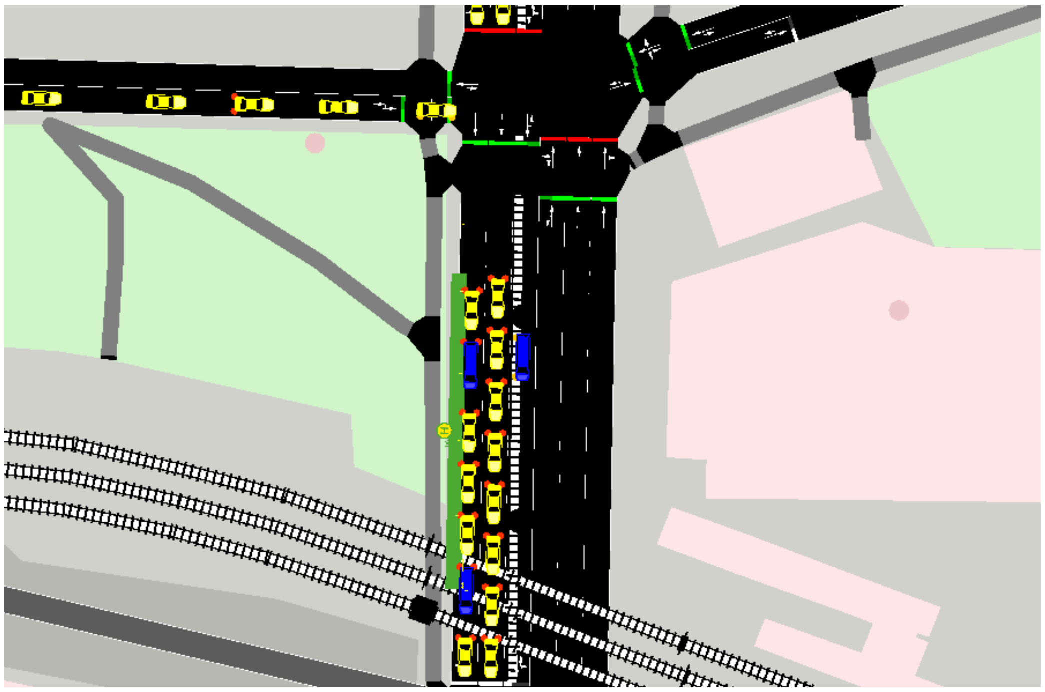
Figure 6.9: AVs with high neighboring traffic

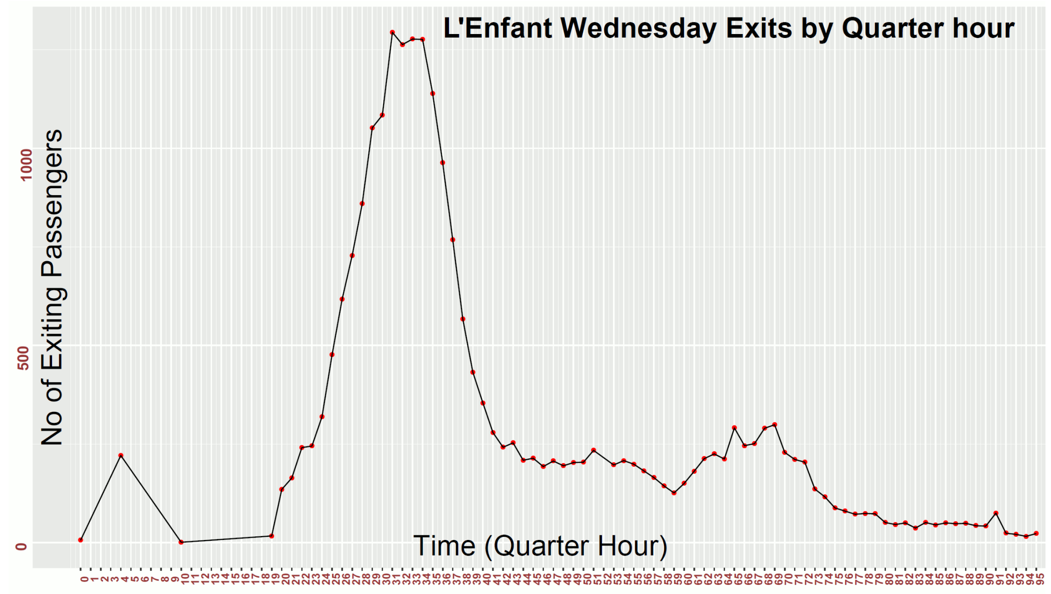

Figure 6.10: L'Enfant Metro passenger weekday exits

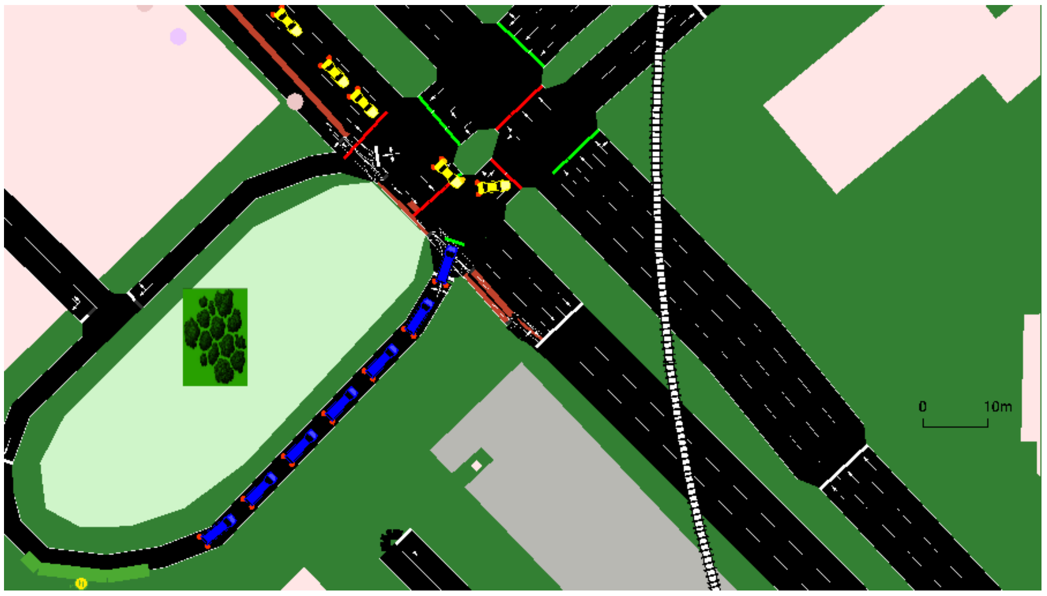

Figure 6.11: Queuing at 7th St. station

into strategic locations, enabling mass transit to become a more useful tool for the traveling public.

The work presented in this chapter was submitted to the 90th IEEE Vehicular Technology Conference (VTC'19-Fall), September 2225, 2019, Honolulu, Hawaii, USA. [132] (Author acceptance notifications tentatively due on June 17^{th} 2019)

Chapter 7: Coordinating AV Dispatch with Smart Remote Parking

Increased growth of public spaces often results in the increase in vehicular traffic. This is a particular concern for dense spaces like college campuses that facilitate a growing student population and for parking complexes dedicated to expanding organizations like commercial companies and military facilities. Mass transit solutions must evolve by incorporating new technology to leverage autonomy and reduced carbon footprints while simultaneously incentivizing travelers to increase their use.

In this chapter, I present an extension of our previous smart parking integrated with Autonomous Vehicle tasking work by integrating the two with a vehicle advanced warning system based on Bluetooth technology and a self forming mesh network. By detecting and authenticating vehicles bound for parking lots prior to their arrival, we can transact with an AV route planning and tasking system earlier than a vehicle's arrival at the parking facility, thereby reducing traveler wait times at the transfer point to an AV. Consequently, our system encourages parking in more remote lower traffic areas, thereby reducing localized problematic traffic congestion when coupled with maximal occupancy of automated shuttles. I deployed my Bluetooth sensor nodes to an off-site parking lot on George Mason University's Fairfax Campus, and conducted feasibility experiments to explore our proof-of-concept. I also outline a constraint-based shuttle assignment algorithm that minimizes traveler wait times while maintaining consistent AV service. By utilizing systems like these, densely utilized spaces such as college campuses, are enabled with new technology for combating traffic without significant cost.

I outline this chapter as follows: Section 7.1 contains additional motivations. A use case at the West Campus parking lot at GMU's Fairfax, VA Campus is described in section 7.2. I outline my solution, including the AV scheduling algorithm in section 7.3. I show experiments with the early-notification system, as well as campus stop planning in section 7.4.4. The chapter concludes in section 7.5.

7.1 Introduction

Personal vehicle traffic on and near large university campuses like the George Mason University's (GMU) Fairfax campus create a huge traffic congestion problems for students and the surrounding residential communities [133, 134]. When roads are packed with vehicles that commute to locations at the same time, commutes are delayed and emergency or essential service vehicles are slowed. The existence of large parking structures and open air lots, while necessary for operation of a university or facility, magnify this problem during common ingress and egress times (such as between 7.30- 8.30 am and 4.00-5.00 pm in office complexes). One way to reduce traffic is to reduce the vehicles on the road, and replaced them with multi-passenger mass transit solutions. Universities like George Mason University (GMU), and surrounding municipalities like the city of Fairfax can provide such solutions. However commuter's use of personal vehicles continues to influence the surrounding travel environment, implying that these solutions do not meet current traveler needs, or that those travelers are not properly incentivized to use them.

Recently, the GMU campus has created an off-site parking lot equipped with large form-factor campus-controlled fixed-schedule fixed-stop shuttles to encourage students to park off campus and ride the shuttle into campus. The campus has incentivized the use of this solution by lowering permit fees for this lot, while raising fees for lots and garages at the campus nucleus. This has made the off-campus lot popular among students, inferring that students are willing to park in less impacting locations as long as the university provides a means to cover the difference in locality to classroom buildings. Solutions of this type are steps in good directions, however the use of fixed-scheduling results in periods where vehicles are traveling their route with few or no passengers, the use of fixed-stop results in reduced passenger traveler need satisfaction. Additionally, large form-factor vehicles suffer from limitations to where they can travel on campus (e.g. roadways wide enough, instead of shared pedestrian routes for true door-to-door service) and occupy the roadways for a longer time due to their size.

Shuttle system designs like these are well understood an easy to manage, although innovations in multi-modal travel in support of *Dynamic Mobility On-Demand (DMOD)* that allow the use of technology to replace fixed schedules and fixed stop and replace fewer large shuttles with many small form-factor automated electric vehicles. In this chapter, I build on prior work by further incentivizing the use of off-campus connected parking solutions such as those deployed at GMU's remote parking lot. I design a system that connects my smart parking solution to an on-demand autonomous vehicle tasking system capable of dispatching small form-factor autonomous vehicles to dynamically created stops nearby student's vehicles within the lot and drop them off near their destination. To accomplish this task, I extend my smart parking mesh network to nearby road services and enable a vehicle tracking system informing this tasking system of incoming travelers, allowing for a reduction in passenger wait times by permitting the AV tasking to respond to the presence of travelers earlier. To enable this, I solve a lingering problem from past work by providing a fast means to authenticate the BLE beacons transmitted by vehicles to this localized mesh network. Additionally, by utilizing the localization services provided by our smart parking services, I am able to further incentives the use of this system by tasking AVs to pick up passengers at their parked vehicle. This further reduces passenger wait times, and reduces the need for dedicated shuttle shelter spaces and fixed shuttle stops.

7.2 Motivating Use Case

To help explore my solution, I use a route currently under consideration for AV use at the GMU campus, that links a large outdoor parking area near the west edge of campus (referred to as West Campus) to the center of campus where a majority of the academic and administrative buildings are located. I show an overview of this route in Figure 7.1. Here I see the so-called West Campus parking lot in blue, primary academic buildings in yellow, interior multi-level parking structures in red. County roads are shown as thick

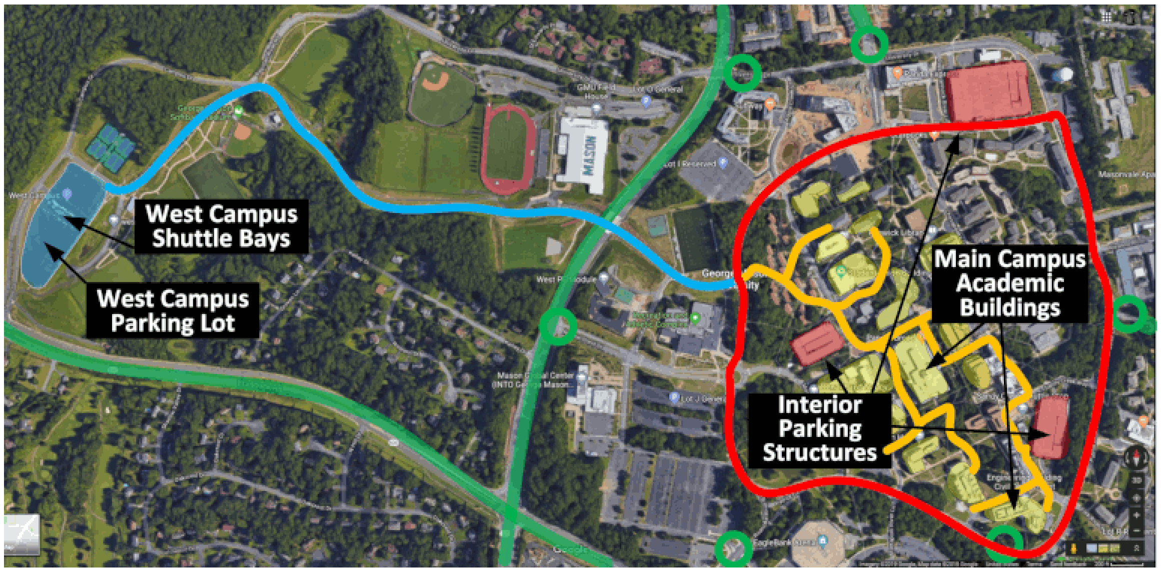

Figure 7.1: Campus Map with Exemplar AV and Pedestrian Routes

green lines, with primary campus ingress and egress points as green circles. The planned on-road AV route is shown as a light blue line, with purple lines specifying potential paths that a small form-factor AV could use to continue into the center of campus using existing pedestrian pathways (after light repaving). It is important to note, that the on-road AV route travels under Route 123 (center of the map) avoiding an intersection scenario at that location. This route then, becomes the most direct path from the West Campus lot to the center of campus. One of the existing traffic problem for the campus is the untenable traffic created during period of high classroom attendance, particularly for evening graduate school classes. During these times traffic often reaches a stand-still around Patriot Circle, preventing the flow of traffic into or out of campus, and greatly effecting the neighboring community (shown as a dark red loop in Figure 7.1, in part due to the placement of large parking structures in the interior of campus (shown as red areas).

7.3 Proposed Solution

My solution is an extension of my past work, organized into six distinct components, that together make up a collaborative smart parking and autonomous vehicle tasking system. I show an overview of my solution in Figure 7.2, with the main contributions in this paper shown in green. Here I see my system's components: An advanced warning system for detecting vehicles bound for our target parking lot; A smart parking occupancy detection system that uses in-vehicle Bluetooth beacons we implemented in a similar outdoor setting previously [117]; An on-demand multi-passenger dynamic-stop AV dispatch solution assisted by a mobile application I outlined earlier [132]; and A near-vehicle pickup service that uses vehicle localization outputs from the smart parking solution, to schedule and route AVs to locations corresponding to passengers' parked vehicles.

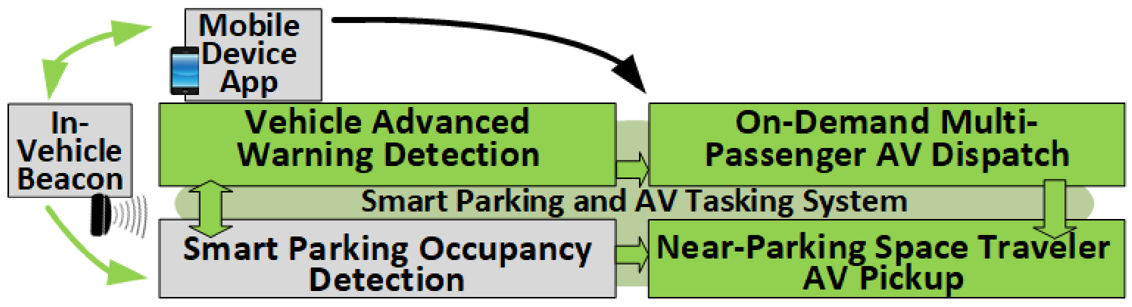
Mobile Device App
In-Vehicle Beacon
Vehicle Advanced Warning Detection
On-Demand Multi-Passenger AV Dispatch
Smart Parking and AV Tasking System
Smart Parking Occupancy Detection
Near-Parking Space Traveler AV Pickup

Figure 7.2: Solution Overview

Each of these sub-systems is collaborative, and shares information about vehicles and passengers over a combined campus-wide WiFi network, and a mesh-network (for locations without WiFi). My parking occupancy detection is based on Bluetooth Low Energy (BLE), and requires each vehicle to be equipped with my custom authenticated BLE beacon. This enables the vehicle detection system to discover vehicles near-by and *en-route* to the parking lot so that AVs can be tasked to pickup riders from them with reduced wait times. My

AV dispatch solution collects passenger destination information and couples it with real-time traffic and road conditions to consolidate as many travelers as possible to the fewest number of AVs traveling to a small geographic area with close destinations. This system also consumes data from the two other systems to create a third service that routes AVs to passenger's approximate parked vehicle locations to reduce passenger travel times and further incentivize their use.

7.3.1 Mobile Application and In-Vehicle Beacon Bootstrapping

A mobile application is used to collect passenger destination information prior to travel, and performs key exchange and pseudonym exchange used by the localization service. When a traveler enters their vehicle, this application connects to the in-vehicle beacon to deposit this key material and initialize the travel scenario. This application then becomes the primary interface for the traveler to use to communicate with our system.

