Smart Parking Zones Using Meshed Bluetooth Sensor Networks

Paul Seymer(✉), Duminda Wijesekera, and Cing-Dao Kan

George Mason University, Fairfax, VA 22030, USA
{pseymer,dwijesek,cdkan}@gmu.edu

Abstract. Developing seamless smart parking solutions remain an active research area. User coordination burden remains high, and systems based on complex instrumentation are often expensive and costly to maintain. Contemporary parking lot management systems are slow to adopt new technology, particularly in cases where there is no perceived immediate return on investment. We propose the use of a low cost, low power, Bluetooth Low Energy (BLE) based outdoor localization system coupled with a Random Forest classifier and dual-mode Bluetooth mesh network to provide space occupancy detection and sensor data transfer. To balance computational demands with cost, we leverage *fog computing* paradigms to shift computational capability near the sensor network where it can be used without an expensive network back-haul to a data center or cloud. We provide operational experiment results and analysis, and a study on the effects on accuracy from various network complexity and radio map minimization schemes.

Keywords: Smart parking · RF localization · Bluetooth networking

1 Introduction

Contemporary *smart parking* solutions are plagued with enduring problems. First is the excessive cost, due to the need for one sensor per parking space and retrofitting of lots with networking and power. This is a significant problem for large lots and those in remote locations without existing networking support or power. Second is a continued usability concern caused by traffic bottlenecks at ingress and egress payment support points. Those that do not suffer from these problems often use *crowd-sourced* occupancy detection features or rely on smartphone apps that offload the coordination burden directly onto the user and require a complicated technology back-end.

Solutions to these problems require a low cost and low power wireless solution that provides seamless occupancy tracking without the need for significant or expensive lot alterations. To reduce deployment complexity, the solution should cover multiple parking spaces per sensor and provide some degree of vehicle detection and tracking to prevent the need for user-provided payment support.

In past work [32] we presented our zone-based space occupancy and vehicle detection solution over our Bluetooth-based wireless mesh network along with

M. Helfert et al. (Eds.): SMARTGREENS 2019/VEHITS 2019, CCIS 1217, pp. 245–269, 2021.
https://doi.org/10.1007/978-3-030-68028-2_12

experiments and results from the deployment of our system at a parking lot used by the Center for Collision Safety and Analysis (CCSA) on the George Mason University's Fairfax, VA campus. In this work we present an extension of that paper, including an updated prediction model, an ablation analysis detailing the effects of a reduced sensor network and radio fingerprinting schemes on prediction accuracy and mesh network density (Sect. 4) so we may reduce network size and site deployment burdens without making our solution unusable, and an expanded related work section that includes a more direct comparison with our work (Sect. 5). When necessary, we simplified prior work explanations as well as provided additional clarity to tables and figures (citing where appropriate).

2 Solution Overview

We show an overview of our solution in Fig. 1. Our Bluetooth sensor network is deployed to a parking lot and collects Received Signal Strength Indicator (RSSI) measurements of custom Bluetooth Low Energy (BLE) beacons deployed inside parked vehicles. Prior to use, a radio map of RSSI values observed at each sensor node is created for each space in the lot, and used to create a Random Forest machine learning model (each space is a different class in the model). When powered on, our nodes create a self-forming authenticated mesh network to transport this RSSI data back to a central *sink* node.

To balance power hungry and network heavy computation and the use of low-power sensor nodes, we leverage *fog computing* concepts by locating the computational capability necessary to perform occupancy prediction near the sensor network. We divide this computation into two groupings based off of memory, power, and computational demand, where heavy one-time operations such as model training occur in a *cloud* environment and data collection and use of the model occurs in a *fog* network. We simulate our cloud environment, and establish the *fog* network in the CCSA center at the edge of the sensor network. This allows us to offload expensive computation and deploy lower power and lower cost sensor nodes. We depict this arrangement in Fig. 2.

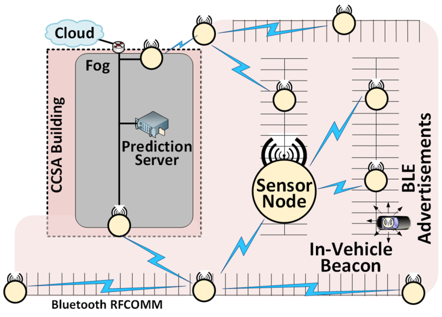

Fig. 1. Solution overview.

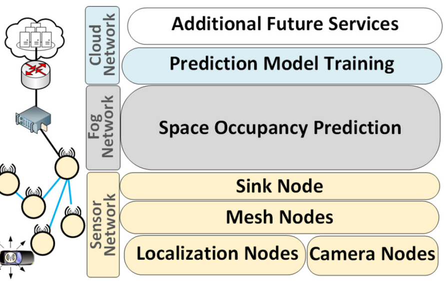

Fig. 2. Cloud/Fog/Sensor computation.

In addition to occupancy detection, we deploy a BLE-based vehicle ingress and egress detection capability to the entrance to the lot, coupled with an object recognition camera (for comparison).

2.1 Parking Space Occupancy Detection

Our current space occupancy detection system and Bluetooth mesh network is an evolution of prior work [31,32]. This section presents that prior work.

Zone Based Occupancy Detection. Preliminary experiments in prior work had a per-space occupancy detection goal [31]. In that work, training accuracy was above 90% however testing results were significantly less accurate. Several factors influence this outcome, from 2.4 GHz interference from nearby WiFi access points (the CCSA lot has upwards of 35 such access points from a nearby residential community) to vehicle size and orientation differences. To remedy this situation, we shifted from a per-space detection goal to a per-zone one. By creating *zones*, or contiguous parking space areas, we can mitigate small prediction errors due to a difference of a small number of spaces. We constructed zones as shown in Fig. 3. We believe that such a solution remains viable, as parking lot owners may only be interested in the area a vehicle is parked in within a lot, rather than an individual space. We outline the experiments that guided this decision and created an improved prediction model in Sect. 3 (Fig. 4).