7.3.2 On-Demand Multi-Passenger AV Dispatch

Previous work [132] outlined a vehicle dispatching system that consumes passenger-provided destination information and tasks an AV to rendezvous with the traveler at the *last mile* end of a multi-modal travel scenario. A simplified diagram of this system is shown in Figure 7.3, customized for the GMU campus deployment.

My solution incentivize students to park in the West Campus (ways from the main classroom area) lot and take an AV to class. Additionally, the West Campus Lot houses a 20-25 shuttle terminal connecting this lot to Fairfax County's mass transit network and other GMU campuses. By providing a safe and convenient AV solution that connects these different travel modes into a cohesive and seamless multi-modal solution, students are further incentivized to use mass transit and forgo the use of personal vehicle parking entirely. One drawback of multi-modal transportation is the wait times at each modal transfer. In the chosen scenario, students who park in the West Campus lot and wish to transfer to a shuttle must wait for the next shuttle to arrive based on a pre-defined schedule. This wait time can

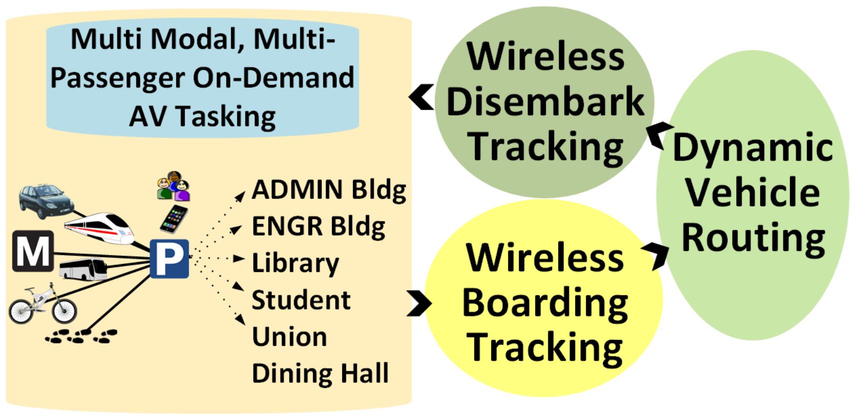
Multi Modal, Multi-Passenger On-Demand AV Tasking
M
P
ADMIN Bldg
ENGR Bldg
Library
Student Union
Dining Hall
Wireless Disembark Tracking
Wireless Boarding Tracking
Dynamic Vehicle Routing

Figure 7.3: Dispatch System Overview

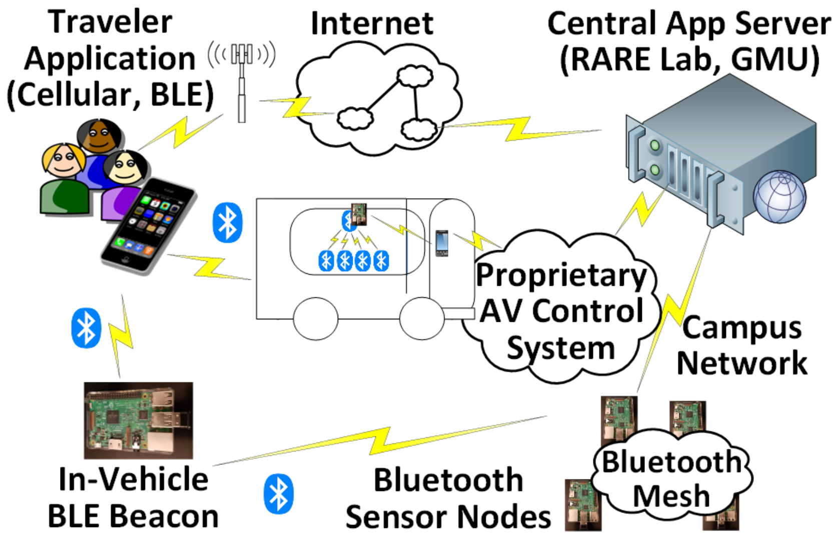
Traveler Application (Cellular, BLE)
Internet
Central App Server (RARE Lab, GMU)
Proprietary AV Control System
Campus Network
Bluetooth Mesh
In-Vehicle BLE Beacon
Bluetooth Sensor Nodes

Figure 7.4: Architecture Overview

be reduced if the shuttle scheduling switches to an on-demand system, the proposed system allows users to specify destinations when starting their travel, which our tasking system uses in AV planning. This can be more tightly coordinated when the traveler's exact location is known to the tasking system. However reliance on passenger provided GPS coordinates can create security and privacy vulnerabilities, such as a malicious user misrepresenting their location to cause unauthorized tasking of AVs. A partial mitigation for this is formed from the proposed use of local Bluetooth sensing and authenticated beacon broadcasts from traveler's vehicles, explained in sections 7.3.4 and 7.3.4 respectively.

AV Assignment and Tasking

I informally define the AV scheduling problem as a constraint satisfaction problem. In order to satisfy the constraints, I create a basic operating scheme and learn from the consequences of my choices to improve the algorithm in the future. Unlike contemporary fixed scheduling algorithms for buses and other surface mass transit, my solution takes into account the current vehicle occupancy, along with passenger provided details and passenger proximity to establish manifests that group passengers in such a way that the amount of required AVs and passenger wait times are both minimized. As these two characteristics are often at odds, I establish the following order of prioritized constraints, and design our scheduling and tasking algorithms around them:

1. Do not task an AV that is full, or low on power
2. Start an AV route at or before vehicle timeout is reached
3. Do not make a passenger wait past their timeout
4. Favor travel inside AV over walking, unless difference is excessive (e.g. 30 seconds)
5. Fill up an AV before tasking a new AV (soft constraint)
6. Assign an AV that matches passenger destination
7. Assign an AV that is closest to passenger destination

8. Notify passenger in the event constraints are violated

9. Once a passenger had boarded, do not change assignment

10. Attempt to improve assignment as new information becomes available

I show a basic manifest creation functions in Algorithm 6. The algorithm assigns (lines 6-19) a passenger to a vehicle based on passenger-indicated preferences (lines 7-9) and prioritized constraints. The call in line 10 to `get_best_AV()` (lines 20-28) returns (line 28) a single AV that best satisfies constraints (line 27) from all available AVs with empty seats (line 22). We attempt to find AVs that match the passenger-provided destination or are within a passenger-acceptable walking distance from that AV's stop (line 25), then sort this final list first according to the most full AVs to reduce the number of required AV's, then second by AV's with the smallest remaining timeout period so that passengers will have minimal wait times once on-board (line 27). The AV at the begging of this sorted list is the preferred AV for this passenger, and is returned (line 28) back to the calling function (line 10). If this call was successful, the passenger is assigned to this AV's manifest (line 12), otherwise a new AV is tasked for this passenger (lines 14-16). Our algorithm assigned the AV with the highest current battery life (line 14) so that AVs with lesser charge will have more time on their charging pad to charge their batteries. In the event no spare AV's exist, then the fleet is not large enough to accommodate this passenger, and they are notified of this situation. They can then choose to cancel their AV request, which removes them from the scheduling algorithm, or they can wait until an AV becomes available (line 19, and line 16 in Algorithm 7)

Algorithm 7 determines when a particular AV should begin its planned route to its destination in lines 4-17. There is a loop (line 5) that cycles through each *in-service* AV with a current passenger manifest (line 6) to determine which AVs are full or close to violating a maximum wait-time constraint of any of its passengers (lines 7-11). At either of these two conditions, the AV must depart for its destination (line 12). At this time, the AV is in-use an no longer available for additional tasking (line 13). This loop is used to attempt to assign a previously unassignable passenger, as real-world conditions for AV availability, passenger participation, etc may now be such that an AV is available for these passengers (lines 14-16). Once the loop is executed, a sleep time (line 17) is introduced such that real-world events can occur, requiring another run of this procedure. The second procedure (lines 18-31) in Algorithm 7 is used for the previously mentioned unassignable passengers, but also for further optimization of assigned passengers who have not yet boarded the vehicle. This further optimization attempts to correct cases where the scheduling algorithm placed a passenger onto an AV destined for a location other than their intended final destination by determining the end-to-end travel time of newly available AVs (lines 23-28) to determine if a better selection (line 29) is available, and performs an assignment swap (lines 30-31). We do not explicitly show the `add_traveler_to_manifest()` function, however, acknowledge that this function must confirm with the passenger over the mobile application that the new assignment is received, prior to removing them from their old assignment (line 31).

7.3.3 Smart Parking Occupancy Detection

I designed, implemented and tested a Bluetooth based smart parking system in previously [117]. I extended that solution as part of our AV tasking solution to enable tracking and localization of vehicles on the West Campus parking lot. Briefly, this system uses a Bluetooth-based sensor network to measure radio signal strength from vehicles equipped with our custom BLE beacon, relaying this information over a routed self-forming Bluetooth mesh network to a supervised learning classifier to predict parking space occupancy. I make one modification to this system to support our early-warning system, explained in the next subsection. I leave deployment details of the smart parking solution to the West Campus lot as future work.

QoS Improvements to Bluetooth Mesh Network

My Smart Parking solution deploys a mesh network to enable coverage for large outdoor lots. In these cases, external network connectivity is often limited, resulting in the configuration

Algorithm 6 AV Assignment / Manifest Creation

```
 1: global variables
 2:     m, active AV manifests
 3:     p_u, list of unassigned passengers
 4:     dst_table, pre-computed table of inter-destination distances
 5: end global variables
 6: procedure assign_AV(traveler info ti)
 7:     ti_at ← ti.predicted_lot_arrival_time
 8:     ti_to ← ti.max_wait_time
 9:     ti_dst ← ti.destination_building
10:     $v$ ← get_best_AV(ti_at, ti_to, ti_dst)
11:     if $v$ != null then
12:         m.add_traveler_to_manifest($v$, ti)
13:     else if there currently exists unscheduled AVs then
14:         Select unused AV, n, with highest current battery
15:         Task n with new route to ti_dst
16:         Add ti to new manifest for n in m
17:     else
18:         Inform ti no AV may be present upon arrival
19:         p_u.append(ti)
20: procedure get_best_AVs(ti_at, ti_to, ti_dst)
21:     pv_list ← {}
22:     for each AV v with empty seats in m do
23:         v_dst ← m.v.destination_building
24:         v_to ← m.v.remaining_wait_time
25:         if v_dst ¡= ti_dst and v_to ¡ ti_to then
26:             pv_list.append(v)
27:     sort_by_most_full_then_shortest_timeout(pv_list)
28:     return (pv_list[0])
```

Algorithm 7 AV Tasking and Dispatch

```
 1: global variables
 2:     m, active AV manifests
 3: end global variables
 4: procedure start_AV
 5:     while True do
 6:         for each AV v in m do
 7:             t_curr ← current time
 8:             p ← passenger with lowest wait time in v
 9:             t_p ← wait time of p
10:             ts_p ← timestamp p boarded v
11:             if v is full or t_curr - ts_p ¿= t_p  then
12:                 Instruct v to begin assigned route
13:                 remove v from m
14:             for each unboarded passenger u in m.v do
15:                 if v_dst != u_dst then
16:                     attempt_reshuffle(u)
17:         sleep()
18: procedure attempt_reshuffle(u)
19:     u_at ← u.predicted_lot_arrival_time
20:     u_to ← u.max_wait_time
21:     u_dst ← u.destination_building
22:     u_v ← u.vehicle_assigned
23:     v ← get_best_AV(u_at, u_to, u_dst)
24:     u_v_arr ← Expected destination arrival for u_v
25:     u_v_walk ← Expected walk time for u for u_v
26:     v_arr ← Expected destination arrival for v
27:     v_wlk ← Expected walk time for u for v
28:     if v != null and v != u_v and
29:     (u_v_arr + u_v_wlk) ¿ (v_arr + v_wlk) + 30s then
30:         m.add_traveler_to_manifest(v, u)
31:         m.remove_traveler_from_manifest(u_v, u)
```

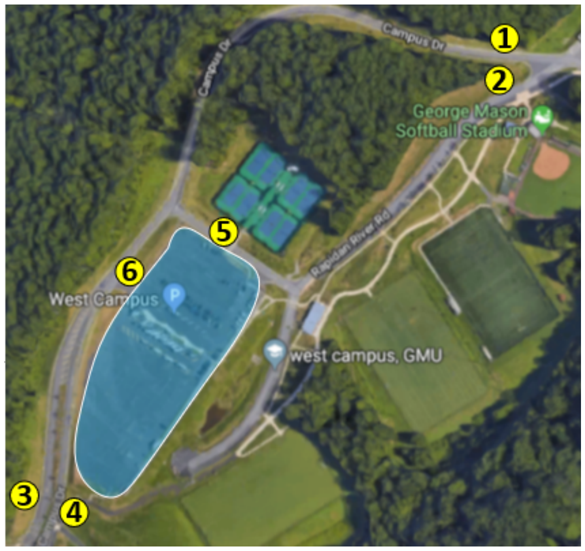

Figure 7.5: Node Placement

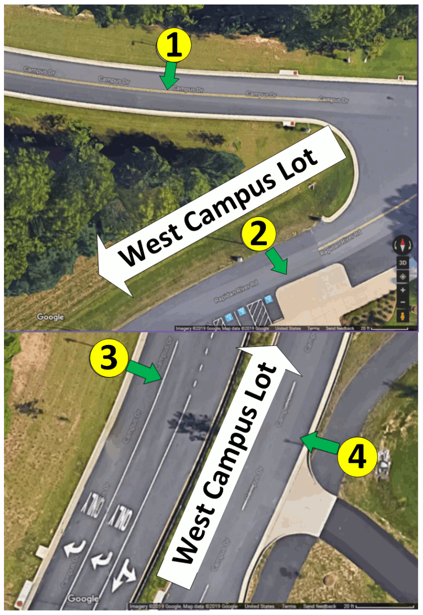

Figure 7.6: Close-up

of a central sink node within the network where external network connectivity can be made. For our West Campus use case, the only available campus network connection is located near nodes 1 and 2 shown in Figure 7.6), requiring that I extend out mesh network from the West Campus lot to this intersection. This increase message propagation delays that grows as the mesh network's maximum hop count grows. To mitigate this delay, I modified our mesh network design to add an additional message queue within each mesh node for *high priority* messages such as the ones used in the early warning system outlined in the next section.

7.3.4 Early Vehicle Ingress/Egress Detection

An extension of the use of local Bluetooth sensing and our mesh network for smart parking enables detecting the approach of known travelers well in advance of reaching their indicated parking lot. I use a simple detection method to enable predicting vehicles that are intent on parking in the West Campus lot based on the observation of their presence along roadways leading to the lot, and supplying this information to the AV dispatching service. Figure 7.6

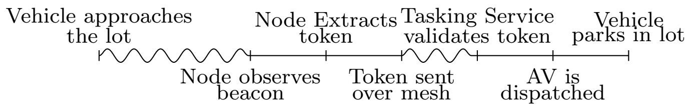

Figure 7.7: Stages of Vehicle Detection

show two images of road connections from the parking lot to surrounding roadways. First is the intersection between the county owned Braddock Road (below), and GMU owned Campus Drive, approximately 0.2 miles south of the first entrance to the lot. The second (above) is an intersection between Campus Drive and Rapidan River Road (also owned by GMU) that is 0.4 miles and 0.2 miles respectively from the nearest entrance to the lot. I placed six sensor nodes, similar to those used in my parking space occupancy detection system outlined in Section 7.3.3 near each of the numerically marked locations shown in Figures 7.5 and 7.6. When vehicles equipped with my authenticated beacon transmitter approach and pass by these sensor nodes, the signal strengths of their beacons are measured and used in conjunction with sensor information from the parking lot to predict the vehicle's is traveling direction. Vehicles that are determined to be *en-route* to the parking lot are submitted to the AV tasking system for including in vehicle manifest and route planning. I show the chronology of events in Figure 7.7. This early warning system informs the AV dispatching service of the arrival of a traveler's vehicle to reduce shuttle wait time. To remain useful, detecting the traveler must take less than the time required to park and walk to a shuttle shelter. I capture the difference in notification of passenger presence to an AV shuttle tasking system in Equation ((7.3.4)). Here, notice the benefit of a passenger early warning notification as the time the vehicle parks a vehicle t_{pk} minus the time when a vehicle's beacon is first observed minus the time it takes for that beacon's authenticated message to be transmitted over the mesh network t_{md} and validated by the system t_v. I use the QoS modification of our mesh network to minimize t_{md}, and have found that t_v occurs near-instantly on the hardware we use, however as message volume increases, so would each of these delays.

$$\Delta t_{warn} = (t_{pk} - t_n) - (t_{md} + t_v), \tag{7.1}$$

Where:

Δt_{warn}:	advanced notice time.
t_n:	time when vehicle beacon is first observed.
t_{pk}:	time until vehicle parks in lot.
t_{md}:	mesh network propagation delay
t_v:	time to validate authentication token.

Beacon Authentication

Each beacon must be authenticated to identify the traveler who sent it. I construct an authentication protocol using beacon broadcasts from both the lot, and the in-vehicle node. A 128-bit UUID field, and two 16-bit *major* and *minor* identifiers grant 160 total usable bits for each beacon, however contemporary encryption such as AES in CBC mode requires the transmission of a minimum of 256 bits (ciphertext block and initialization vector). To mitigate this I configure each lot node to broadcast an IV within its beacon's UUID field, as this value does not require secrecy to ensure the secrecy of the message it helps encrypt. I use the remaining pair of 16-bit major and minor fields to store a unique 32-bit identifier for each smart parking lot, previously exchanged with the in-vehicle beacon during bootstrapping (Section 7.3.1). I configure the in-vehicle node to continuously poll for and record BLE beacons with this pre-exchanged lot identifier, and then use the recovered IV from the UUID field to create an authentication token broadcast over the UUID field of its own beacon broadcast, along with the traveler's pseudonym spanned across the *major* and *minor* identifiers. During the bootstrapping phase, traveler pseudonyms are sent forward to the parking lot so its nodes can identify traveler beacons. To be resistant to replay attacks and adhere to proper AES implementation, an IV is only used to encrypt one beacon broadcast, and is replaced by lot nodes every 5 seconds. The lot node relays this information along with its current and previously broadcast IV back to the central node for authentication. Two IVs are sent to cover cases where the node's IV changes to a new value during this process. The central node generates the IVs for the entire network to use and submits them over the mesh network. Each IV is assigned a time window for use so all nodes broadcast the same value, and a bulk of IVs are transmitted over the mesh network such that at any give time, each node has the next hour's worth of IVs in its memory. This requires just over 92Kb of storage, which each of our nodes can trivially store.

7.3.5 Near Parking Space Dynamic AV Traveler Pickup

Once the smart parking system detects a parked vehicle on the lot for the first time, it notifies the tasking system to route the AV to this location so the passenger may board the shuttle near their vehicle. This is particularly valuable when the lot is large, and lot owners wish to forgo the deployment of shuttle shelters for waiting passengers. In my system, travelers shelter in their vehicle until the AV arrives. We leave optimization of this to future work.

7.4 Experiments and Discussion

7.4.1 Vehicle Tracking Experiments

My first set of experiments shows the efficacy of our advanced warning system. I deployed my sensor nodes to strategic locations near to the target parking lot shown in Figures 7.5 and 7.6. Here, nodes 1-4 are placed at the closest intersections for roads leading to the lot, and nodes 5-6 are placed at the two main entrances of the lot. My test vehicle was equipped with the in-vehicle beacon, and driven along several paths into or around the lot. Each of these runs started either at the GMU Field House (east of nodes 1-2) or on Braddock Road (south of nodes 3-4), and the vehicle either parked in the lot or drove past it. Beacon

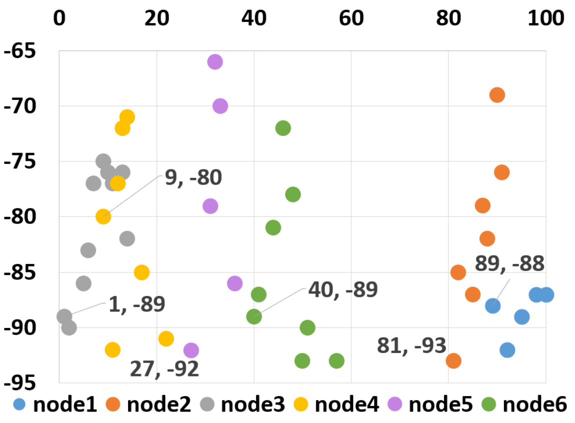

Figure 7.8: Bypass: Rapidan River Rd.

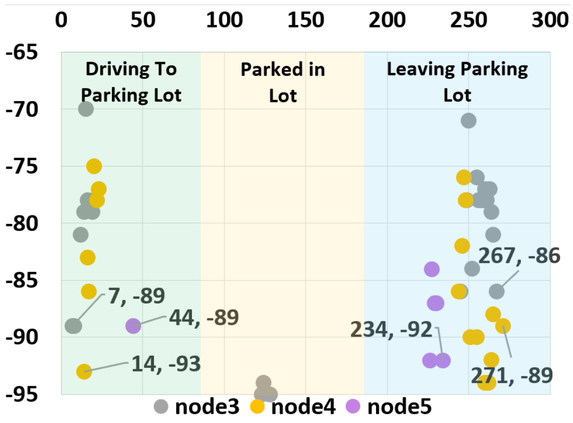

Figure 7.9: Parked in South Lot

observation times for the vehicle parked in the lot are given in Figure 7.9, with colored boxes indicating the true state of the vehicle. In this example, the vehicle is first detected approximately 70 seconds prior to parking (t_{pk} - tn, from Equation (1)). Notice that beacon observation times for the vehicle bypassing the lot in Figure 7.8, showing a delay between initial detection of the vehicle (by node 3), and confirmation that it in fact *through-traffic* (by node 1) of 80 seconds. I conclude that the early warning solution will only be practical if the traveler indicates their destination as the West Campus lot in advance, which we enable through the use of the proposed smart phone bootstrapping process in Section 7.3.1.

7.4.2 Early Warning Message Propagation Experiments

My second set of experiments address message propagation delay discussed in Equation 1. In my lab environment, I constructed a series of experiments with wireless nodes, aimed at studying the average increase in propagation delay when a new node hope was introduced. I noticed the results from these experiments in Figure 7.10, indicating an average delay increase per node hop of 2.93 seconds (t_{md} for each node hop, from equation 1). Independent experiments for Bluetooth transfer ranges (cite) indicate a distance between 10m and 100m, however I selected a conservative baseline of 30m as my baseline inter-node distance. The

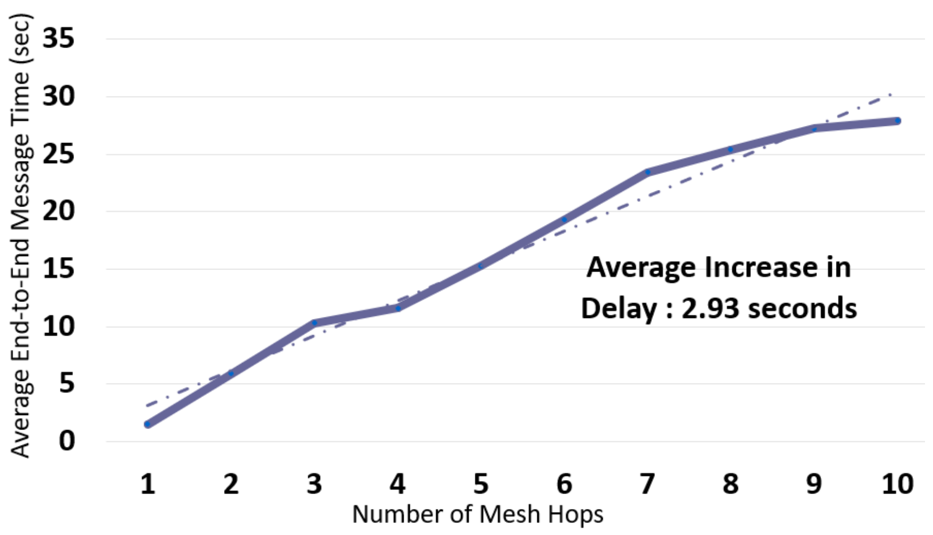

Figure 7.10: Average Message Propagation Time per Mesh Hop

speed limit on the GMU campus is 25 mph, covering this inter-node distance in 11.176 seconds without stop signs or other delays. This is an encouraging result, as I can safely extend our routed mesh network to support even earlier advanced warning deployments farther away from the parking lot by simply adding mesh nodes without concern that propagation delay will negatively influence the solution. I seek to repeat this experiment and our smart parking solution when deployed to the West Campus lot to obtain more realistic measurements of this delay under operation load and ambient interference.

7.4.3 Campus Class Bubble Map and Vehicle/Pedestrian Loads

To determine useful stops for the AV that I expect students to select, I scraped GMU's Spring 2019 Schedule of Classes to determine building occupancy for various parts of the day for each day of the week. I consider students arriving to campus when a class starts, and departing campus when a class ends. This represents the worst-case scenario for traffic, as each student would travel by car and be on campus roads at the same time as their classmates. I assume that each class listed has 30 students. I show the resulting profile for Mondays in Figure 7.11. Note that there are several spikes where many students are in transit, and select the median spike at 1630 hours to explore further (highlighted using the

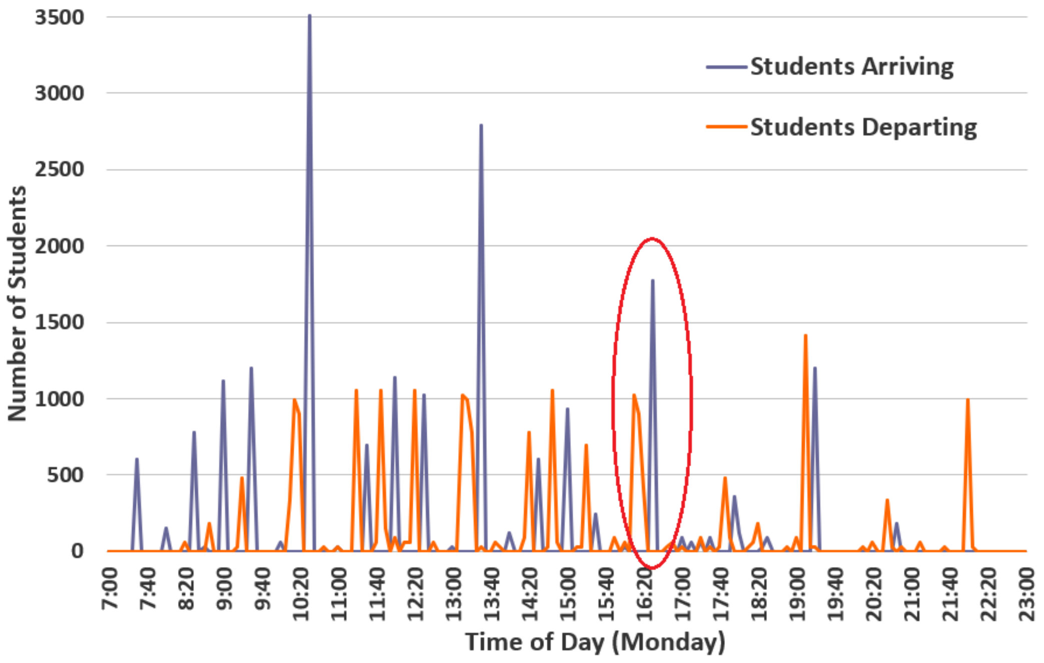

Figure 7.11: Summary Arrival/Departure Data, Mondays, Spring 2019

red oval). I mapped these locations to a bubble map in Figure 7.12 (shown as approximate sized yellow bubbles for buildings with more than 100 students). I chose three stop locations that provide coverage for most student load, shown as blue bus symbols. Additionally, I show all entrances/exits to campus as red or orange chevrons for primary (most practical) and secondary (locations less used), respectively.

7.4.4 AV vs On-Campus Vehicle Travel Time Comparisons

In the absence of detailed histories of where each student parks, and of on-campus traffic data, I use Google Maps to make approximations of how long our AV route will take, and how this compares to students driving to parking decks and walking to class. I use Google's approximation of vehicle travel times for our AV as it travels on campus streets, and assume our AV will slow to 10 mph while traversing pedestrian walkways, coupled with it's online traffic data for more realistic estimates. In this discussion, I set Google's traffic estimates to 1630 on Mondays for consistency with the rest of this chapter. Comparisons are made

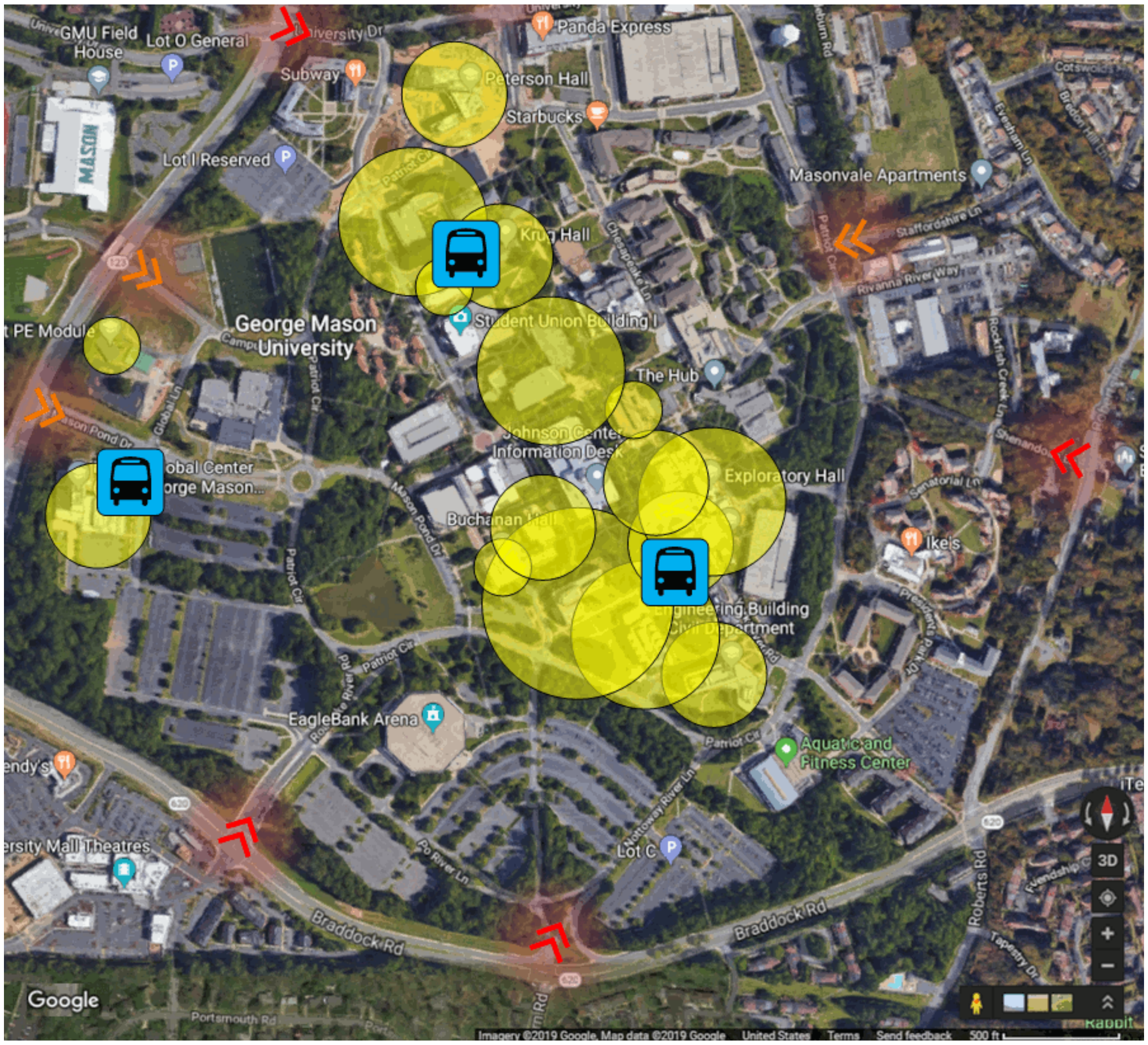

Figure 7.12: Class Occupancy Bubble Map, Mondays @1630

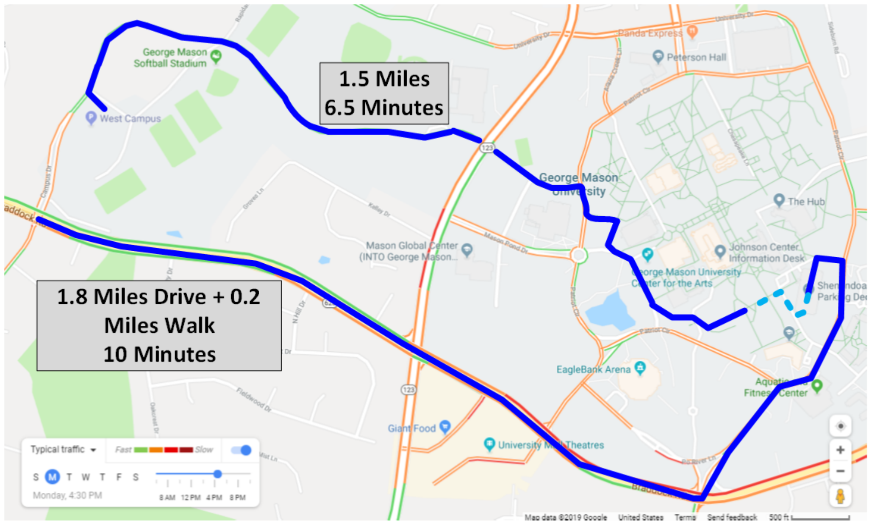

Figure 7.13: AV Operational Route vs. On-Campus Driving Route

between the duration of time a traveler starts at one of these entrances and either drives to the West Campus lot and takes and AV to the south east stop (in Figure 7.12) or drives to the Shenandoah Parking Deck (the closest garage to that stop) and walks to the south east stop. I illustrate one instance of this comparison in Figure 7.13. Notice that the AV (top map), delivering a traveler to the stop in 6.5 minutes, compared to the vehicle parking in the parking deck (bottom map) in 10 minutes. For simplicity, I consider the time a traveler takes to locate a space in the West Campus lot and within the Shenandoah Parking Deck to be comparable. When I factor the drive to the West Campus from the start of the shown vehicle route (approximately 1 minute), I see that the AV solution saves 2.5 minutes from the total commute time for the traveler. Similarly, traveling to the West Campus lot from the north and south east entrances to campus adds an additional 3 minutes of commuting time, while the same route directly to the parking Deck saves 2 and 3 minutes respectively, making the AV route take 2.5 minutes and 3 minutes longer (respectively). While the travel times do not favor the AV solution, they remain comparable, providing confidence that students can choose this solution without significantly impacting their commute, but gain the benefits of reduced permit costs for off-campus travel, and a reduction in on-campus traffic.

7.5 Conclusion

I extended my prior work that enables smart parking systems to be coupled with AV tasking to reduce traffic in densely packed localities like college campuses. I use Bluetooth technology to enable our AV tasking system to react earlier to the presence of travelers, so that routing and occupancy can be maximized while remaining customized to meet traveler needs. Using systems like ours, AV usage can be better integrated into *smart* environments to provide more efficient and effective operations.

The work presented in this chapter was published in *Coordinating AV Dispatch with Smart Remote Parking,* Paul Seymer, Chaitanya Yavvari, Duminda Wijesekera, Cing-Dao Kan. To appear in proceedings of the 2nd IEEE Connected and Automated Vehicles Symposium (IEEE CAVS'19), September 2223, 2019, Honolulu, Hawaii, USA. [135]

Chapter 8: Ongoing Work: Deployment to West Campus

This chapter contains preliminary results from ongoing experiments being conducted at the West Campus lot at GMU's Fairfax Campus. This deployment is to a parking lot much larger than past work and presents additional challenges to localization and mesh networking, outlined in the next subsection.