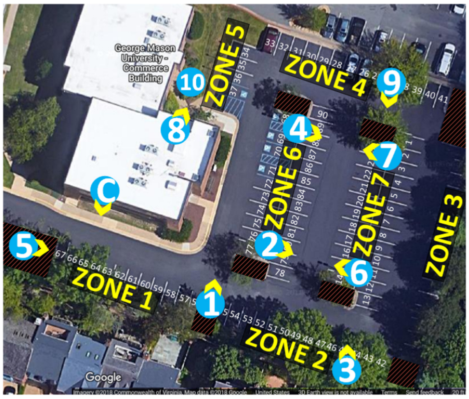

Fig. 3. Parking space zone map [32].

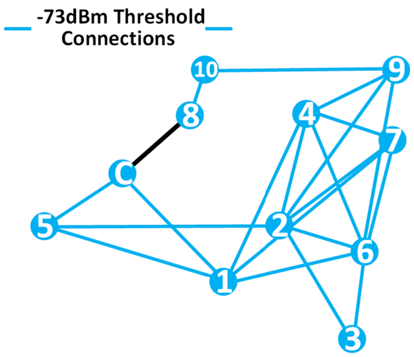

Fig. 4. Initial mesh connectivity.

Sensor Node Placement. Mounting locations for sensor nodes must not prevent occupancy of existing spaces nor interfere with vehicle movement. For our target lot, we utilized existing lamp posts, trees, and building window space to provide coverage for all spaces (see Fig. 3). Nodes were mounted at least 8 ft in the air to encourage line-of-sight with a maximum number of spaces. Experiments in prior work [31] gave us some insight into network performance and localization accuracy, causing us to include a redundant communication path from the *sensor* network to the *fog* network inside the building (through nodes "C" and "8"), and add additional sensor nodes. Node 10 is a special mesh-only node that is part of the mesh network but does not observe BLE beacons. Additionally, in this work we performed an ablation analysis on our prediction model determining that several of these nodes could be safely removed without significantly decreasing occupancy prediction accuracy. We detail these outcomes in Sect. 4.

Radio Map Construction. A radio map was constructed of the lot using our in-vehicle beacon and the sensor deployment shown in Fig. 3. Data was collected for each space for a 5 min window. Such a process, however, may be prohibitively time consuming for larger lots. To remedy this, we examine the effects on our model of reducing the number of spaces fingerprinted in Sect. 4.

Radio Fingerprint Feature Selection. Initially, we constructed a model that used a set of features based on descriptive statistics (median, variance, etc.) of all observed RSSI values. After additional analysis and experimentation we concluded that such a feature set was vulnerable to interference, attenuation, and other issues outlined in Sect. 3.1. In many cases these influences are ephemeral, but have severe effects on consistency of values, adding a great deal of noise to our features. We replaced this large unstable featureset with a single, maximum RSSI value observed within a time window at each node. We detail experiments that support this decision in Sect. 3.

2.2 Fogged Bluetooth Mesh Network

Each of our sensor nodes has identical hardware (except for the camera node) and is assigned a specific role (mesh only, sink, localization sensor, camera, etc.) within a configuration file. This role dictates its function, and allows a change in function post-deployment without physically modifying nodes. We use Bluetooth in EDR mode to perform mesh communication so we can partially avoid overlapping frequencies used by our BLE advertising channel-based localization. As Bluetooth has trouble penetrating walls we avoid deploying multiple mesh-only nodes to enable *fog* network communication, and instead communicate over the building's existing wired Ethernet. One mesh node is designated as the *central* sink node, providing a gateway for sensor data to be sent to the prediction system on the *fog* network.

Initially, our mesh network was deployed using a managed flooding approach. After conducting experiments we discovered that the high degree of message duplicates and inter-node connections was compromising our node's ability to receive beacon broadcasts from our in-vehicle beacon. To remedy this, we developed a simple routing algorithm to reduce network links, and a measurement sampling configuration reducing the total number of messages transmitted. We detailed these changes and their effects in Sect. 3.2.

Algorithm 1. RSSI based authenticated meshnet formation [32].

```
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:     authmsg $\leftarrow$ construct_authentication_message
21:     Open RESTful HTTPS connection to fn
22:     POST authmsg
23:     authreply $\leftarrow$ HTTP reply from POST
24:     if authreply is valid then
25:         return True
26:     return False
27: procedure auth_to_ble_network(node_info bn)
28:     authmsg $\leftarrow$ construct_authentication_message
29:     Open RFCOMM connection to bn
30:     Send authmsg
31:     authreply $\leftarrow$ Receive from bn
32:     if authreply is valid then
33:         return True
34:     return False
```

Authenticated Link Formation. Network formation occurs differently on the *fog* and *sensor* networks due to the difference in existing protocols used on Bluetooth and Ethernet networks. For example, node discovery on our *fog* network uses a slimmed-down implementation of SSDP [31], while nodes are discovered on the sensor network using Bluetooth's Service Discovery Protocol (SDP). The *fog* network is flat, while the sensor network is formed outward from the central *sink* node. This allows new sensor nodes to join the network and have a path back to the central node to carry our authentication messages.

Mesh network construction is shown as pseudocode in Algorithm 1 [32]. When each nodes starts up, *join_network* (line 1) repeatedly executes until at least one viable network hop is found. Next-hop discovery for fog-connected nodes is shown in lines 2–8, and for Bluetooth-connected nodes in lines 9–18. Our fog discovery (line 3) is implemented using a simplified SSDP service detailed in [31], and authentication (lines 5–6, procedure outlined in lines 19–27) is performed over a RESTful API written in Python and Flask over HTTPS (line 8). Bluetooth discovery (lines 11–12) uses BLE advertisements over a previously used SDP protocol, to support our signal-strength based routing algorithm outlined in Sect. 3.2. Bluetooth authentication proceeds in line 13. If authentication is successful for at least one node (line 15), a best node is selected (line 17), message queues are initialized (line 18), and the newly joined node begins broadcasting its presence on its respective medium (fog/Ethernet, or Bluetooth mesh) in line 16. This allows for the mesh network to form out from the central *sink* node, as it is the only node to advertise itself on boot, while all other nodes wait for a broadcasting node to come in range. Experiments and analysis that lead us to this algorithm are found in Sect. 3 and prior work [32].

Encrypted Message Transfer. Prior to deployment, each node exchanges and stores AES and HMAC key material with the central node. Our mesh network supports two main types of messages: authentication messages and parking system messages (RSSI data, heartbeats, camera messages, etc). Each message occupies a single AES block (128 bits) composed of a 4-bit *message type* to aid in message processing implementation (0000 for heartbeats, 0001 for RSSI data, etc.), a 16-bit sending node identifier, and 108 bit message text payload. This entire block is encrypted with the AES key and signed with the HMAC key (we implement a SHA-256 HMAC). This block is then pre-pended with a 2 bit mode identifier and 16 bit recipient node identifier and sent out over the network, each to allow for message routing and receipt without every node attempting decryption on every message. This message format is shown in Fig. 5. Additional implementation details can be found in [32].

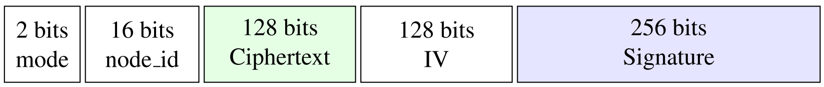

Fig. 5. 530 bit encrypted and signed message [32].

2.3 Vehicle Identification and Tracking

Our solution provides vehicle detection in two ways. First, is a pure wireless solution detailed in Sect. 2.3. For comparison, we include an object recognition camera-based solution in Sect. 2.3 and use it to perform attestations for our BLE beacon (i.e. a particular beacon does indeed belong to a vehicle).

Algorithm 2 . BLE Only Vehicle Identification Procedures [32].

```
 1: procedure initialize()
 2:     $parked\_records \leftarrow \{\}$                        ▷ init history data structure
 3:     $veh\_entered \leftarrow \{\}$                              ▷ init enter records
 4:     $veh\_exited \leftarrow \{\}$                                ▷ init exit records
 5:     detect_veh_kill_flag = False
 6:     beacon_measure_intvl = 5min
 7:     start thread detect_vehicle_enter()
 8:     start thread detect_vehicle_exit()
 9: procedure detect_veh_enter(each received beacon $b$)   ▷ called for each beacon
    central receives
10:     if $b.veh\_id$ not in $parked\_records$ then
11:         create entry for $b.veh\_id$ in $parked\_records$
12:         notify mgr new vehicle parked ($b.veh\_id$)
13:     $parked\_records[b.veh\_id].last\_seen \leftarrow b.time$
14: procedure detect_veh_exit()
15:     while detect_veh_kill_flag = False do
16:         $n \leftarrow time.now()$
17:         for each $veh\_rec \in parked\_records$ do
18:             $r \leftarrow parked\_records[b.veh\_id].last\_seen$
19:             if $(n - r) > beacon\_measure\_intvl$ then
20:                 notify mgr new vehicle exited ($b.veh\_id$)
21:         sleep (beacon_measure_intvl)
```

Algorithm 3 . Hybrid BLE/Camera Vehicle Identification Procedures [32].

```
 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:     Start thread to loop through calls to $attest\_veh\_thread(d)$
10: procedure attest_veh_thread(datastructure $d$)
11:     if $d$ has new event $e$ then
12:         if $e$ is a new Beacon event $b$ then
13:             Find matching Camera event $c$
14:         if $e$ is a new Camera event $c$ then
15:             Find matching Beacon event $b$
16:         if both $c$ and $b$ exist then
17:             Send attestation alert ($c$,$b$) to central node
```

Ingress and Egress Tracking. We leverage our localization solution to provide a means to track vehicles traveling into and out of the lot. When our nodes view a beacon that has not been observed in some time, we can assume this is a newly parked vehicle. Similarly, for beacons that are not seen for some time, we can assume the vehicle has left the lot. This forms a basic Bluetooth-only vehicle detection system, which we outline in Algorithm 2. The algorithm performs some setup in lines 8–12, including the data structures it will need to store beacon data. Two threads are launched (line 7–8) that loop through calls to procedures (line 8 and 13) that observe these data structures for changes (lines 9 and 18) in beacon timestamps, making notifications (lines 11 and 19) of vehicle detection when a threshold (initialized in line 6) has been exceeded.

Camera-Based Vehicle Detection and BLE Beacon Attestation. We also compared efficacy of our vehicle detection system to a low power, low cost object recognition camera by locating both at the lot entrance. We used a Jevois-A33 Smart Camera [17] connected via USB to one of our sensor nodes, and configured the camera to use one of it's pre-programmed recognition algorithms (Jevois Darknet YOLO module [16]) to determine if an object was a "car" or some other object. The sensor node was also equipped with our Bluetooth receiver and provided identical functionality to other sensor nodes. In addition to each solution being an independent way of detecting vehicles, we can combine them together so that the camera performs a level of attestation of the beacon it sees, to ensure that it indeed belongs to a vehicle. When a vehicle approaches the entrance to the lot, its beacons are detected by our sensor node. When the vehicle passes in front of the camera, the object recognition function is engaged and determines a "car" has passed in front of the camera. We use timestamps of these events to match them, and make an attestation. We provide an overview of this combined detection and attestation functionality in Algorithm 3. Here we start recording BLE beacon and camera detection events in lines 3–4. Thresholds that we experimentally determined are set in lines 5–6. A third thread is launched in line 9 that continuously runs an attestation procedure that matches new beacon events with new camera events based on timestamps, submits an alert back to the central node in line 17. We conducted feasibility experiments for our implementation of this algorithm in Sect. 3.3.

3 Experiments and Results

Our sensor nodes are constructed with Raspberry Pi 3s and after-market Bluetooth USB adapters [35]. All of our code is written in Python, using pybluez [18] libraries and the Bluez Linux Bluetooth stack [14]. We run a stripped down version of Ubuntu Mate on the Pi, however any Linux operating system compiled for the Pi 3 should perform well. We used Weka [12] and scikit-learn [4] to provide our machine learning libraries.

Table 1. *Offline* lot spaces [32].

Name	Source	Spaces	Rate
Tripod	Tripod	1-84, 89-90 89-90	Constant
350_o	350Z	set 2	Constant
370_o	370Z	1-90	Constant
TL_o	Acura	set 3	Constant

Table 2. *Over Mesh* lot spaces [32].