8.1 New Challenges

In contrast to prior localization work at the CCSA lot deployment of my solution to the West Campus lot incurs several new challenges. First, the lot itself is much larger, and it composed of multiple space locations in areas without continuous lines of sight. This may make the deployment of multiple overlapping sensors cost prohibitive. Similarly, a mesh network created that spans such distance will have many more hops than previously encountered, creating additional message delay. Second, the lot will be in use while experiments are underway, which will create a scenario where space fingerprints are created in the presence of temporary obstruction created by other vehicles. While I attempted to fingerprint around these vehicles, some obstruction was unavoidable. Additionally this in-use notion combined with the size of the lot made the time required to fingerprint each space for the entire lot prohibitive, so my solution will need to function well even in the absence of a complete radio map. Lastly, while I could fingerprint the entire lot at once, I chose to fingerprint in sections and combine the fingerprinting data together into a single predictive model. I explore this decision and its impact further in section 8.3.

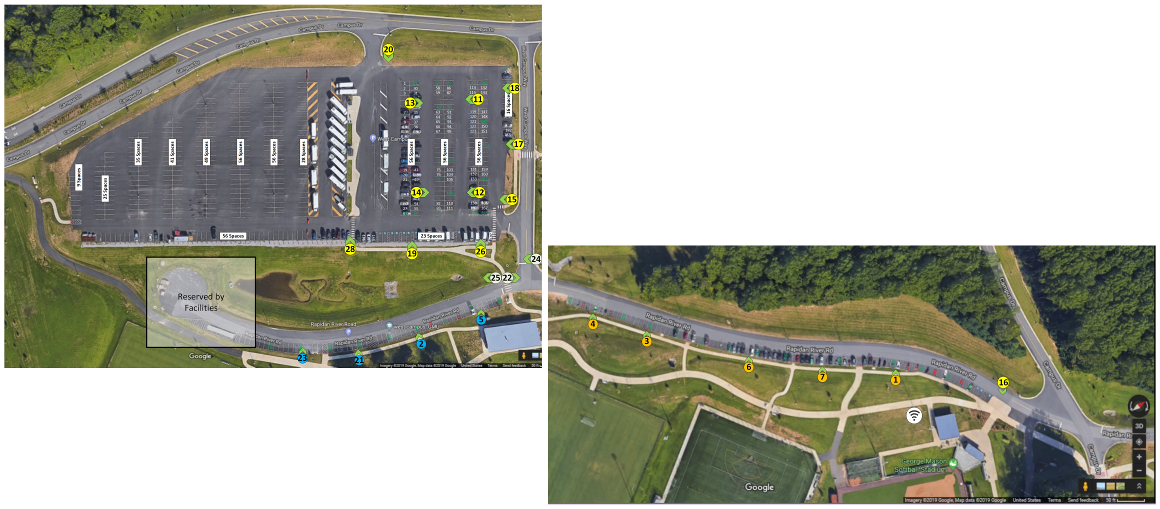

Figure 8.1: Lot Node Placement, Primary Campus WiFi Uplink

8.2 Deployment

Node deployment to West Campus is shown in figure 8.2. The main bulk of West Campus parking is contained within a large contiguous parking lot (left, upper), flanked by a long strip of parking spaces that link the intersection of Campus Drive and Rapidan River Road (right) to a campus facilities depot (left, lower). Each of these sections have differing elevations, creating line-of-sight issues for our mesh network. During the period of my experiments, the "south" portion of the main lot was blocked off and was not included in the study.

The resulting node deployment was 11 nodes, 4 nodes, and 6 nodes for the main lot, south and north portions of the parking strip, respectively. Additionally, three nodes were placed at the center of each of these areas to provide mesh connectivity and additional localization data for a grand total of 24 nodes for West Campus.

8.3 Experiments, Preliminary Results, and Observations

I conducted three separate fingerprinting activities to maximize the amount of available instrumentation nodes within the main lot. Spaces were selected based on current occupancy at the time of fingerprinting, with an attempt to be as distributed and comprehensive as possible. This resulted in 24 spaces fingerprinted in the main lot, 9 spaces fingerprinted in the south portion of the parking strip, and 16 spaces in the north portion of the parking strip. This lowered fingerprinting sample resulted in between approximately a 1:10 and 1:15 fingerprinted space to total space ratio.

While my analysis is ongoing, I was able to observe that modeling each of the three fingerprinted areas produced a prediction capability with 100% accuracy. This is mainly attributable to the physical distance between each fingerprinted space, as no selected fingerprinting space was located next to one another and the large number of nodes within the main lot was sufficient to produce discriminating beacon measurements.

To provide testing data, I used the same vehicle to test both localization and our early warning system implementation to test 14 passes of a vehicle driving into the lot and parking in various locations. To deploy the early warning capability, the entire mesh network was deployed creating a slightly different degree of node observations. In the case of the testing deployment all 24 nodes contributed their beacon observations, while in the fingerprinting stages only the nodes pertaining to each of the three lot areas were active. This created a challenge when using the prediction model created with the training data, as it effectively operated without data points from nodes observing a beacon from another parking lot section. Additionally, since the test experiment used spaces that were near-by but not identical to the spaces used for training, we needed to consider each prediction as being "correct" if the resulting space prediction was near-by. This was largely a qualitative measurement, however predictions were considered near-by if they resulted in a space within 10-15 spaces or similar distance if no spaces existed. I also applied the "zoned" approach used in previous experiments, creating three zones to match each of the parking lot areas (main, south strip, north strip).

Taking the original fingerprinting model and applying to to the 14 test spaces resulted in correct lot area prediction and near-by space prediction of 26.78% and 33.92% respectively. Intuitively, this low result is due to problems created by the mismatch between fingerprinting node counts and testing node counts. To explore this impact, I removed all test sensor data that was not produced by nodes originally used in the fingerprinting data, and repeated the prediction executions. This change resulted in each test space predicting to the correct zone 100% of the time, and each space matching a near-by fingerprinted space 89.29% pf the time. While this result is not strongly validated, it will guide future work I will perform.

Chapter 9: In-Flight Aircraft Smart Space Security using Multi-Entity Trust Evaluations

In this chapter I propose a hybrid of distributed and centralized multi-trust models specific to aircraft and in-flight use cases that evolves while the aircraft is in flight to assist in mitigating cyber risk of *in-flight smart spaces*. This system is fed by passenger records and service interactions, and human metrics from government and other sources that are each time and monetarily expensive for an adversary to replicate. Additionally, outputs from out passenger tracking and vehicle tracking systems can be integrated into this system to enable it with additional indicators. These metrics are used to augment a traditional peer recommendation-based trust management system. Data used in trust calculations will remain on the sensitive networks that are authorized to contain them or in a trusted cloud so that collection tradecraft is not exposed, allowing only non-sensitive trust calculations to be used elsewhere in the model. This allows all entities to leverage this data without exposing it to unauthorized parties.

Smart spaces and facilitate human-to-human and human-to-machines interactions. Similarly *smart cities* inter-connect more infrastructure to provide enhanced services and human accessibility of them. In the Air-travel domain, in-flight networking and *NextGen* are two emerging analogues of Smart Spaces and Smart cities equivalents in the sky. These changes are aimed at evolving airports, transportation infrastructure, and passenger services to provide more efficient and secure flight operations with better traveler experiences. It is only a matter of time before airports and aircraft themselves advance and become *smart spaces* by provide seamless connectivity to passengers to remain competitive. This will result in an increase of diverse passenger-provided mobile computing platforms connecting to in-flight networks. As the number of IP enabled National Airspace Systems (NAS) is expected to exceed 50% by 2020, the risks posed by cyber attack increase for all connected systems.

Traditional cyber incident response often begins post-compromise, sometimes days after an attack occurs. If this conveys to aircraft networks, such a delay may result in life threatening consequences. If an attacker were to compromise the aircraft through in-flight Smart spaces, human life and property could be lost before ground-based defenders would have an opportunity to act. Entities responsible for aircraft security such as the TSA must be enabled to respond to new types of threats by deploying preventive risk-based computer network defense (CND) to these emerging environments. Many challenges exist for this potential attack space/surface. First the lack of past experience with cyber attacks on these specific networks makes network-based detection difficult. Second, because multiple organizations, both inside and outside the government are involved in air travel and network communications, a central authority authorized to collect and interpret network artifacts and aggregate them with other intelligence to make threat determinations does not exist. Third, because traditional network defenses are often expensive, heavy, and require a great deal of human care, duplicating these capabilities across the multiple independent entities involved may be cost or physically prohibitive. Government and regulatory organizations like DHS and the FAA already have access to private records and other information that potentially can speak to the trust and integrity levels of individuals in travel scenarios. This information is currently in use for air travel security through programs such as TSA Pre✓and Secure Flight. One intent of NextGen is to implement new ways to use similar information to improve public safety and the efficiency of air travel. Commercial entities who provide in-flight networks have access and permissions to collect network observables. Humans in an aircraft's flight crew and fellow passengers who interact with others can each provide valuable observations of activities on-board aircraft. While each of these individual data sources lacks enough contextual information to detect cyber attacks, we suggest combining them into an air travel entity trust model to evaluate the risk of particular aircraft, passengers, airports, etc. can supply air travel security services with preliminary information needed to observe and respond to suspicious activity, and enable ISPs and technology vendors monitoring networks to more efficiently target defensive resources to the in-flight networks that most need them.

If an untrusted passenger loses access to aircraft WiFi networks, or is not allowed to connect to on-board servers or the Internet due to untrustworthy activity, they may be denied the connectivity required to conduct their attack. Similarly, if expensive tools and human defenders can be engaged only when needed and on a per-passenger or per-aircraft basis, such efficiency gains may overcome prohibitive computation, bandwidth, and personnel costs enabling detection of network threats that appear while an aircraft is in flight.

9.1 Introduction

Aircraft operators have been enhancing their on-board commercial offerings to include in-flight WiFi, and in-flight entertainment services. Implementers of systems like these continue to add services and increase connectivity of ground and air systems, including cabin management systems for flight crew, document management services for cockpit operations, weather reporting, and access to ground-based airline networks. It is only a matter of time before airports and aircraft themselves advance to include more technology and become *smart spaces* to remain competitive.

As shown in Figure 9.1, in-cabin computing services are delivered to passengers through touchscreen displays on the backs of seats commonly called Seat Display Units (SDU), connected to a Seat Electronic Unit (SEU) under the seat. These units are connected to Floor Disconnect Boxes (FDX), which allow network connectivity to a centralized on-board In-Flight Entertainment Server (IFE). Current implementations of this infrastructure, such as *OnBoard* [136] connect to a commodity Microsoft Windows server using common industry protocols such as 802.11, wired Ethernet, or industry specific protocols. At the perimeter to these networks, sits an Iridium antenna to link theses services to an Internet Service Provider (ISP) via satellite. While initially focused on paid passenger services (movies, Internet access, etc.), implementers of these systems continue to add services and increase connectivity of ground and air systems, including cabin management systems for flight crew, document management services for cockpit operations, weather reporting, and limited access to ground-based airline networks. As space is a premium within the aircraft, passengers may

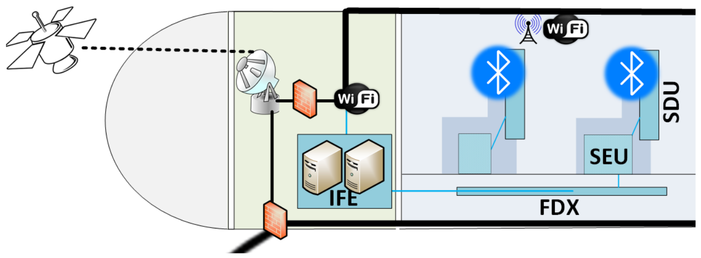

Figure 9.1: On-board Network Components

choose to carry smaller devices into the passenger cabin, should they be allowed to co-opt larger user interfaces such as keyboards, mice, and seat-back video displays found within the in-flight entertainment systems. Strict airport physical security often makes handling of larger electronic items (particularly laptops) problematic and requires special inspection procedures, further encouraging passengers to leave their laptops in checked luggage (or at home). My prior work proposed a middleware solution that enforces fair automated wireless keyboard, video and mouse (KVM) co-opting within a public smart space [104] over Bluetooth and WiFi. I envision a similar solution deployed within each seat to create a single-passenger smart space.This will result in a multitude of diverse passenger-provided mobile devices connecting to in-flight networks. Aircraft systems themselves too will become more connected. In the US alone, advancements due to programs such as NextGen [137] will increase the number of Internet Protocol (IP) enabled National Airspace Systems (NAS) from 36% to between 50% and 60% by 2020. As aircraft are outfitted with interconnected networks the risks posed by cyber-attacks increase for all connected systems.

In the past, on-board avionics systems have benefited from strong physical isolation from passengers, as wired connections were not exposed to the passenger cabin and radio communications required significant investment. Security researchers have suggested that compromise of an IFE networks can enable an attacker to compromise the aircraft's engines and other critical systems [138]. In the US, the Federal Aviation Authority (FAA) has dismissed [139] this suggestion, claiming the systems are sufficiently isolated. This, however, could easily change in the future as the desire for cost savings and network integration increases and adversaries become more innovative. For example, should SEUs and SDUs be connected to shared communication resources within the aircraft's aviation systems, adversaries may have additional hardwired means to access these networks from within the passenger cabin. If passenger devices are allowed to connect to these integrated systems, then these devices become a new attack vector. Both Civil Aviation Authorities (CAA) and their corresponding Aviation Security Authorities (ASA), must respond to new types of threats by deploying computer network defenses to these newly interconnected environments.

There are several potential ways to address these new threats, each of which is problematic for in-flight networks. First, aircraft designers could build network security systems into the aircraft itself. Such defenses must constantly counter evolving threats in a timely fashion. Accreditation and certification processes for aviation are often time consuming and may be cost prohibitive for older aircraft. Additionally, engineering cyber attack detection on modern computer networks often requires combined deployments of research-backed network pattern matching techniques to detect known threats, and storage and processing intensive human-guided adversary hunting activities to uncover new threats. While deploying these efforts to a ground-based network such as the airline's central ISP may be feasible, performing the same tasks for the in-flight network is problematic as hosts on that network produce traffic that is not always visible to perimeter defenses or may require probing devices on that network by human driven tools. A network design that taps all in-flight networks and routes copies of traffic to ground defenses may solve this visibility problem, however, throughput produced may exceed a single satellite Internet connection, making it impracticable for all aircraft. It is also the case that network-focused defense is often augmented by host-based defense to gain context into network attack traffic that would otherwise look benign. Airlines could require that all devices that connect to it run some form of host-based defense on the device, however making such a demand may prevent or discourage users from connecting to in-flight systems, or will create a user support demand on the airline. It is unclear if cyber threats for future in-flight networks will be similar to those currently understood by the network defense community, and the expected high diversity of end-user devices (among other challenges) may further confound the ability for defenders to understand and detect these new attacks.

Each of these solutions enables a post-compromise response. For the air travel use case, a delay in network incident response may be too late to avoid life threatening consequences. It follows, then, that prevention-focused defense is required that enforced controls that prevent suspicious devices from joining the network. We suggest that such a focus can extend past the behaviors from a devices to the human actors using the device on air. Government and regulatory organizations may have access to records and other non-public information sources that speak to the trustworthiness of individuals both in and outside of travel scenarios. This information is already in use for air travel security in the US through programs like TSA Pre✓ [140], and the intent within NextGen [137] is to continue to study and implement new ways to use it. We propose a means of using this additional information in a trust management enhanced preventative-focused network defense strategy by deploying a hybrid of distributed and centralized multi-trust models specific to aircraft and in-flight use cases that incorporates multiple trust data sources both from within and external to the air travel scenario, evolving trust values over the duration of a flight to control network access and inform network defense. Our trust model provides a basis for an Aviation Security Authority to insert trust metrics into this system, similar to the way Transportation Security Administration (TSA)'s uses Secure Flight [141] to inform ticketing agents. By combining this information with other on and off-network observations while an aircraft is in flight, an ASA can more efficiently target network defenses to discover network attacks that would be otherwise undiscoverable and make better decisions while aircraft is in-flight to prevent an attack. Similar uses of trust for intrusion detection [76] have been worked on in the past, but not applied to a complex air travel scenario.

9.2 Proposed Solution

The most significant challenge faced by a trust evaluation system in this environment is the need to form a trust opinion quickly with little direct past experience at hand. Passengers who have never flown before or have little to no history with the aircraft or airline do not have enough experience to form trust easily. I address this problem by augmenting traditional peer reputation-based trust computation with third-party input from airport, airline industry, and security entities with knowledge of each passenger and their past use of air travel services. Computed trust values can be used by passengers to make decisions about connectivity and resource sharing, or by an ASA or ISP to determine which in-flight network is potentially at risk so that decisions can be made to engage defensive countermeasures.

Figure 9.2 shows a high level view of the proposed trust model. The model computes a trust value for an entity (passenger, aircraft, etc.) from four trust metrics calculated from known facts about the entity (past actions, network activity, etc.). This computed value is evaluated against a policy to decide if the entity under evaluation is trusted, or untrusted.These evaluations are used to make change in the operating environment (e.g. the network state) either by granting or denying access, deploying additional network monitoring, removing services from the network, or other actions in response to trust calculations (further discussed in Section 9.3.3). These changes then effect future trust metrics (good or bad recommendations are created, ASA or CAA make observations, etc.), and the trust computation evolves informing new trust calculations. I detail this operating model throughout this chapter, and define it's building blocks in the next several sections.

9.2.1 Trust Data Storage

Trust data is defined as all facts used to compute trust. Storage of trust data has many constraints. First, our trust model must be capable of computing trust at any time, even

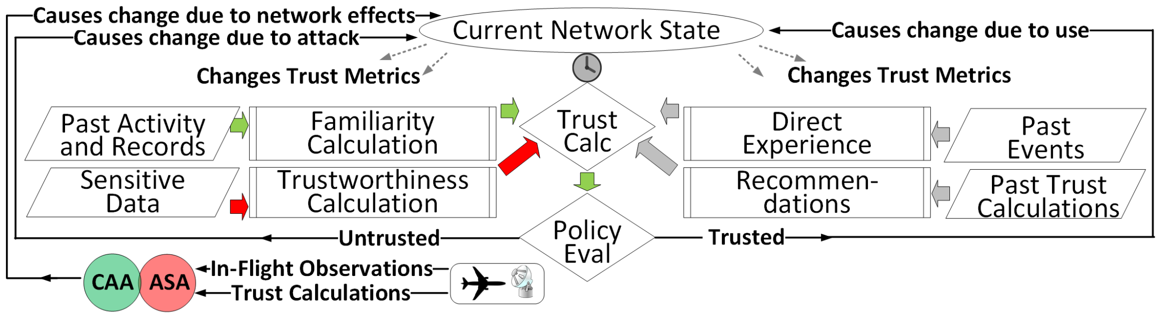

Figure 9.2: High Level Trust Model with Defensive Application

when required data sources are offline. Second, we must not create additional demand on airline industry systems that currently store this data. Individual passengers will have limited storage needs compared to airlines and airports as they interact with large volumes of passengers, flights, aircraft and airports. Third, computation must occur in an environment authorized to store private or sensitive data. This prevents completely distributed computation. Lastly, any centralized storage and computation environment must be robust as it may become overwhelmed by computational volumes needed to support modern day air traffic.