Name	Source	Spaces	Rate
350_m1	350Z	set 4	Constant
350_m2	350Z	set 1	Constant
350_m3	350Z	set 1	60s sample
370_m	370Z	set 1	60s sample
Rogue_m	Rogue	set 1	60s sample

During the course of our radio map creation and operational experiments, we assembled several training and testing datasets. Initially, we deployed our beacons to tripods in an attempt to create a radio map that was not biased toward a particular vehicle orientation, however we determined through testing of the effects of vehicle attenuation on the beacon's signal that training data produced from a beacon inside a vehicle produced a better model (see Sect. 3.1). As a result, we re-fingerprinted the lot after mounting a beacon behind the rear view mirror of a vehicle, and created several testing sets to explore accuracy when the model is applied to different vehicle shapes and when data is collected in an *offline* mode and over the live mesh network. We summarize each of these datasets in Tables 1 and 2 (sample rates explained in Sect. 3.2). In these tables, *set 1* spaces include 8, 20, 25, 27, 34, 36, 44, 53, 58, 64, 75, and 83. *set 2* spaces includes 25, 27, 29, 34, 36, 38, 39, 56, 58, 60, 62, 64, 67, 70, 71, 75–77, 79–81, and 87–89. *set 3* 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. *set 4* spaces includes 25, 27, 29, 34, 36, 56, 58, 60, 71, 72, and 75–76. Earlier work used the entire target parking lot [31], however at the time we conducted our new experiments, there were unmovable objects located in what is shown in Fig. 3 that prevented us from parking in those spaces. As a result, we kept the zone in the figure to prevent the need to renumber spaces, but have removed mention of it in our analysis.

3.1 Improved Prediction Model

We outline improvements to our Random Forest classifier for occupancy prediction in this section. We realized experimentally, that only maximum RSSI values produced consistent results. As a consequence, our first improvement was a reduction in feature-set size from 38 in prior work to 10 features per space, per time interval. In Table 3 we show a summary of results using this new feature-set trained against our tripod model (with both $n = 100$, and $n = 1500$ trees in forest), tested with TL_o, 350_o, and 370_o *in-vehicle* data sets. Training results, 10-fold cross-validated (CV) occupy the first row, while test data occupies the remaining rows.

While our training accuracy was 100% in the optimized tripod model, our success at predicting other vehicle occupancy on a per-space basis ranged 8.94% and 14.05% with the smaller model ($n = 100$) and decreased with the optimized

Table 3. Init. per-space tripod model.

Dataset	n = 100		n = 1500	
	TP	R	TP	R
Training	99.88%	1	100.0%	1
TL_o	11.23%	0.80	10.44%	0.87
350_o	8.94%	0.79	8.18%	0.83
370_o	14.05%	0.74	10.91%	0.82

Table 4. Init. zoned tripod model.

Dataset	n = 100		n = 1500	
	TP	R	TP	R
Training	99.68%	1	99.68%	1
TL_o	42.46%	0.83	38.95%	0.82
350_o	16.06%	0.71	18.48%	0.77
370_o	43.33%	0.81	43.93%	0.82

model to between 8.18% and 10.91%. Even after introducing additional sensors, we were forced to abandon the tripod model and adapt a zone-based approach (outlined in the next subsection) to obtain a viable solution.

Zone Based Occupancy Detection. To explore the effects of a move from per-space to zoned prediction prior to investing in re-fingerprinting the lot, we used the same tripod training data and divided the lot into zones as described in Sect. 2.1. We retrained our model using 6 classes instead of 90, (i.e. 6 zones assembled from all 90 spaces). Our *zoned* training and testing result is shown in Table 4. This was a significant improvement, however well below an acceptable level of accuracy. As a result, we replaced our tripod training set with one constructed from beacons located within a vehicle. We discuss this approach in the next subsection.

In-vehicle Effects on Beacon Attenuation. Upon taking a closer look at the individual measurements of our tripod model and comparing them to data collected from in-vehicle beacons, we saw that the location and orientation of the beacon has a significant impact on it's viability at producing our radio map. When mounted on a tripod, the beacon avoids any attenuation produced by the vehicle's chassis, seats, etc. In past work [31] we attempted to compensate for this by uniformly increasing the RSSI values for beacons, however additional analysis concluded that the attenuation is not consistent in every direction. For example, the rear-view mirror in our vehicles produced a 6 dBm RSSI decrease when measured from the same distance as a 1 dBm RSSI decrease due to a vehicle's front windshield. The errors created by this inconsistency were the limited factor in our model, as artificial compensations produced more problems than they solved. As the line-of-sight between the beacon and each node differs with each space the vehicle is parked in, we had to rebuild our radio map. We discuss our replacement map in the next subsection.

In-vehicle Fingerprinting. We re-fingerprinted the lot using the 370Z due to convenient availability of the vehicle. We then retrained our model and repeated the per-space and zoned tests that we performed for the tripod models. Results

Table 5. Per-space in-vehicle model.

Dataset	n = 100		n = 1500	
	TP	R	TP	R
Training	99.63%	1	99.56%	1
TL_o	42.11%	0.92	40.79%	0.95
350_o	12.78%	0.85	13.19%	0.89

Table 6. Zoned in-vehicle model.

Dataset	n = 100		n = 1500	
	TP	R	TP	R
Training	99.78%	1	99.67%	1
TL_o	89.65%	0.99	89.39%	0.99
350_o	85.28%	1.0	79.86%	1

are shown in Tables 5 and 6. Our trained model evaluated to accuracy above 99% for both per-space and zoned, however the zoned model produce a far superior result in for the smaller (n = 100) model of between 85.28% and 89.65% in our test vehicles. Additionally, the ROC areas increased for this zoned model to 99% and 100%. These tests were performed with data collection in *offline* mode, as they were collected and recovered from the node's directly. We explore the use of collecting data over our active mesh network in the next subsection.

3.2 Over-Mesh Experiments

Our first mesh design used a managed flooding algorithm with no route construction, which created a significantly dense and chatty network. In particular, node 1 (see Fig. 3) became an overloaded single point of failure as all mesh traffic was sent through it. As a result, we found that our Bluetooth radios spent a large amount of time communicating data instead of observing beacons. To remedy this, we deployed an additional sensor node (node 8) to allow for a redundant path, along with a simple route creation technique outlined earlier in Sect. 2.2 and sampling techniques outlined in Sect. 3.2. We discuss the experiments that lead to this change in the following subsections.