We suggest using a private commercial cloud infrastructure for storage and calculation, as it is an ideal scalable, *always-on* environment. If data is forwarded by providers to this environment, the cloud can be scaled as demand scales, optimizing cost and availability without additional demand on airlines and airports, some of which may not operate 24/7. This also prevents consumers of those calculations from directly accessing the data that is used. Additionally, performing calculations within a central place greatly reduces network overhead introduced by distributed trust models. We recommend that the cloud environment be owned by a trusted and airline-independent third party, such as the CAA, so that it can consume private passenger records with less risk of misuse.

In addition to passenger travel records and air travel service interactions, our model uses an additional metric computed by the ASA. This metric carries more sensitivity and is more restrictive in use. We choose to store this data within a second sensitive cloud authorized by the ASA to store sensitive information. We could place all data and all calculations within this sensitive cloud, but we choose to split our storage and calculations between a private commercial and a sensitive ASA cloud to give more options of operating modes and configurations. We define these metrics and calculations in more detail later in this paper.

9.2.2 Trust Metrics

Determining trust is a multidimensional problem, and often a computational process derived from a situation-specific calculation using subjective facts known to a particular individual. For example, research suggests that consumer trust when using e-commerce applications differs from physical storefronts, as the former is based on trust in underlying technology while the latter is often based on personal interactions with the individuals within [142]. Intuition suggests that passengers form trust differently with airlines than they would with other passengers. We capture these differences by performing trust calculations differently depending on the entity computing trust. To calculate a final trust value (detailed in Section 9.2.4), we first assemble four core *trust metrics* used in its calculation that characterize facts about participants in the air travel use case. We outline these metrics in this section.

Familiarity

One metric commonly associated with trust is familiarity, or the knowledge of a particular entity based on past experiences [143]. Zhang and Ghorgani calculate familiarity [144] in the context of e-commerce as a composition of perceptions of repeated experiences and human factors from the consumer such as the rate at which humans forget those past experiences. When considering familiarity of an airline to a passenger, trust formation is comparable. Familiarity in the other direction (from the airline to the passenger), however, may not possess a comparable construct to these human factors. For example, an airline cannot be holistically queried for perceptions of other entities. Instead, we extend this familiarity definition to leverage additional values gained from information exchanged between passenger and airline. Passengers who buy into airline membership programs often submit sensitive personal information, becoming more familiar to the airline. Airports become more familiar to airlines (and vice versa) when airlines use their gates or establish a flight hub. In all of these cases, the acts required in the formation of familiarity take effort and financial commitment, making them hard to falsify.

Different entities may have different standards for familiarity. To allow for a computation requester to dictate this standard, we use a policy object that contains static weights for each type of experience. This yields the following equation to compute familiarity between entity x and entity y:

$$familiarity_{xy} = \sum_{i=0}^{|E_{xy}|} E_{xy}[i] * w_{xy}[i], \tag{9.1}$$

where E is an array of past experience values particular to the x,y relationship, w is the array of weights for those experiences. Format of the array E_{xy} are the same throughout the entire system. Table 9.1 is a sample set of facts we include in calculating E, including multi-entity metrics that an CAA can collect. When initially computed, *familiarity* as

Table 9.1: Familiarity Trust Data

$E_{airlineairport}$	# past flights, # gates, hasHub
$E_{airlinepassenger}$	# past flights, flyer miles, program memberships, future bookings, isEmployee, hasCreditAcct
$E_{ISPpassenger}$	pastFlightTime, hasAcct, hasHomeService
$E_{CAApassenger}$	# lifetime flights, # lifetime miles

an unbounded numerical value, which could easily bias our weighted sum. We must bin these values into a fixed numerical scale. The *Likert Scale* is a frequently used ranking scale employed to measure human attitudes and responses in survey questions [145]. To minimize complexity, we use a similar linear scale: f_y = [-2.0, +2.0] where -2.0 is a completely

Algorithm 8 Compute Familiarity Value

```
1: procedure ComputeF(Trustor $x$,Trustee $y$,weights $w$)
2:     $f \leftarrow 0$
3:     Retrieve $E_{xy}$ records from database
4:     for $i \in \{1, ..., n\}$ do
5:         $familiarity$ += $E_{xy}$ [i] * $w_{xy}[i]$
6:     $f \leftarrow$ normalize($familiarity$)                ▷ to scale [-2.0, 2.0]
7:     Return $f$
```

unfamiliar entity, and +2.0 is a most familiar entity. But familiarity alone cannot compute a robust trust opinion, as malicious entities can be familiar ones, and passengers who have never flown before may be non-malicious.

Trustworthiness

Humans also form trust based on the perception of character, sometimes due to the roles or positions individuals hold. As previously mentioned, *NextGen* indicates an interest in leveraging passenger characteristics stored in various clearance and pre-screening databases to enhance airport security. We supplement that notion with the assumption that certain job functions within the airline industry and other industries such as law enforcement and first responders, carry with them implicit features that indicate trust. We use these characteristics to form a metric called *trustworthiness (t)*, computed in the sensitive cloud by the ASA or other authorized party with access to necessary passenger information. To remain consistent with other metrics, we use a similar linear scale: $t_y = [-2.0, +2.0]$ where -2.0 denotes a least trustworthy entity, +2.0 denotes a most trustworthy entity. Similarly, organizations such as airlines and airports can potentially have law enforcement specified risk ratings based on past history of criminal activity, etc. We leave implementation details of how an ASA explicitly computes trustworthiness values as such methods involve sensitive process and tradecraft known only to the ASA.

Direct Experience

Many existing trust systems use direct experiences as a basis for trust calculations [146] [81] [87]. Our third trust metric is also based on perceptions from direct past experience, however, we expect the existence of this metric to be rare as a passenger may not have any direct experience with other passengers unless they are traveling with a large familiar group. Passengers may also not have much experience with the aircraft, airline, or in-flight ISP unless they are frequent flyers. In our model, passenger direct experience $d_y = [-2.0, +2.0]$ where a value of -2.0 represents a least favorable past experience up to +2.0 for a most favorable one.

In some cases business entities such as airlines can also form direct experiences for passengers. For example, flight-crew, trained to identify suspicious behavior can provide experiences in a similar way to TSA's existing use of *See Something Say Something* [147] to report suspicious behavior to law enforcement. The on-board ISP can also form experiences based on observed network activity and its identification of suspicious behavior within the network that connects the in-flight satellite traffic to the Internet. These kinds of observations, however, may be subjective and error prone, and will need to be adjudicated by a trusted third party (the ASA) before they can be injected into the trust system to reduce the effects of false positives. Additionally, such experiences cannot be shared with passengers directly as public knowledge of them may potentially divulge sensitive defensive and detection methods used by the ISP or trusted third party, present privacy problems, or could otherwise be used as a tool of abuses by one passenger towards another (e.g. flooding false reports). We explain in Section 9.2.6 how these valuable observations can be used by the trust model to enable air travel security to dynamically compute trust during flight.

Recommendations

Our last trust metric is based on recommendations of peers within the trust system [83]. While a single passenger may not have much experience with other entities, other passengers within the airport or on-board the flight may and can be a source for peer recommendations.

Many reputation-based trust systems rely on the ability to easily broadcast requests for recommendations to peers on a local network. Completed trust calculations are stored within the cloud for future use as recommendations. This enables the trust system to obtain recommendations without broadcast overhead, even when the providing entity is offline.

While consumers are generally permitted to form their own opinions about commercial entities based off of personal experience and share them publicly, other business entities (such as airlines or airports) may fear legal issues when forming opinions about other businesses. It may also be difficult for a large entity to make a determination about another entity without significant data collection effort. Trust data that could be used to calculate such a recommendation are already captured within our familiarity metric. For these reasons, we will not include commercial entity to commercial entity recommendations.

9.2.3 Data Privacy and Anonymous Computation

We assume that all communication channels in our model use Secure Sockets Layer (SSL) or some other secure tunnel and that ISPs have obtained informed consent from passengers so they can freely monitor network traffic. These entities are responsible for implementing sanitization requirements that may exist before forwarding observations to the sensitive cloud. We believe this to be in-line with existing commercial network monitoring privacy rules (additional details are left out of the scope of this paper).

Secure multi-party computation, homomorphic encryption, and other cryptography based distributed computation techniques [148] provide the potential for partially mitigating privacy concerns by allowing distributed computation of encrypted data without the need for decryption. It is unclear if the benefits of using these methods overcomes their high communication overhead, or if implementation can be accomplished without leaking information. For example, if the entity initiating computation varies the thresholds used to determine if an entity is trusted or not, they may eventually recover the values used regardless of the protection methods placed on the inputs. Additionally, rules for handling sensitive information may not accommodate these new techniques and satisfy the risk tolerance of data owners. This may result in the need to prevent data from leaving a sensitive network regardless of it's level of encryption.

Our model allows users to specify weights in a policy object that they pass to the cloud for use in computed trust value. While in most cases this prevents users from seeing the source data used in calculations of these individual metrics, it would be possible for a user of the system to manipulate their policy ranges to expose raw trustworthiness, familiarity, and recommendation values for any party. For example, if all weights are set to 0 except for trustworthiness with a weight of 1.0, the returned value would be that entity's trustworthiness value.

To mitigate this, we combine the use of pseudonyms and a limited-use token granting system to insulate true identities, and prevent mass-computation activities. Our system implements a true name to pseudonym map within the commercial and sensitive clouds. When a trust calculation is needed, the trustee requests a new token from the commercial cloud and gives it to the trustor. Trust calculation requests are sent to the cloud along with this token, so the cloud service knows the request is authorized. The token expires after a period of time or upon an event that matches the nature of the original request (single use, flight-duration only, etc.)

Use of tokens and pseudonyms in this way, however, does not prevent exposure of passenger identity in cases when combining an entity's direct experience with tokens at calculation submission time. Entities have direct experience values based on another entity's true identity and must have a means to match them to a pseudonym when needed without exposing a link between them, thereby accidentally defeating it's anonymity benefits. This, coupled with policy value manipulation could enable an adversary to attack the trust model and recover information about those metrics.

To prevent this kind of exposure we could prevent entities from being able to combine direct experience with our trust model or use a global policy unknown to entities. These solutions, however, reduce the efficacy of the model and would not necessarily provide forward security. Instead, we require that true identities are never exchanged between passengers and extend our token granting system to issue additional pseudonyms when direct experience is recorded for other passengers. This is done either over the network for computer provided services between passengers, using a common key exchange protocol, or over direct transfer using *NFC*, a common data transfer capability on many mobile devices.

9.2.4 Computing Final Trust Value

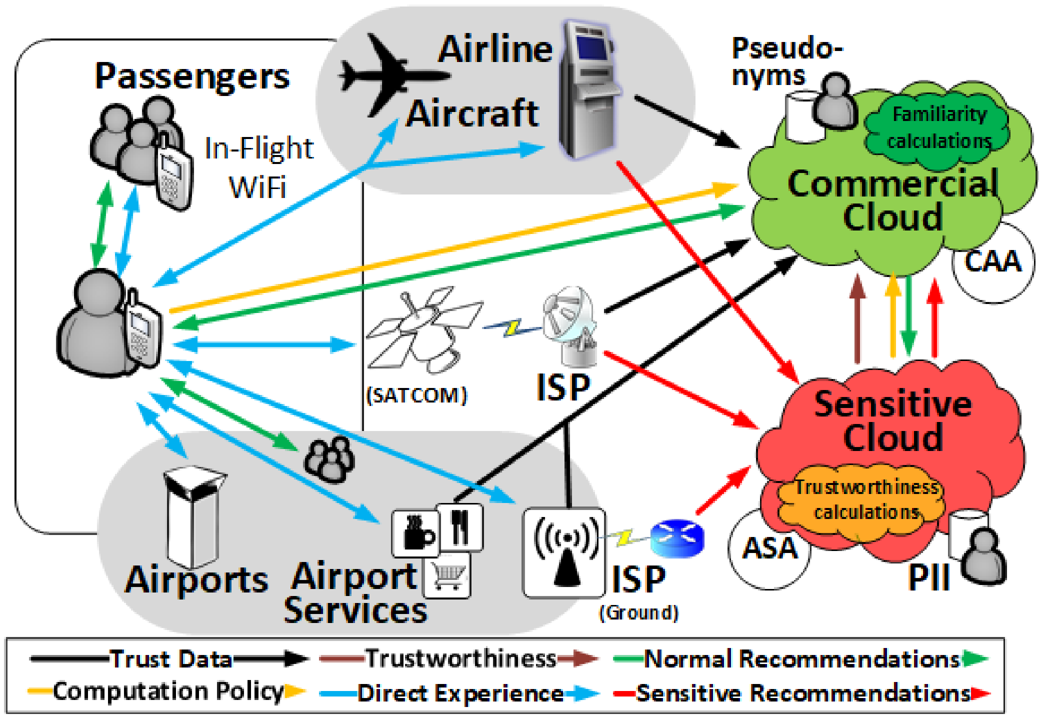

Figure 9.3: Trust Formation and Computation Overview

We show a high level overview of our model in Fig. 9.3, showing sources for familiarity, trustworthiness, direct experience, and recommendations along with their requisite storage locations. Here the final trust value is computed using Algorithm 9. To prevent any passenger from abusively computing trust for arbitrary passengers (e.g. mass collection) a trustee runs the procedure *getToken* (line 1) to receive a single-use timed token. The trustor then sends this token along with a computation policy P to the cloud for processing, causing *ComputeTrust* (line 7) to execute. The token is checked to ensure it is valid, otherwise the request for computation is denied (Line 8). Then, values needed for the calculation are retrieved from a database within the cloud, and passed along with the policy parameter to a function that will evaluate those values against policy (line 13). Within that function, any computational shortcuts due to policy are evaluated. For example, a particular entity may wish to automatically trust (or distrust) any entity with a very high (or very low) *trustworthiness* value. Similarly a passenger may wish not to trust the recommendations of any other passengers, and rely solely on its own experience, or external metrics to make it's decisions. If none of the shortcut conditions are met a final trust value is computed as follows:

$$T_{x->y}{=}d_y * \alpha + \overline{f_y} * \beta + \overline{r_y} * \delta + t_y * \gamma \tag{9.2}$$

where $\alpha, \beta, \delta, \gamma$ are configurable weights stored in the policy object (line 16). Weights sum to 1.0, so that the final trust value will occur on the same scale as other metrics ([-2.0, +2.0]). $\overline{f_y}$ and $\overline{r_y}$ is the average of familiarity and recommendations values (respectively) appropriate for the truster x, and trustee y. Within the algorithm representation, P.weight represents the weight value found in the policy object that corresponds to the weights in 9.2.

9.2.5 Trust Evolution and Updates

Trust calculations are based on facts as they are known at a particular point in time. Entities can afterwards perform actions that generate new facts, which may result in a different trust value if re-computed. To maintain an accurate understanding of trust, users of the model must know when new data arrives so that they may request a new calculation. With distributed trust management systems, trust calculations are performed within the trustor, requiring that involved parties be online to receive new data via a broadcast. We

Algorithm 9 Compute Final Trust Value

```
1: procedure GetToken(Key $sk$, Pnym $nym2$, Time $expiry$)
2:     $timestamp \leftarrow$ current time
3:     $nonce \leftarrow$ random 256 bits
4:     Store ($nonce, nym2, timestamp, expiry$) in cloud database
5:     return (token $\leftarrow$ nym2, $\mathcal{E}_{sk}(nonce, timestamp)$)
6:
7: procedure ComputeTrust(Policy $\mathcal{P}$, Pnym nym1,Token $tkn$)
8:     if (!isValid($tkn$)) then
9:         return (null, ”denied”)
10:    Delete(tkn) so that it may not be reused
11:    Retrieve $\overline{f}$,$t$,$d$,$\overline{r}$ from cloud database for nym1
12:    for each $policy \in \mathcal{P}$ do
13:        if EvaluatePolicy($policy$,$\overline{f}$,$t$,$d$,$\overline{r}$) == 1 then
14:            T $\leftarrow$ Policy Specified Trust Value
15:        else
16:            T $\leftarrow d * P.\alpha + \overline{f} * P.\beta + \overline{r} * P.\delta + t * P.\gamma$
17:    Store T in cloud database
18:    if (T ¿ P.trustValueThreshold) then
19:        return (v, ”trusted”)
20:    return (v, ”untrusted”)
```

designed our trust model to run in a cloud environment both for privacy concerns outlined in Section 9.2.3, but also to mitigate these connectivity requirements as they do not fit well into the air travel scenario. Additionally, to prevent a broadcast storm to all users of the trust model, we employ a periodic trustor-initiated trust re-computation during the time where the value is needed, and prior to any new uses. We use this process in our motivating use cases in Section 9.2.6.

9.2.6 Motivating Use Cases

To discuss in more detail how our model is used to secure aircraft smart spaces and on-board networks, we explore three motivating use cases. The first two use cases concern a passenger computing trust for other passengers to decide to authorize sharing of resources, and computing trust for an aircraft's WiFi connection (respectively). The third use case builds upon the first two, and presents the trust-informed network defense and attack discovery scenario referenced throughout the paper.