Effects of Over-Mesh Data Collection. With our mesh network active and collecting node data at the central *sink* node, we repeated our testing experiments using the 370Z, 350Z, and Acura TL test vehicles. Results are found in Table 7, computed against our zoned model (and per-space model for comparison). We continued to see a large amount of message queuing due to the volume of messages produced by a single beacon, as several beacons broadcasts can be observed by each node, for each time slice. This also produced a delay in assembling all of the required node data to make a prediction at the *sink* node. Results were close to the *offline* results in the last subsection, however there was a notable decrease in accuracy which due to this overload of messages. In some cases, no beacons were observed from nodes that produced data when in *offline* mode.

Effects of Down-Sampling. To improve accuracy and consistency we were forced to determine a means to reduce overall messages transmitted throughout

Table 7. Initial over-mesh results.

Model (Dataset)	TP	R
Zoned (350_m1)	46.33%	0.82
Zoned (350_m2)	61.94%	0.85
Per-Space (350_m1)	1.98%	0.60
Per-Space (350_m2)	9.72%	0.82

Table 8. Prediction using *Single Max*.

Model (Dataset)	TP	R
Zoned (350_m1)	66.67%	0.68
Zoned (350_m2)	75%	0.75
Per-Space (350_m1)	8.33%	0.85
Per-Space (350_m2)	8.33%	0.91

the network. To explore this, we first conducted an experiment where we reduced our mesh-collected test datasets to a single maximum RSSI measurement for each 5 min observation widow. This would be the most extreme reduction of message traffic that would remain within our experimental parameters. Results for prediction using these values is shown in Table 8, which showed an improvement in prediction allowing us confidence in investing in creating a data-sampling scheme at the nodes, and re-collecting new sampled test data.

To further explore sampling prior to making code changes, we took our training set and produced downsampled training sets to determine if an accurate model could still be constructed from a sample per 30 s, 60 s, 120 s, and 300 s (maximum). We show these results in Table 9. We see diminishing returns after a 60 s sample rate, so we coded that into our meshnet and use it moving forward.

Table 9. Down-sampled per-space results [32].

Sample rate	TP	R	Sample rate	TP	R
30 s	99.78%	1	60 s	99.78%	1
120 s	98.33%	1	300 s	97.78%	1

Finalize Design: Sampled and Routed Mesh Results. Before we created our downsampling scheme we attempted to mitigate message overloads with artificial delays so that messages could queue up in nodes and *burst* across to other nodes when connections were made. Our experiments showed, however, that this did not solve the problem. Once we implemented our downsampling and routed configurations, we repeated this measurement for comparison in Fig. 6. We see the routed mesh results are consistently and significantly below the message delays of the initial flooded mesh configuration, and the total messages seen during our experiments were several orders of magnitude less in our new scheme. During this experiment, we set a minimum delay for messages to 100 s to reduce risk of compounding messages and allow for fewer connection establishments when message counts are slow, however in later iterations this delay was removed to speed up availability of occupancy prediction results at the *sink* node.

We then repeated our testing with the 370Z, and added the Nissan Rogue to add diversity in vehicle size and orientation. We were unable to repeat the experiment with the 350Z, however we used downsampled results from the prior over-mesh experiments (350_m2) to form a new dataset for that vehicle (350_m3). These results are listed in the first three lines of Table 10. We also discovered during our experiments that there was a slight increase in accuracy when we used a per-space model, but performed post-processing on the prediction results to match the predicted space to it's corresponding zone. We refer to that in our results as *post zoned* results (lines 4–6 of Table 10). Lastly, we performed an additional post processing step where we determine the final prediction value of the vehicle to be the zone that occurred more than 50% of the time, referred to as *Majority*. We show results from this in the last three lines of the table.

Table 10. 60 s sampled mesh results [32].

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 60 s Post Zoned (350_m3)	95.83%	–
370 60 s Post Zoned (370_m)	85.41%	–
370 60 s Post Zoned (Rogue_m)	75%	–
370 60 s Post Zoned Majority (350_m3)	100%	–
370 60 s Post Zoned Majority (370_m)	83.33%	–
370 60 s Post Zoned Majority (Rogue_m)	75%	–

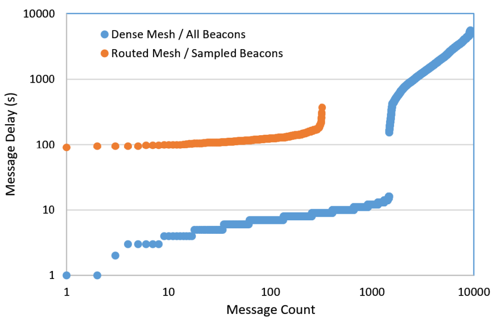

Fig. 6. Message delay comparison [32].

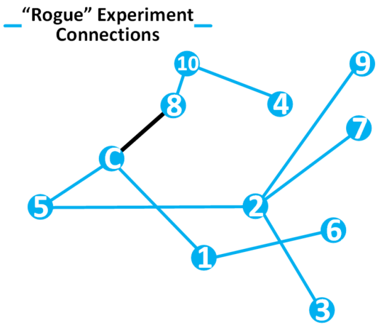

Fig. 7. Routed mesh connections.

3.3 Camera vs. BLE Based Vehicle Entrance/Exit Detection

We compared our BLE-only vehicle ingress/egress detection with outputs from an object recognition camera. To accomplish this, we mounted the camera onto one of our BLE nodes and aimed at the ingress/egress location in our target lot from a window in the near-by building (Fig. 8). We then drove a vehicle into and out of the lot 10 times and compared the outputs from both our node and the object recognition camera. When we analyzed the RSSI values from our node, we determined a consistent threshold for vehicle proximity was a value above −70 dBm. We then counted the number of beacons observed above this threshold. The object recognition camera labeled the vehicle as "car", and timestamps were recovered for each instance where this recognition was made. These outputs are shown in Table 11, along with a range of delays between then the BLE node detected the beacon and when the camera produced a detection output. We see that the camera had a delay of several seconds between the time a beacon was first observed and a recognition event was processed. This is due to the lack of a significant computational demand when detecting the presence of our BLE beacon.