Network Bootstrapping

For all of our use cases, a device has a known link to a single passenger, but a passenger can have multiple devices on the network. To join the in-flight network, passengers are required to register each device on the network (e.g. during the WiFi access purchase process). We extend this activity to include the use of a trust client on each registered device, that facilitates the trust protocol. Tethered devices do not require registration, as long as the device that interfaces directly with the in-flight network is registered. Our model aggregates activities across all devices for each passenger so that trust is established independent of which device a passenger is using. Cases where passengers gain access through another passenger's connection is out of scope for this paper, as we believe WiFi providers will not permit such an arrangement.

Use Case 1: Passenger Trusting Another Passenger

The first use case concerns a problem well explored by trust management research for nodes in Service Oriented Provisioning, Wireless Sensor Networks (WSNs), and secure ad-hoc network routing (see Related Work). One passenger (the *trustor*:*pax1*) wishes to compute trust for another passenger (the *trustee*:*pax2*) to determine if it is safe to allow them access to a service they are providing, such as media (music, movie) sharing. We use this use case to illustrate our token exchange protocol in Fig. 9.4, used when computing trust for passengers.

When the trustee decides to connect to the trustor's service, they retrieve an authorization token from the commercial cloud according to the procedure outlined previously in Algorithm 9. The trustee shares this token with the trustor, who submits this token along with it's computational policy in a trust computation request to the commercial cloud (*ComputeTrust* procedure in Algorithm 9). The connection is permitted if the trustee computes (9.3) to a trust level above the trustor's minimum threshold for that service. Periodically during this connection, the trustee requests a new token, and submits it to the trustor so they may recompute trust to ensure the trustee remains trusted. If the trustee computes

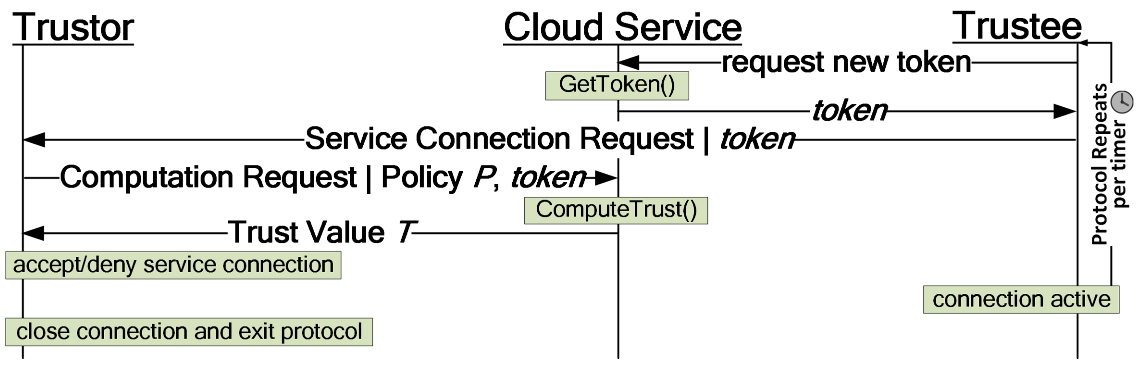
Trustor
Cloud Service
Trustee
request new token
GetToken()
token
Service Connection Request | token
Computation Request | Policy P, token
ComputeTrust()
Trust Value T
accept/deny service connection
close connection and exit protocol
connection active
Protocol Repeats per timer

Figure 9.4: Authorization Token and Trust Update Protocol

as untrusted, the trustor terminates their service connection.

$$T_{pax1->pax2}=d_{pax2} * \alpha + \overline{f_{pax2}} * \beta + \overline{r_{pax2}} * \delta + t_{pax2} * \gamma \tag{9.3}$$

Use Case 2: Passenger Trust Aircraft WiFi Network

In use case two, we compute trust for an aircraft entity. Here, a passenger (trustor:pax) calculates trust for the aircraft (trustee:a/c) they are boarding, to determine if it is safe to use it's WiFi connection. When the passenger has their boarding passes scanned at the ticketing desk (prior to entering the jetway), they scan a Quick Response (QR) code displayed nearby that contains an initial authorization token, with an expiration time of the current flight's expected duration. Upon scanning, the client begins to compute trust for the assigned aircraft over its current Internet connection (airport WiFi, the device's mobile carrier, etc). The trust calculation is shown in 9.4.

$$T_{pax->a/c} = d_{a/c} * \alpha + \overline{f_{a/c}} * \beta + \overline{r_{a/c}} * \delta + t_{a/c} * \gamma \tag{9.4}$$

$$t_{a/c} = \sum_{i=1}^{n} t_{pax_i} \tag{9.5}$$

Familiarity is computed as the average familiarity of the aircraft from its airline, current airport, and on-board ISP. Recommendations are collected from computations of fellow passengers and averaged. Trustworthiness is retrieved as a single value for the aircraft, as an average of all passenger trustworthiness values (9.5). As more passengers board the aircraft and while the aircraft is in flight, new trust calculations are made, evolving trust. Should the trust value drop below the passenger's threshold, the computation client will notify the user so they can disconnection from the network, or not connect in the future.

Use Case 3: Trust informed Network Defense

In our third use case, an ASA and an aircraft ISP work cooperatively to defend on-board networks. As previously mentioned, these are generally two distinct entities, as some countries have strict privacy and data sharing laws that prevent an ASA from network collection activities and prevent ISPs from violating privacy agreements with users outside of informed consent to be monitored. To maintain generality and simplicity, we consider further legal interpretations outside the scope of this paper.

For this scenario, an ASA computes trust for all aircraft within its operating domain, similarly to use case 2, but according to 9.6, where $d_{a/c}$ and d_{pax} represents secret or sensitive knowledge that the ASA knows about the aircraft or passenger and additional facts from adjudicated observations from flight crew, ISP, and other non-public or non-shareable information. Here, the ASA will use a policy object that specifies a higher weight for α and γ, as it most likely trusts its own analysis over the recommendations from passengers and inference from flight records. Other metrics are calculated in the same way as passenger initiated calculations. In many trust models, negative information about bad actors within the environment is distributed so that peers can make more accurate calculations of trust. In this scenario, however, sharing that information could expose defensive tradecraft and potentially tip off the network adversary. We explore this notion further in Section 9.3.4.

$$T_{ASA->a/c} = d_{a/c} * \alpha + \overline{f_{a/c}} * \beta + \overline{r_{a/c}} * \delta + t_{a/c} * \gamma \tag{9.6}$$

$$T_{ASA->pax}=d_{pax} * \alpha + \overline{f_{pax}} * \beta + \overline{r_{pax}} * \delta + t_{pax} * \gamma \tag{9.7}$$

Once computed, trust values for aircraft in flight can be ranked to determine which networks are least trusted so that the ASA may better target cost and computationally expensive defensive activities toward areas that most need them. These activities include specific requests to the ISP (or conduct them directly if authorized) to deploy additional security controls to perform more resource intensive monitoring (thereby collecting more information for the trust model) or take preventative measures to alter the network to prevent a perceived future attack. For example, networking hardware on-board the aircraft can be instructed to forward flow data, filtered to include only connections to or from that specific passenger's devices, to a ground-based group of network analysts. We rank a sample set of activities in Fig. 9.5 that could be taken in response to change in trust calculations, color coded to show actions taken within the ground ISP in light green, and those taken on the on-board network in light blue. Once protocols and patterns emerge that are abnormal, the traffic

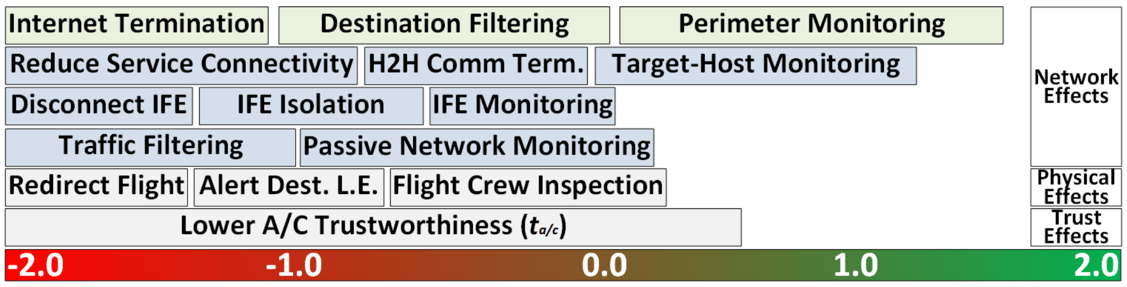

Figure 9.5: Sample ASA response based on trust level

itself can be captured or sampled and either routed back to other security tools on the ground, or examined in real time through a remote connection from an analyst or stored for evidential purposes. Each of these activities produces more observables for the trust system to consume, outside of their direct incident response value.

The addition of increased monitoring activity and other defensive observations in turn accelerates the convergence of trust to a stable value as new facts are produced and consumed by the trust management system. Should trust fall further, more intense monitoring occurs, eventually resulting in motivations for an ISP to perform traffic filtering or disable the network entirely, an on-board network operator to disconnect or isolate integrated services, or in the most extreme cases of flight compromise, an ASA ordering a flight to alter course. We explore this convergence in Section 9.3.2.

9.3 Evaluation and Discussion

Our model is fed by passenger records and service interactions that require time, effort, and monetary cost to produce, making them hard to falsify by an adversary. I augment this information with metrics from the ASA about individuals known to them, allowing the model to function even when passengers have never flown before or otherwise have few records or interactions to be used in computing their trust. Sensitive or private data sources used in trust calculations remain on isolated networks that are authorized to contain them, and are keyed to pseudonyms when stored. This implementation insulates analysis methods used by the ASA and it's partners (any external intelligence sources that support the ASA's mission), allowing only non-sensitive trust calculations to be used by entities within the model. This allows the model to leverage these sources without exposing them to unauthorized parties.

An ASA may not always have the capability or permission to collect information or prevent activities on in-flight networks, and a network provider most likely does not possess sensitive intelligence about those communicating on that network. When I apply my trust model to decision making in the defense of on-board networks, I enable new defensive capabilities for both of these groups when their individual knowledge intersects. In my model, when expensive tools and human defenders can be engaged only when needed and on a per-passenger or per-aircraft basis, they may be deployable remotely and across multiple aircraft, overcoming existing prohibitive computational, bandwidth, and personnel costs.

9.3.1 Policy Customization Concerns

Most passengers will have little knowledge of optimal weights, particularly with regard to externally provided trust metrics. Additionally, there is no practical way to consume differing policies along with passenger provided recommendations. Consequently, a set of incoming recommendations with vastly differing weights, introduces instability in confidence of the values the trustor receives. Re-computation of recommendations using the trustor's policy is an option, however this could become computationally prohibitive at scale, and performing calculations outside of my token granting infrastructure undermines the protections it provides. I could also include a policy checking process that compares the old policy with the new one to form a measure of difference, however it is unclear of its effect on the accuracy of final trust values.

Instead, in our experimental implementation, we chose to allow users to specify trust thresholds but protect the integrity of the recommendation system by calculating metrics universally and setting weights universally, otherwise I open the validation up to effects and bias that I do not yet fully understand. This act creates an additional layer of protection against the trust metric recovery attacks outlined in Section 9.2.3, however it requires that I form an initial set of weights that place the model in a reasonable operating state (see Section 9.3.5).

9.3.2 In-Flight Trust Convergence

A passenger who is initially trusted should become untrusted after performing a sufficient amount of malicious activities. Untrustworthy passengers should be prevented from computing as trusted until they demonstrate good behavior over a long period of time (perhaps

Table 9.2: Network Observation Weights

Activity	d_{ASA}	*Activity*	d_{ASA}
IP Scan	-0.1	Rogue Encrypted Tunnels	-1.0
Port Scan	-0.1	Service Probing	-1.5
Login Failures	-0.2	Infrastructure Login Attempts	-2.0

across multiple flights). These calculations should *converge* to values that agree with our intuition about how humans would determine trust. The speed at which the model converges in-flight will depend primarily on the value assigned to a particular observation by the ISP, flight crew, or other observer. We show sample reconnaissance activities with corresponding trust values in Table 9.2. Each of these activities can most likely be deployed directly into in-flight networks with a minimum space, weight, and cost investment. Network observations from passengers aren't the only sources for gaining awareness of network attacks. Activities such as frequent reboots of servers, significant performance degradation on network devices, incorrect routes or excessive traffic between otherwise quiet devices can be signs of trouble. While such behaviors are similar when a device is mis-configured or malfunctioning, they can also be the outcome of an advanced attacker's activity.

The desire to maximize safety forces to only consider potentially malicious observations in the proposed model, so that a passenger's trust value may only decrease during a flight. This is done primarily for two reasons. First is a practical reason, as investigating all benign activity on a network is computationally prohibitive as these activities will far outnumber malicious ones and will not produce indications of malicious acts. Second, is the desire to not introduce a means to attack the trust model itself by allowing an adversary to repair their reputation after initially producing a malicious act. This will contain malicious actors and protect the network for the entire duration of a flight until such time that the events can be more thoroughly investigated and a passenger can explain their behavior. This may result in an initial set of false positives, however follow-up tuning can adjust the scaling and contents of the events in the table, preventing false detections.

9.3.3 Trust Triggered Incident Response

The proposed model allows for targeted incident response through two avenues. First, response can be engaged when a flight contains an individual passenger who's trust values have fallen below a threshold (9.7). An ASA could request that traffic to or from that individual be examined with more advanced tools. Second, responses can be engaged when a flight's trust value has fallen below a threshold (9.6). As aircraft trust is based on the trust values of it's individual passengers, we would expect that detection would occur in the first case, sooner than the second case where trust values of benign passengers keep medians above a trust threshold longer.

9.3.4 Monitoring vs. Response

Advanced adversaries will monitor the networks they attack for signs of them being discovered. When network defenders respond by deploying active defenses, that act potentially acknowledges discovery to the attacker. A choice must be made to prevent further damage or instead continue to monitor and make additional observations about the adversary's tradecraft. A similar choice is made in the proposed model when the ASA chooses to lower trustworthiness values in response to reported activity. Defender actions may be discoverable, should an adversary be capable of establishing a link between its actions and lower trust values.

Some mitigations exist in this case. For example, an ASA could only return lower t values to calculations requested from passengers who are above a certain trust level, or could mask an individual passenger's value change by lowering the aircraft's t value instead. Ultimately, the ASA will need to be mindful of these issues and the affects of defensive action on adversary behavior.

9.3.5 Simulation Results

To assist us in assigning preliminary weights and thresholds we implemented portions of our model in Java, with simulated trust data and aircraft, airline, and airport compositions

Table 9.3: Expected v Computed T:α=.43,β=.06,δ=.07,γ=.44

d	f	r	t	E(T)	O(T)
2.0	-2.0	2.0	-2.0	0.00	0.00
2.0	-2.0	-2.0	-2.0	-0.10	-0.28
0.0	2.0	0.0	-2.0	-1.00	-0.76
0.0	-2.0	2.0	-2.0	-1.00	-0.86
0.0	-2.0	-2.0	-2.0	-1.50	-1.14
0.0	-2.0	0.0	2.0	1.00	0.76

based on public air travel statistics.

Selecting Initial Weights

Without experimental data and existing trust calculations, I had to construct a set of weights that agree with prevailing intuitions. For example, a passenger who has a low familiarity value and low trustworthiness should not be considered trusted without a maximally positive set of direct experiences and positive recommendations, and should only become slightly trusted so that bad future behaviors will result in a fast convergence to a negative trust value. Familiarity and trustworthiness are metrics derived from facts obtained from implicitly trusted sources (e.g. CAA, and ASA), while other passengers' direct experience is a potentially biased opinion. Intuition says familiarity and trustworthiness should have higher weights than opinions, and that our own experiences are more trustworthy than recommendations from other passengers. I experimented with various weight values that support this intuition, to produce the following initial set of weights in my policy: α=0.43, β=0.06, δ=0.07, γ=0.44. Table 9.3 show a sample of the resulting trust computations using these weights, for several user types and compare these with a subjective expected value that is in line with our intuition of trust.

Choosing an initial trust value threshold

The trust model is designed such that the shift of the 5 point Likert scale places the boundary between *trusted* and *untrusted* at 0.0. Users of the trust model must still assign a minimum trust values for trustees or aircraft WiFi, based on their individual risk appetite and an understanding of where the passenger population falls on that scale. I ran the simulator using weights from the previous section, with parameters based on public air travel statistics and estimates for passenger trustworthiness based on similarly sourced estimates of an individual's trustworthiness (I chose public statistics about credit ratings, criminal records, known traveler number assignments, and percentage of population with security clearances). Initially, no passenger has experiences with other passengers, causing this simulation to be exclusively based on f and t. Results are shown in the histogram in

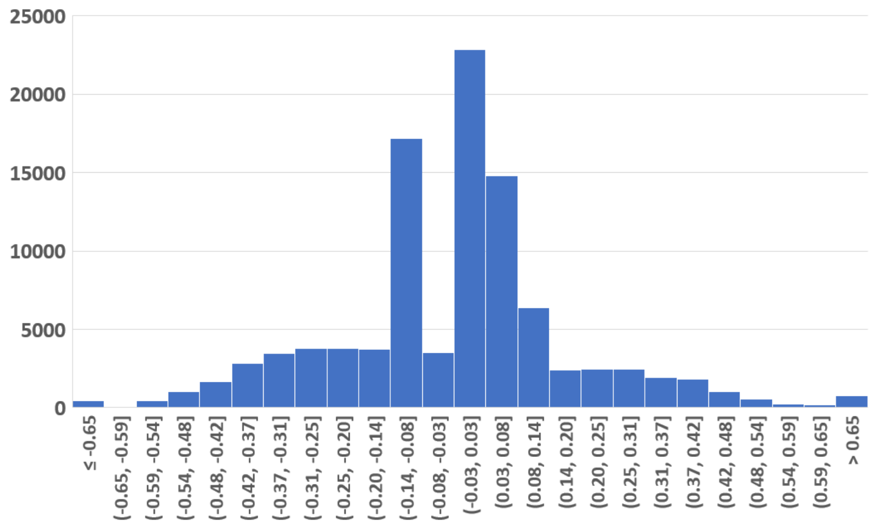

Figure 9.6: Baseline Trust Levels for Passenger Population

Figure 9.6. Four bins with most passenger counts are between -.14 and +.14. This result agrees with prevailing intuition as most of the population is unknown to the ASA and has not produced significant trust data for f, d, and r.

9.4 Conclusion

If the air travel industry is to remain competitive, it must adopt smart space concepts on air, and in doing so must also take actions to secure newly connected networks. In this chapter, I provided a trust model that can be used to make decisions about aircraft network safety so that security services can take actions that they previously would not have cause to engage, due to a lack of visibility into those networks. This model safely aggregates multiple public and private trust data sources into a unified model that all entities can use and contribute to, enabling network defense without prior knowledge of attack methods against these new networks.

The work presented in this chapter was published in *In-Flight Aircraft Smart Space Security using Multi-Entity Trust Evaluations,* Paul Seymer and Duminda Wijesekera. Proceedings of the 37th IEEE/AIAA Digital Avionics Systems Conference (DASC'18), September 2327, 2018, London, England [149]

Chapter 10: Dissertation Hypotheses, Discussion, and Evaluation

This dissertation proposes the following hypotheses:

1. Low cost, sustainably low-power Bluetooth-only sensor networks can be used to form a viable multi-space per sensor wireless smart parking solution based on radio localization without the need for additional sensor or network technology.

2. Smart parking and AV scheduling can be coupled to both incentivize and replace the *last-mile* use of congestion-creating personal vehicles within dense locales such as Washington DC, and George Mason University's Fairfax Campus.

3. Use of integrated wireless systems like smart parking with autonomous vehicle scheduling and tasking can provide enhancements to both.

4. Travel tracking and coordination enabled by my solution can be used to make travel safer, while maintaining traveler anonymity through the use of non-attribution technologies.

I validate and discuss these hypotheses in the next section.

10.1 Addressing Hypothesis 1

In Chapters 4, 5, and 8, I present my battery-powered self-forming routed sensor network constructed from low-cost commodity hardware with only Bluetooth radio as a physical carrier (data collection using BLE and data transfer using EDR/RFCOMM). For a parking lot of over 100 spaces, only between 7 and 11 nodes were needed to provide a high level of zoned occupancy detection accuracy. While some specific spaces within a lot may require additional sensoring, my solution demonstrates that multiple spaces can be covered by a single sensor node.

My experimental platform provides a high degree of solution flexibility due to its over-powered computational capability, however the cost of each node remains under $90US. Existing commodity Bluetooth beacons can be purchased for as low as $14US [150]. It follows that the per-node cost of a production-ready deployment of our solution would cost between these bounds.

10.2 Addressing Hypothesis 2

In Chapter 6, I outline a system that allows passengers to easily coordinate on-demand AV shuttles that integrate with mass transit options like the Washington DC metro. This system is expanded in Chapter 7 to address traffic congestion by further streamlining and incentivizing the use of AVs as a *last mile* travel mode. Coordinating and tracking travelers wirelessly allows for the seamless travel between modes, and the automated back-end tasking enabled by the data from this tracking allows for the travel mechanism provider to react to changes in travel conditions and passenger disposition, further saving the travel from such demands. Placement of AVs at strategic locations that fill existing gaps in travel modes connects underutilized mass transit with the destinations travelers wish to go to, preventing the use of congestion causing personal vehicles.

10.3 Addressing Hypothesis 3

In Chapter 7, I show experimental results from additional features of my smart parking solution that integrate with passenger support applications, such as a mobile app that allows travelers to indicate their final destinations. One of these features is a vehicle early-notification system, that detects incoming vehicles destined for the smart parking lot. By engaging an AV tasking solution in advance of parking, traveler wait times will be reduced. By utilizing the localization feature of my smart parking solution, AVs can also be tasked to (or close to) the traveler's vehicle, allowing parking lot owners to forgo construction of expensive passenger shelters, and further minimizing the time and effort burden's placed on travelers. These characteristics mutually incentivize the use of both systems.

10.4 Addressing Hypothesis 4

I show in Chapter 9, work that creates a trust establishment and management system that consumes public and private passenger data to make assessments of the fidelity of in-flight networks. I see the outputs from anonymized (but traceable by a trusted party) traveler tracking components of systems like my smart parking and tasking systems when applied to multi-modal travel that precedes flights as data sources to this system. Travel characteristics, and the patterns formed from them then become a foundational element for trust establishment in these systems, equipping network defenders and other components of air travel security with additional means to assess the traveling public. Privacy preserving features within my multi-modal travel tracking system, based on the use of disposable pseudonyms and existing privacy features of Bluetooth Low Energy (e.g. hardware address randomization) are found in Chapters 6 and 7, and are used in the trust management system presented in Chapter 9. Chapters 6 also further explores anonymity features used to track passengers during the embark, travel, and disembark process of *last-mile* travel with AVs.

10.5 Limitations and Future Work

I acknowledge the following limitations and outline future work:

- Test vehicles were few in number, as I did not have access to a large variety of vehicles.
- Experiments were short term. I seek to deploy our nodes on a long-term basis in the future to conduct more comprehensive real-world operational testing.
- Software support limits precision to a single RSSI. This may not ultimately matter but I suspect close-measured spaces are masked by this lack of precision.

- As my current experimental deployment to the West Campus Lot at GMU is underway, I was not able to include it in its entirety in this thesis, and will publish those results in future work.

- There is no real-time clock (RTC) on the Raspberry Pi, so I had to establish a WiFi connection to each deployed node to allow the operating system to set the system clock at the beginning of each experiment run. A soldier-on RTC solution for the Pi costs around $6US-$8US per node, but was not deployed prior to my experiments due to a perceived risk of unnecessarily damaging the nodes when attaching the module. While not influential in my results or my solution, it complicated the setup for each experiment run, and detracts from the purity of my *Bluetooth-only* solution claim.

- Experimental control such as the start and stop of record-keeping for each experiment run was achieved via SSH over WiFi to each node. I could have automated this process, but felt most confident in observing each node directly (to ensure it was receiving properly, etc.). This was done exclusively to control my experiments, and was not a requirement for the solution to function properly.

Chapter 11: Conclusions

In this dissertation, I present research that advances solutions to problems in smart parking and how it can be used to enable autonomous vehicle scheduling and dispatch. When systems like mine are deployed, travelers will have additional benefits available to them when deciding to replace the use of personal vehicles with multi-modal mass transit solutions.

11.1 Peer Reviewed Work

1. Paul Seymer, Chaitanya Yavvari, Duminda Wijesekera, and Cing-Dao Kan. ”Coordinating AV Dispatch with Smart Remote Parking”. To appear in proceedings of the 2nd IEEE Connected and Automated Vehicles Symposium (IEEE CAVS’19), September 2223, 2019, Honolulu, Hawaii, USA.

2. Paul Seymer, Chaitanya Yavvari, Duminda Wijesekera, and Cing-Dao Kan. ”Privacy preserving traveler tracking and on-demand multi-modal traffic informed tasking for autonomous vehicles near L’Enfant Plaza Metro”. Submitted to the 90th IEEE Vehicular Technology Conference (VTC’19-Fall), September 2225, 2019, Honolulu, Hawaii, USA.

3. Paul Seymer, Duminda Wijesekera, and Cing-Dao Kan. ”Smart Parking Zones using Dual Mode Routed Bluetooth Fogged Meshes”. In Proceedings of the 5th International Conference on Vehicle Technology (VEHITS’19), May 3-5, 2019, Heraklion, Crete, Greece.

4. Paul Seymer, Duminda Wijesekera, and Cing-Dao Kan. ”Secure Outdoor Smart Parking using Dual Mode Bluetooth Mesh Networks”. In Proceedings of the 89th

IEEE Vehicular Technology Conference (VTC'19-Spring), April 28 May 1, 2019, Kuala Lumpur, Malaysia.

5. Paul Seymer and Duminda Wijesekera. "In-Flight Aircraft Smart Space Security using Multi-Entity Trust Evaluations". In Proceedings of the 37th IEEE/AIAA Digital Avionics Systems Conference (DASC'18), September 2327, 2018, London, England

6. Paul Seymer and Duminda Wijesekera."Implementing Fair Wireless Interfaces with Off-The-Shelf Hardware in Smart Spaces". In Proceedings of the 18th International Conference on Internet Computing and Internet of Things (ICOMP '17), July 17-20, 2017, Las Vegas, Nevada, USA.

7. Charalampos Andrianakis, Paul Seymer, and Angelos Stavrou. "A Scalable Web Object Inspection and Malfease Collection". In Proceedings of 5th USENIX Workshop on Hot Topics in Security (HotSec '10). August 10th, 2010, Washington, DC.

8. Paul Seymer, Angelos Stavrou, Duminda Wijesekera, and Sushil Jajodia. "QoP and QoS Policy Cognizant Module Composition". In Proceedings for IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY '10). Fairfax, VA 2010.

Acknowledgments

The authors affiliation (pseymer@mitre.org) with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author. Approved for Public Release 19-2465. Distribution Unlimited.

Appendix A: List of Abbreviations

AV	Autonomous Vehicle
ITS	Intelligent Transportation System

Bibliography

Bibliography

[1] Mass transit is collapsing everywhere. TheHill. [Online]. Available: https://thehill.com/opinion/campaign/387498-mass-transit-is-collapsing-everywhere

[2] Weekly retail gasoline and diesel prices. US Energy Information Administration. [Online]. Available: https://www.eia.gov/dnav/pet/pet˙pri˙gnd˙dcus˙nus˙a.htm

[3] Trends in public transportation ridership: Implications for federal policy. Congressional Research Service. [Online]. Available: https://fas.org/sgp/crs/misc/R45144.pdf

[4] J. D. Hall, C. Palsson, and J. Price, "Is uber a substitute or complement for public transit?" *Journal of Urban Economics*, vol. 108, pp. 36 – 50, 2018.

[5] Arlington launches review of $1 million bus stop. ARLnow.com. [Online]. Available: https://www.arlnow.com/2013/06/25/arlington-launches-review-of-1-million-bus-stop/

[6] Uber. Uber Technologies Inc. [Online]. Available: https://www.uber.com/

[7] Study: Ride-sharing decreases public transit use. Chicago Tribune. [Online]. Available: http://www.govtech.com/fs/transportation/study-ride-sharing-decreases-public-transit-use.html

[8] A lot of cities want roboshuttles, including d.c. but will they work? The Washington Post. [Online]. Available: https://www.washingtonpost.com/local/trafficandcommuting/a-lot-of-cities-want-roboshuttles-including-dc-but-will-they-work/2018/11/25/8f178ebe-ed02-11e8-8679-934a2b33be52˙story.html

[9] From Model T to driverless: Ford to launch fleet of robot cars in washington. The Washington Post. [Online]. Available: https://www.washingtonpost.com/local/trafficandcommuting/from-model-t-to-driverless-ford-to-launch-fleet-of-robot-cars-in-washington/2018/10/21/6d98119e-d2f6-11e8-b2d2-f397227b43f0˙story.html

[10] Autonomous vehicle pilots across america. National League of Cities. [Online]. Available: https://www.nlc.org/resource/autonomous-vehicle-pilots-across-america

[11] Autonomous vehicles working group. Office of the Deputy Mayor for Planning and Economic Development. [Online]. Available: https://dmped.dc.gov/page/autonomous

[12] DC autonomous vehicles principles statement. Office of the Deputy Mayor for Planning and Economic Development. [Online]. Available: https://dmped.dc.gov/publication/dc-autonomous-vehicles-principles-statement

[13] Boston Consulting Group, "Reshaping urban mobility with autonomous vehicles - lessons from the city of boston," World Economic Forum, Tech. Rep., 2018.

[14] WMATA SmartTrip card. Washington Metropolitan Area Transit Authority (WMATA). [Online]. Available: https://smartrip.wmata.com

[15] WMATA trip planner. Washington Metropolitan Area Transit Authority (WMATA). [Online]. Available: https://www.wmata.com/schedules/trip-planner/

[16] Olli self-driving vehicle range. Local Motors. [Online]. Available: https://localmotors.com/meet-olli/

[17] parksol. (2018) Ultrasonic sensor with indication. [Online]. Available: http://www.parksol.lt/product/ultrasonic-sensor-with-indication/

[18] P. LLC. (2018) Parkmobile. [Online]. Available: https://parkmobile.io/

[19] C. de Coulomb, "Premier mémoire sur lelectricité et le magnétisme, mém. acad. roy. sci., 569 (1785)."

[20] Q. Dong and W. Dargie, "Evaluation of the reliability of rssi for indoor localization," in *2012 International Conference on Wireless Communications in Underground and Confined Areas*, Aug 2012, pp. 1–6.

[21] S. Oguejioforo., V. Okorogu, A. Abe, and O. OsuesuB., "Outdoor localization system using rssi measurement of wireless sensor network," in *International Journal of Innovative Technology and Exploring Engineering*, vol. 2, no. 2, January 2013.

[22] A. H. Olevall and M. Fuchs, "Indoor navigation and personal tracking system using bluetooth low energy beacons," Master's thesis, Uppsala University, 2017.

[23] R. Faragher and R. Harle, "An analysis of the accuracy of bluetooth low energy for indoor positioning applications," in *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, Tampa, Florida, Sep 2014, pp. 201 – 210.

[24] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[25] O. Silver, "An indoor localization system based on ble mesh network," Master's thesis, Linkoping University, 2016.

[26] F. S. Daniay and A. T. Cemgil, "Model-based localization and tracking using bluetooth low-energy beacons," *Sensors*, vol. 17, no. 11, 2017.

[27] *Bluetooth Mesh Profile Specification*, Bluetooth SIG Mesh Working Group Std., Rev. 1.0, Jul. 2017.

[28] H. Kim, J. Lee, and J. W. Jang, "Blemesh: A wireless mesh network protocol for bluetooth low energy devices," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 558–563.

[29] S. M. Darroudi and C. Gomez, "Bluetooth low energy mesh networks: A survey," *Sensors*, vol. 17, no. 7, p. 1467, 2017.

[30] S. Liniger, "Parking prediction techniques in an iot environment," Master's thesis, University of Zurich, 2015.

[31] H. Fabian, "A public parking management system for zurich," Master's thesis, University of Zurich, 2015.

[32] H. C. Yee and Y. Rahayu, "Monitoring parking space availability via zigbee technology," *International Journal of Future Computer and Communication*, vol. 3, no. 6, p. 377, 2014.

[33] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, and G. Luo, "Smartphone-based real time vehicle tracking in indoor parking structures," *IEEE Transactions on Mobile Computing*, vol. 16, no. 7, pp. 2023–2036, July 2017.

[34] L. Hobi, "The impact of real-time information sources on crowd-sourced parking availability prediction," Master's thesis, University of Zurich, 2015.

[35] Parking Panda. [Online]. Available: https://www.parkingpanda.com/

[36] P. Sadhukhan, "An iot-based e-parking system for smart cities," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1062–1066, 2017.

[37] E. E. Tsiropoulou, J. S. Baras, S. Papavassiliou, and S. Sinha, "Rfid-based smart parking management system," *Cyber-Physical Systems*, vol. 3, no. 1-4, pp. 22–41, 2017.

[38] E. Karbab, D. Djenouri, S. Boulkaboul, and A. Bagula, "Car park management with networked wireless sensors and active rfid," in *Electro/Information Technology (EIT), 2015 IEEE International Conference on*. IEEE, 2015, pp. 373–378.

[39] H. Al-Kharusi and I. Al-Bahadly, "Intelligent parking management system based on image processing," *World Journal of Engineering and Technology*, vol. 2, no. 02, p. 55, 2014.

[40] J. A. Vera-Gmez, A. Quesada-Arencibia, C. R. Garca, R. Surez Moreno, and F. Guerra Hernndez, "An intelligent parking management system for urban areas," *Sensors*, vol. 16, no. 6, 2016.

[41] T. Lin, H. Rivano, and F. Le Moul, "A survey of smart parking solutions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3229–3253, Dec 2017.

[42] Passport Labs, INC. Passportparking. [Online]. Available: https://www.passportinc.com/

[43] L. Mainetti, L. Patrono, M. L. Stefanizzi, and R. Vergallo, "A smart parking system based on iot protocols and emerging enabling technologies," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 764–769.

[44] B. M. K. Gandhi and M. K. Rao, "A prototype for iot based car parking management system for smart cities," *Indian Journal of Science and Technology*, vol. 9, no. 17, 2016.

[45] M. Patil and V. N. Bhonge, "Wireless sensor network and rfid for smart parking system," pp. 188–192, 2013.

[46] P. Saayman, I. Craig, and F.R.Camisani-Calzolari, "Optimization of an autonomous vehicle dispatch system in an underground mine," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 106, no. 2, pp. 77–86, 2006.

[47] Z. Song, H. Schunnesson, M. Rinne, and J. Sturgul, "Intelligent scheduling for underground mobile mining equipment," *PLOS ONE*, vol. 10, no. 6, pp. 1–21, 06 2015.

[48] F. Sun, Y. Pan, J. White, and A. Dubey, "Real-time and predictive analytics for smart public transportation decision support system," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–8.

[49] F. Sun, A. Dubey, J. White, and A. Gokhale, "Transit-hub: A smart public transportation decision support system with multi-timescale analytical services," *Cluster Computing*, pp. 1–16, 2017.

[50] Y. Wang, S. Ram, F. Currim, E. Dantas, and L. A. Saboia, "A big data approach for smart transportation management on bus network," in *2016 IEEE International Smart Cities Conference (ISC2)*, Sep. 2016, pp. 1–6.

[51] I. Skarga-Bandurova, M. Derkach, and I. Kotsiuba, "The information service for delivering arrival public transport prediction," in *2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, Sep. 2018, pp. 191–195.

[52] A. M. Said, E. Abd-Elrahman, and H. Afifi, "A comparative study on machine learning algorithms for green context-aware intelligent transportation systems," in *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, Nov 2017, pp. 1–5.

[53] X. Fu, M. Vernier, A. Kurt, K. Redmill, and U. Ozguner, "Smooth: Improved short-distance mobility for a smarter city," in *Proceedings of the 2Nd International Workshop on Science of Smart City Operations and Platforms Engineering*, ser. SCOPE '17. New York, NY, USA: ACM, 2017, pp. 46–51.

[54] M. Vernier, K. Redmill, U. Ozguner, A. Kurt, and B. A. Guvenc, "OSU SMOOTH in a smart city," in *2016 1st International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE) in partnership with Global City Teams Challenge (GCTC) (SCOPE - GCTC)*, April 2016, pp. 1–6.