Table 11. Daytime camera recognition vs BLE detection ($\leq$ 70 dBm.) [32]

Vehicle	Observed	Recog.	Detection
Direction	Beacons	Objects	Δt
Enter (10)	2–3	1–2 "car"	0–3 s
Exit (10)	1–4	1–3 "car"	2–11 s

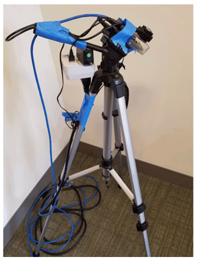
Fig. 8. Camera/BLE node rig.

We performed this experiment during daylight hours, and repeated another 10 passes at night. Our solution functions regardless of light levels, while the camera failed to detect any object in darkness. Additionally, for each ingress pass at night, we parked the vehicle in a different parking space within the lot. For spaces that were far from the ingress beacon, there was a clear gap between entrance and exit (Fig. 9) signifying when the vehicle passed by the beacon on its way to a far parking space. For spaces that were near the beacon, there was a much less pronounced difference (Fig. 10), indicating that our BLE-only solution may perform poorly when vehicles park in spaces near the ingress point of the lot. We conclude that a preferred solution leverage all nodes in a lot to perform detection, rather than a single node near the lot's entrance. We will explore this in future work.

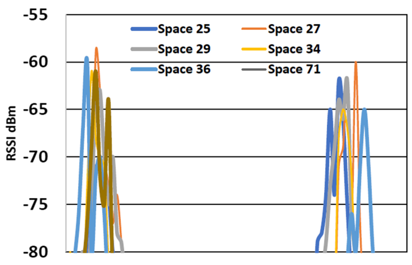

Fig. 9. Far space detection [32].

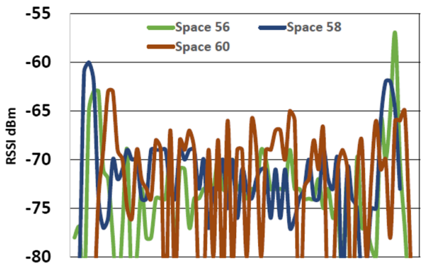

Fig. 10. Near space detection [32].

4 Prediction Model Ablation Study

For very large parking lots, fingerprinting each space may be a prohibitive burden. Furthermore, for zone-based detection, it may not be necessary. To explore the effect of removing sensor nodes (i.e. deploying a smaller sensor network) from the model and removing fingerprinting data (i.e. skipping parking spaces during fingerprinting) from the radio map has on prediction accuracy we conducted an ablation study. This technique exposes certain behaviors of our model for our target environment by removing data from our model piece-by-piece and studying the effects on prediction accuracy. Our goals with this exercise are to study effects and produce a minimum sensor network size (number of nodes) and smallest radio map (fewest spaces fingerprinted).

Each set of experiments begins with the default model from prior work [32], however we found that a small but consistent increase in accuracy was achieved by setting the algorithm to construct a model with random tie-breakers between attributes that are of equal value, and tuned our new model to enable this option.

Effects of Node Reduction. Recall from Sect. 2 that our prediction model is formed by combining maximum RSSI values from each node in the sensor network. For our 10 node deployment, the model is created from 10 values per instance (one for each node). We perform our node reduction experiments by computing attribute importance for each of these values, using the average impurity decrease method implemented in Weka, and remove the least important node. We chose this model as it is model dependent, and not specific to our observations about the physical layout of the sensor network.

Once a node is removed, we retrain our model with the remaining values and compare its accuracy. Each retraining cycle also computes attribute importance to produce a new least important feature, followed by removal of the next least important node, and so on until only a single node is left. This produces 10 accuracy measurements, one for the original model and 9 corresponding to each new model.

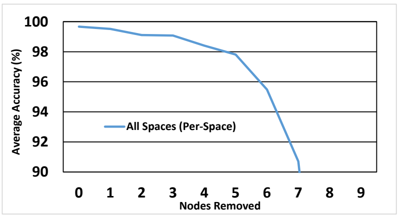

Fig. 11. Effects per-space fingerprinting. (Color figure online)

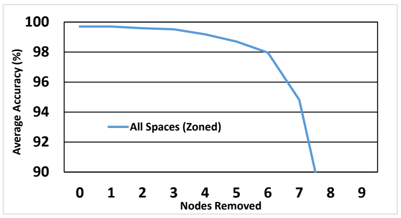

Fig. 12. Effects zoned fingerprinting. (Color figure online)

We performed this activity for both our per-space model and our zoned model, and show the output in blue in Figs. 11 and 12 (respectively). Here we see the original radio map, with node reduction as explained above, as a blue curve. For both per-space and zoned models, we were able to remove 7 nodes (of 10 total) before accuracy dropped below 90%. Additionally, our zoned model remained about 98% after 6 nodes removed, while the per-space model behaved similarly only up to 4 nodes removed. This makes intuitive sense, as our zoned model effectively masks some prediction errors found in the per-space model, when incorrectly classified nodes produce a prediction that remains in the correct zone. While encouraging, accuracy here is computed only across training data.

To obtain a minimal sensor network size with practical accuracy we must test the models produced with these reduced network size using test data used to validate our original models in Sect. 3.2 (i.e. differing vehicles, sampled and collected over mesh). We tested these reduced node models using the over-mesh 350 and Rogue datasets, and show results in Figs. 13 and 14. The left figure shows how our new model behaves with node reduction, showing consistent results even with removal of 2 to 4 sensor nodes. The right table shows the same result with the *majority* post processing we used in prior work, described in Sect. 3.2. This post processing had a marginal improvement in prediction accuracy, particularly for the *rogue* test set, however in three cases (2 in the *350* set, 1 on the *rogue* set) where this post-processing technique decreased accuracy. These three cases were limited to cases when 6 and 9 nodes were removed, where successful predictions were in the minority of other incorrect predictions.

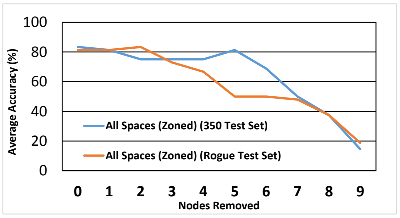

Fig. 13. Node reduction testing results.

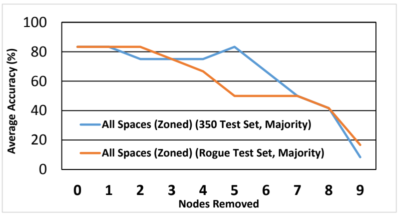

Fig. 14. Effects zoned fingerprinting.