[55] T. Janasz and U. Schneidewind, *The Future of Automobility.* Cham: Springer International Publishing, 2017, pp. 253–285. [Online]. Available: https://doi.org/10.1007/978-3-319-40967-2˙13

[56] M. Kumru, E. Debada, L. Makarem, and D. Gillet, “Mobility-on-demand scenarios relying on lightweight autonomous and connected vehicles for large pedestrian areas and intermodal hubs,” in *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, Sep. 2017, pp. 178–183.

[57] H. Mathisen, “A mobile application for booking autonomous vehicles-combining the sharing economy and self-driving cars,” Master’s thesis, NTNU, 2018.

[58] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar, and J. Light, “The personal server: Changing the way we think about ubiquitous computing,” in *UbiComp 2002: Ubiquitous Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2498, pp. 194–209.

[59] D. Husemann, C. Narayanaswami, and M. Nidd, “Personal mobile hub,” in *Proceedings of the Eighth International Symposium on Wearable Computers (ISWC ’04)*, vol. 1, Oct 2004, pp. 85–91.

[60] F. J. Ballesteros, E. Soriano, and G. Guardiola, “Octopus: An upperware based system for building personal pervasive environments,” *Journal of Systems and Software*, vol. 85, no. 7, pp. 1637 – 1649, 2012, software Ecosystems.

[61] F. Ballesteros, E. Soriano, K. Leal, and G. Guardiola, “Plan b: an operating system for ubiquitous computing environments,” in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, March 2006, pp. 10 pp.–135.

[62] F. Ballesteros, G. Guardiola, K. Leal, and E. Soriano, “Omero: ubiquitous user interfaces in the plan b operating system,” in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, March 2006, pp. 5 pp.–83.

[63] A. Bennaceur, P. Singh, P.-G. Raverdy, and V. Issarny, “The ibicoop middleware: Enablers and services for emerging pervasive computing environments.” in *PerCom Workshops.* IEEE Computer Society, 2009, pp. 1–6, 978-1-4244-3304-9.

[64] B. Johanson, A. Fox, and T. Winograd, “The interactive workspaces project: Experiences with ubiquitous computing rooms,” *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 67–74, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2002.1012339

[65] M. Romn, C. Hess, R. Cerqueira, R. H. Campbell, and K. Nahrstedt, “Gaia: A middleware infrastructure to enable active spaces,” *IEEE Pervasive Computing*, vol. 1, pp. 74–83, 2002.

[66] T. Pering, K. Lyons, R. Want, M. Murphy-Hoye, M. Baloga, P. Noll, J. Branc, and N. De Benoist, “What do you bring to the table?: Investigations of a collaborative

workspace," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, ser. UbiComp '10, 2010.

[67] Steelcase Inc. (2015, (Accessed July 15, 2015)) media:scape. [Online]. Available: http://www.steelcase.com/products/technology/collaboration

[68] D. Molyneaux, S. Izadi, D. Kim, O. Hilliges, S. Hodges, X. Cao, A. Butler, and H. Gellersen, "Interactive environment-aware handheld projectors for pervasive computing spaces," in *International Conference on Pervasive Computing.* Springer, 2012, pp. 197–215.

[69] Microsoft Corp. (2015, (Accessed March 10, 2015)) Remote desktop protocol. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa383015%28v=vs.85%29.aspx

[70] RealVNC. (2015, (Accessed March 10, 2015)) Vnc. [Online]. Available: https://www.realvnc.com

[71] M. Annamalai, A. Birrell, D. Fetterly, and T. Wobber, "Implementing portable desktops: a new option and comparisons," Microsoft Research, Tech. Rep. TR-2006-151, October 2006. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=55966

[72] R. Cáceres, C. Carter, C. Narayanaswami, and M. Raghunath, "Reincarnating pcs with portable soulpads," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 65–78. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067179

[73] A. Ferscha and S. Vogl, "Pervasive web access via public communication walls," in *Proceedings of the First International Conference on Pervasive Computing*, ser. Pervasive '02. London, UK, UK: Springer-Verlag, 2002, pp. 84–97. [Online]. Available: http://dl.acm.org/citation.cfm?id=646867.706696

[74] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *In Proceedings of IEEE Symposium on Security and Privacy (SP '96)*, 1996, pp. 164–173.

[75] L. McNamara, C. Mascolo, and L. Capra, "Trust and mobility aware service provision for pervasive computing," in *1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI)*, 2006, pp. 603–610.

[76] F. Bao, R. Chen, M. Chang, and J.-H. Cho, "Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection," *IEEE transactions on network and service management*, vol. 9, no. 2, pp. 169–183, 2012.

[77] J.-H. Cho, A. Swami, and R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1001–1012, 2012.

[78] F. Bao, R. Chen, and J. Guo, "Scalable, adaptive and survivable trust management for community of interest based internet of things systems," in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS '13)*. IEEE, 2013, pp. 1–7.

[79] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 5, pp. 1253–1266, 2014.

[80] B. Carminati, E. Ferrari, and M. Viviani, "Security and trust in online social networks," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 4, no. 3, pp. 1–120, 2013.

[81] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 47, 2013.

[82] D. Quercia, M. Lad, S. Hailes, L. Capra, and S. Bhatti, "Strudel: Supporting trust in the dynamic establishment of peering coalitions," in *Proceedings of the 2006 ACM symposium on Applied computing*. ACM, 2006, pp. 1870–1874.

[83] D. A. Al-Arayed and J. P. Sousa, "General trust management-gtm," in *IEEE Second International Conference on Social Computing (SocialCom '10)*. IEEE, 2010, pp. 857–864.

[84] L. Capra, "Engineering human trust in mobile system collaborations," in *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 6. ACM, 2004, pp. 107–116.

[85] D. Quercia, S. Hailes, and L. Capra, "Trullo-local trust bootstrapping for ubiquitous devices," in *Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, (MobiQuitous '07)*. IEEE, 2007, pp. 1–9.

[86] S. Buchegger and J.-Y. Le Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2002, pp. 226–236.

[87] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.

[88] P. Michiardi and R. Molva, *Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks*. Boston, MA: Springer US, 2002, pp. 107–121.

[89] S. Bansal and M. Baker, "Observation-based cooperation enforcement in ad hoc networks," *CoRR*, vol. cs.NI/0307012, 2003.

[90] M. Gil Pérez, F. Gómez Mármol, G. Martínez Pérez, and A. F. Skarmeta Gómez, "RepCIDN: A reputation-based collaborative intrusion detection network to lessen the impact of malicious alarms," *Journal of Network and Systems Management*, vol. 21, no. 1, pp. 128–167, Mar 2013.

[91] Symantec Corporation. (2018, (Accessed May 15, 2018)) Cb protection. [Online]. Available: http://www.symantec.com/reputation-based-security/

[92] Carbon Black, Inc. (2018, (Accessed May 15, 2018)) Symantec insight. [Online]. Available: https://www.carbonblack.com/products/cb-protection/

[93] M. Weiser, "Some computer science issues in ubiquitous computing," *Commun. ACM*, vol. 36, no. 7, pp. 75–84, Jul. 1993. [Online]. Available: http://doi.acm.org/10.1145/159544.159617

[94] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The case for cyber foraging," in *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*, ser. EW 10. New York, NY, USA: ACM, 2002, pp. 87–92. [Online]. Available: http://doi.acm.org/10.1145/1133373.1133390

[95] Barco, Inc. (2015, (Accessed March 10, 2015)) Clickshare cse-200. [Online]. Available: http://www.barco.com

[96] WiFi Alliance. Wi-fi certified miracast. [Online]. Available: http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast

[97] Roku Inc. Roku. [Online]. Available: https://www.roku.com

[98] IOGear Inc. Miracast adapter. [Online]. Available: https://www.iogear.com/product/GWAVR/

[99] Alberto Fanjul (albfan). (2015, (Accessed March 10, 2015)) Miraclecast. [Online]. Available: https://github.com/albfan/miraclecast

[100] ZXing, "ZXing Multi-format 1D/2D barcode image processing library," 2012. [Online]. Available: http://code.google.com/p/zxing/

[101] Google Developers. (2017, (Accessed February 1, 2017)) Google beacons. [Online]. Available: https://developers.google.com/beacons/

[102] M. Krzywinski, "Port knocking from the inside out," *SysAdmin Magazine*, vol. 12, no. 6, pp. 12–17, 2003.

[103] J. Lyu. (2016, (Accessed June 10, 2016)) Bluetooth virtual keyboard. [Online]. Available: https://github.com/lvht/btk

[104] P. Seymer and D. Wijesekera, "Implementing fair wireless interfaces with off-the-shelf hardware in smart spaces," in *Proceedings of the 18th International Conference on Internet Computing and Internet of Things (ICOMP'17)*, July 2017, pp. 79–85.

[105] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piche, "A comparative survey of wlan location fingerprinting methods," in *6th Workshop on Positioning, Navigation and Communication*, March 2009, pp. 243–251.

[106] Bluetooth SIG. (2018) Bluetooth technology options. [Online]. Available: https://www.bluetooth.com/bluetooth-technology/topology-options

[107] karulis. (2018) Pybluez - python extension module allowing access to system bluetooth resources. [Online]. Available: https://github.com/pybluez

[108] M. Holtmann and J. Hedberg. (2018) Bluez - official linux bluetooth protocol stack. [Online]. Available: http://www.bluez.org/

[109] StarTech. (2018) Mini usb bluetooth 4.0 adapter - 50m (165ft) class 1 edr wireless dongle. [Online]. Available: https://www.startech.com/Networking-IO/Bluetooth-Telecom/USB-Bluetooth-4-Dongle USBBT1EDR4

[110] Project Ubertooth. (2018) Ubertooth - an open source 2.4 ghz wireless development platform suitable for bluetooth experimentation. [Online]. Available: http://ubertooth.sourceforge.net/

[111] M. Wooley and S. Schmidt. (2018) Bluetooth 5. [Online]. Available: https://www.bluetooth.com/ /media/files/marketing/bluetooth˙5-final.ashx

[112] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[113] P. Seymer, D. Wijesekera, and C.-D. Kan, "Secure outdoor smart parking using dual mode bluetooth mesh networks," in *Proceedings of the 89th IEEE Vehicular Technology Conference (VTC'19-Spring)*, April 2019.

[114] JeVois Inc. (2018) Jevois-a33 smart camera. [Online]. Available: https://www.jevoisinc.com/pages/hardware

[115] L. Itti. (2018) Darknet yolo jevois module. [Online]. Available: http://jevois.org/moddoc/DarknetYOLO/modinfo.html

[116] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[117] P. Seymer, D. Wijesekera, and C.-D. Kan, "Smart parking zones using dual mode routed bluetooth fogged meshes," in *To appear in 5th International Conference on Vehicle Technology and Intelligent Transport Systems.*, May 2019.

[118] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[119] MIFARE. NXP Semiconductors Austria GmbH Styria. [Online]. Available: https://www.mifare.net/en/

[120] Duo mobile: Secure two-factor authentication app. Duo. [Online]. Available: https://duo.com/product/trusted-users/two-factor-authentication/duo-mobile

[121] P. A. Grassi, J. Fenton, E. Newton, R. Perlner, A. Regenscheid, W. Burr, J. Richer, N. Lefkovitz, J. Danker, Y. Choong *et al.*, "NIST special publication 800-63b. digital identity guidelines: Authentication and lifecycle management," *Bericht, NIST*, 2017.

[122] Google authenticator. Google Inc. [Online]. Available: https://github.com/google/google-authenticator

[123] Lyft is now suggesting more convenient pickup locations, because a little walking won't kill you. The Verge. [Online]. Available: https://www.theverge.com/2017/6/26/15875220/lyft-suggested-pickup-location-walking

[124] Types of QR code. DENSO WAVE, Inc. [Online]. Available: https://www.qrcode.com/en/codes/

[125] Radius networks developer blog - fundamentals of beacon ranging. Radius Networks. [Online]. Available: https://developer.radiusnetworks.com/2014/12/04/fundamentals-of-beacon-ranging.html

[126] Bluetooth technology protecting your privacy. Bluetooth SIG, Inc. [Online]. Available: https://blog.bluetooth.com/bluetooth-technology-protecting-your-privacy

[127] SW Ecodistrict plan: A vision for a more sustainable future. National Capital Planning Commission. [Online]. Available: https://www.ncpc.gov/initiatives/swecodistrict/

[128] Autonomous vehicles accord: Global cities' vision for the future. Bloomberg Philanthropies and The Aspen Institute. [Online]. Available: http://AVsInCities.Bloomberg.org)

[129] Capital bikeshare. Motivate International, Inc. [Online]. Available: https://www.capitalbikeshare.com/

[130] D. Metro, "Metrorail ridership data download, october 2015," https://planitmetro.com/2016/03/14/metrorail-ridership-data-download-october-2015/, (Accessed on 03/01/2019).

[131] DDOT, "2016 traffic volume — open data dc," http://opendata.dc.gov/datasets/0162d6516abb4bb48ce3b171dac583e5˙130, 3 2019, (Accessed on 03/01/2019).

[132] P. Seymer, C. Yavvari, D. Wijesekera, and C.-D. Kan, "Privacy preserving traveler tracking and on-demand multi-modal traffic informed tasking for autonomous vehicles near l'enfant plaza metro," in *Submitted to the 90th IEEE Vehicular Technology Conference (VTC'19-Fall).*, September 2019.

[133] Planned growth to strain traffic around stanford. Palo Alto Online. [Online]. Available: https://www.paloaltoonline.com/news/2017/10/20/proposed-growth-to-strain-stanfords-traffic-fighting-effort

[134] Increased traffic around athens, on uga campus leads to frustration. TheRed&Black. [Online]. Available: https://www.redandblack.com/athensnews/increased-traffic-around-athens-on-uga-campus-leads-to-frustration/article˙65c4d6c4-ce90-11e8-911f-1b18f0b21305.html

[135] P. Seymer, C. Yavvari, D. Wijesekera, and C.-D. Kan, "Coordinating av dispatch with smart remote parking," in *To appear in 2nd IEEE Connected and Automated Vehicles Symposium (IEEE CAVS'19).*, September 2019.

[136] ASIG. (2016) Onboard: IFE server and access terminal. Avionics and Systems Integration Group. [Online]. Available: https://www.asigllc.com/products/onboard-ife.html

[137] N. B. Gerald L Dillingham, Gregory C Wilshusen, "Air traffic control : FAA needs a more comprehensive approach to address cybersecurity as agency transitions to NextGen," United States Government Accountability Office, Tech. Rep., Apr. 2015.

[138] H. Teso, "Aircraft hacking: Practical aero series," 2013, 4th Annual HITB Security Conference. [Online]. Available: http://conference.hitb.org/hitbsecconf2013ams/hugo-teso/

[139] S. Ragan. (2013, April) FAA dismisses 'planesploit' creator's claims. Security Week. [Online]. Available: http://www.securityweek.com/faa-dismisses-planesploit-creators-claims

[140] Transportation Security Administration. (2017, Accessed November 20, 2017) TSA precheck. [Online]. Available: https://www.tsa.gov/precheck

[141] ——. (2014, September) TSA secure flight program. Transportation Security Administration. [Online]. Available: https://www.tsa.gov/news/testimony/2014/09/18/tsa-secure-flight-program

[142] D. J. Kim, Y. I. Song, S. Braynov, and H. Rao, "A multidimensional trust formation model in b-to-c e-commerce: a conceptual framework and content analyses of academia/practitioner perspectives," *Decision Support Systems*, vol. 40, no. 2, pp. 143 – 165, 2005.

[143] A. Barr, "Familiarity and trust: An experimental investigation," Centre for the Study of African Economies, University of Oxford, CSAE Working Paper Series 1999-23, 1999. [Online]. Available: https://ideas.repec.org/p/csa/wpaper/1999-23.html

[144] J. Zhang and A. A. Ghorbani, "Familiarity and trust: Measuring familiarity with a web site," in *In Proceedings of the 2nd Annual Conference on Privacy, Trust and Security (PST '04)*, 2004, pp. 23–28.

[145] M. C. Kaptein, C. Nass, and P. Markopoulos, "Powerful and consistent analysis of likert-type ratingscales," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10).* New York, NY, USA: ACM, 2010, pp. 2391–2394.

[146] H. Li and M. Singhal, "Trust management in distributed systems," *Computer*, vol. 40, no. 2, pp. 45–53, Feb. 2007.

[147] Transportation Security Administration. (2014, July) If you see something, say something. [Online]. Available: https://www.tsa.gov/news/top-stories/2014/07/28/if-you-see-something-say-somethingTM

[148] S. Yakoubov, V. Gadepally, N. Schear, E. Shen, and A. Yerukhimovich, "A survey of cryptographic approaches to securing big-data analytics in the cloud," in *Proceedings of IEEE High Performance Extreme Computing Conference (HPEC '14)*, 2014, pp. 1–6.

[149] P. Seymer and D. Wijesekera, "In-flight aircraft smart space security using multi-entity trust evaluations," in *Proceedings of the 37th IEEE/AIAA Digital Avionics Systems Conference (DASC'18)*, September 2018, pp. 1–10.

[150] Radius Networks. [Online]. Available: https://store.radiusnetworks.com/collections/all/products/radbeacon-dot

Curriculum Vitae

Paul Seymer is a PhD candidate in Computer Science at George Mason University's Fairfax, VA campus. He obtained his Master's degree (MS) in Information Security and Assurance at George Mason University in 2012. He also holds a Master's degree (MS) in Computer Science from Hood College, received in 2007, and a Bachelor's degree in Computer Science from the University of Maryland, at College Park, received in 2004.

Paul also holds a full-time position of Lead Infosec Engineer/Scientist and Group Leader with the MITRE Corporation in their Defensive Operations department, Cyber Operations and Effects division, CyberSecurity Technical Center, and has worked there since 2010. Prior to this position, he spent 10 years as a Senior System's Administrator at The Washington Post, after holding a position of Network Engineer for 2.5 years with Cannon Architects and Engineers.

His research interests include Intelligent Transportation Systems, Localization and Networking with Bluetooth Technologies, Intrusion Detection, and Anonymous Network Communication.