When we examine the training and testing results together, we can see that the zoned model maintains performance (e.g. no more than a 10% accuracy loss) when up to 3 nodes were removed. These nodes correspond to nodes 1, 5 and 9. When we examine the network formed by our improved mesh network in Fig. 7, we notice that nodes 1 and 5 are links in the mesh network, and cannot be removed without requiring the network to reform. While we did not perform a network formation experiment with this new arrangement, we know from past data that replacement links between node 4 and 7, and 2 and 6 could have been established instead. We depict this reduced network in Fig. 15.

Effects of Radio Map Reduction. This section includes our analysis of the effects of reducing our radio map size. To remain independent of *a priori* facts about the physical layout of the parking lot or how our nodes behave (after fingerprinting the entire lot), we select spaces to remove from the radio map by simply keeping every 2nd space, and every 4th space. This produces two new radio maps that are half, and one-fourth the size of the original map. Should a balance be achieved between accuracy and space fingerprinting investment, new (perhaps larger) lots can be fingerprinted with less effort.

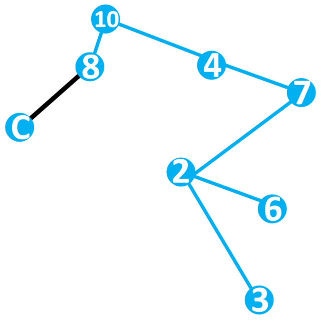

Fig. 15. New *node reduced* sensor network.

We trained two new prediction models, corresponding to each of our planned radio maps: full size (original), 1/2 size, and 1/4 size, and again used our *350* and *rogue* sampled datasets from Sect. 3.2. We show results in Table 12, where we see almost no significant influence on the training set's accuracy but a 20.8333% and 14.5833% decrease in accuracy for *350* and *rogue* testing sets (respectively) with the 1/2 radio map and a 16.6666% and 27.0833% decrease for the 1/4 radio map. Curiously, for the *350* test set, the accuracy counter-intuitively increased with a smaller radio map. We will explore this occurrence in future work. Additionally in the table, we show the outcomes from the "majority" post-processing technique we used in Sect. 3.2 when combining multiple prediction instances. In these cases, accuracy remained the same or was decreased due to rounding error in cases where correct and incorrect predictions totaled the same number (a case we consider a majority "incorrect").

Table 12. Effects of radio map reduction on full-lot accuracy.

Dataset	Full size			1/2 size			1/4 size		
	Acc.	Major.	R	Acc.	Major.	R	Acc.	Major.	R
Training	99.7037	100	1	99.7037	100	1	99.7101	100	1
350_m3	83.3333	83.3333	0.896	62.5	58.3333	0.878	66.6667	66.6667	0.850
Rogue_m	81.25	83.3333	0.899	66.6667	66.6667	0.896	54.1667	50	0.904

Outcomes from these experiments show that decreasing the radio map has a potentially prohibitive effect on prediction accuracy. In these cases, we would need to either combine zones together to further reduce error, or determine a way to perform interpolation on the missing spaces. We believe the latter is a feasibly approach and will explore this in future work.

Balancing Sensor Network and Radio Map Size. When combining a reduced radio map with a reduced sensor network, we are faced with a planning challenge. In the last two subsections, we explored effects of each, however when combining the two techniques together we must be careful not to compound losses in accuracy. We combined our node reduced models together with a reduced radio map, and repeated our training and testing exercises. We show the results in Figs. 16 for training data and Fig. 17 for testing data. Our "majority" post processing technique did not increase accuracy significantly, so we will favor clarity and avoid including those results in this discussion. Additionally our testing data was matched with a zoned model only, as per-space prediction was not possible for a reduced map size (i.e. many of the spaces that correspond to the true location were removed from the map to support the experiment).

Our training data, as expected (per prior discussion), produced a model that evaluated to accuracy above 98% regardless of radio map size up to and including removal of 6 (of 10 total) sensor nodes. Also as expected, the zoned model out performed the per-space model, however accuracy was maintained above 99% up to and including 3 removed nodes. When this process was repeated for testing data in Fig. 17 we see a less pronounced decrease in accuracy across the range of 7 removed nodes. In this case, however, we do see a rank ordering of accuracy with respect to changes in radio map size that favored a larger map. The less fingerprinted spaces, the higher the potential that any test measurement may predict out to the wrong zone. This was expected, however the interest

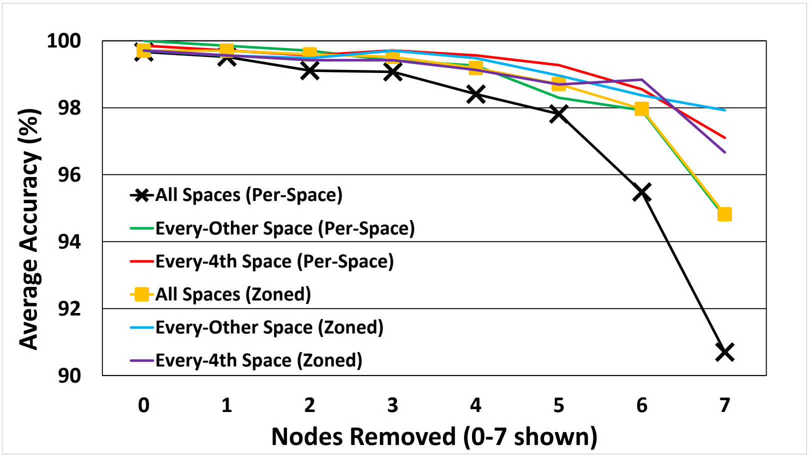

Fig. 16. Combined effects on training data.

here concerns the notion of acceptable losses in accuracy when compared to the deployment time savings of a reduced map and smaller sensor network. We see for both the 350Z and Rogue test sets, reducing the radio map size and a single node improved accuracy. In the case of the rogue dataset, this was maintained for 3 more removed nodes. We suspect that this was due to a specific coincidence where the spaces tested have some values that were creating mis-classification due to less than optimal data in one of the nodes. After examining the model, we noticed that Node 5 was the offending node in this case, and the incorrectly classified spaces were consistently in Zone 1, which happens to be the zone closest to Node 5. This is a bit counter-intuitive as our nodes should be less vulnerable to interference and other attenuation factors with shorter distances from beacon to observing node. In this case, however, a combination of factors are in play. For example, the Rogue is a larger vehicle, with a different internal orientation to the training vehicle, potentially appearing farther away from Node 5's perspective. This part of the lot has fewer observing nodes as well, increasing the effect of Node 5's errors. We seek to explore this topic in future work, and remedy the mis-classification effects from scenarios such as this.

From the data we have assembled, we observe that decreasing radio map size has a significant impact on detection accuracy, while reducing some nodes using the methods we employed can be done with less significant impacts. The *node reduced* sensor network depicted previously in Fig. 15 will still be viable, but at this time must be combined with a full size radio map if accuracy is to be maintained. We will continue to explore this area to improve both our prediction methods, and our node and map downsizing techniques.

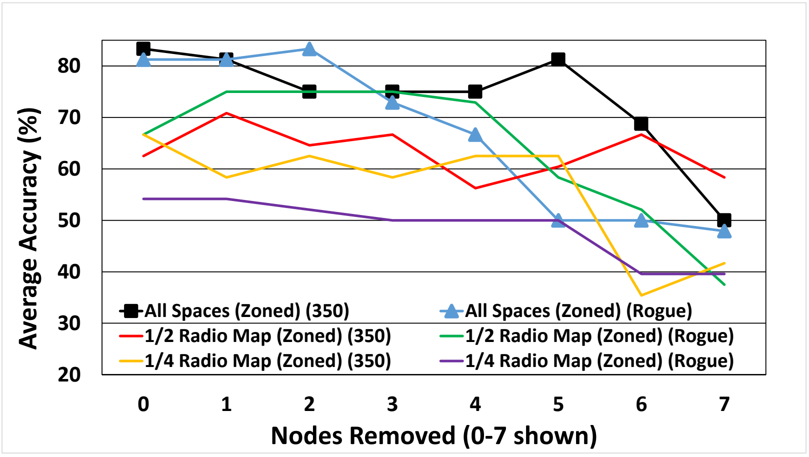

Fig. 17. Combined effects on testing data.

5 Related Work

5.1 Parking Management

Most paid parking lots (and even some free lots) have some degree of parking management deployed, either to collect information on space occupancy or support billing or payment transactions. These often come in the form of a human attendant that handles all aspects of driver and vehicle interaction, or an attendant-less lot with automated ingress and egress ticketing systems.

Crowd-Sourced Space Availability. Many commercially managed lots and street parking owners employ so-called *crowd-sourced* occupancy detection capabilities (ParkMobile [24]) where users of the system indicate when and where they have parked through a smartphone app or near-by electronic kiosk so that fees can be transacted electronically. Similar systems [28,29] often include a means to reserve spaces in advance of parking and often forgo the use of ingress and egress ticketing to speed up this interaction.

There are two main problems, however, with these approaches. First, they rely entirely on the accuracy and trustworthiness of the vehicle operator to correctly indicate where they have parked. Second, they place the entire coordination burden on the end user, substituting the time it takes to get into and out of the lot with time the user must spend indicating where they have parked. Our solution replaces all of these parking system characteristics by removing the ticketing stations entirely, as well as any requirement for the vehicle driver to interact with our system when parking. This represents the optimal *seamless* parking experience.

5.2 Indoor and Outdoor Localization

Outdoor localization is thought to be a solved problem with the deployment of GPS, however fine grained precision of static locations with low cost and low-power hardware continues to face challenges. Contemporary estimates of smartphone GPS accuracy under ideal weather and line-of-sight conditions, for example, is estimated to be just under 5 m [26]. Some research [13] has been performed combining GPS data with historic *crowd-sourced* space occupancy information to provide localization and predict space availability. Systems like these may have a large data storage burden when forced to maintain large amounts of past parking data. There is a larger concern, however, with systems that rely on parked-vehicle provided GPS data: The parking management system is trusting localization data provided to it by the entity that parks. This empowers user-exploitation, particularly when large fees are involved. Parking management systems must perform localization (or verification) from an infrastructure-controlled system, where sensing local to the parking space performs confirmation of vehicle location. Our system does precisely this, however there is a small possibility that the in-vehicle beacons could be manipulated in a way such that malicious activity could focus on directing a signal in such a way at our sensor nodes, that the prediction model would fail or produce incorrect results. We believe, however, that this would be very time-consuming for an adversary and is not something we feel is necessary to defend against at this time.

Indoor localization systems were created to provide location services indoors, where GPS cannot reach. Some solutions on vehicle dead reckoning such as work by Gao et al. that leverages inertial sensors within an in-vehicle smartphone to detect vehicle movement [10]. Liniger [22] combines GPS data with BLE beacons and a vehicle's state obtained from its On-Board Diagnostic (OBD-2) connection to determine is a vehicle is parked or moving. Others use physical detection of occupancy through use of infrared sensors aimed at spaces [38]. Many wireless localization solutions, in contrast, use signal strength measures (often RSSI measures) to perform position prediction using a variety of radio technologies (which can also be used in outdoor environments). For example, Oguejioforo et al. [27] uses RSSI measurements of IEEE 802.15.4 (low-rate wireless Personal Area Networks) radios combined with linear distance estimates, while Fabian [7] provides localization in their parking management solution that uses trilateration. Two problems exist with use of euclidean distance measurements. First, to make use of the estimates, the geography of the environment they are used in must be measured so that some indicated distance from a receiver can be interpreted as a particular location. Second, single sensor measurements are often error prone, and combining several distances to form an accurate single determination of location would compound these errors. An alternative technique is to employ the use of radio maps where sensing remains static but measurements (so called "fingerprints") at predetermined locations are taken and combined together. This modeling can be done and data can be manipulated in many different ways. For example, Faragher and Harle perform develop several localization techniques that compare the use of BLE and WiFi radios [8]. Silver [33] combines various measurement value filtering techniques and compares accuracy of *disc trilateration* and k-nearest neighbor fingerprinting techniques. Daniay et al. [5] leverages RSSI fingerprinting of moving BLE beacons to feed k-nearest neighbor and Neural Network algorithms. Additional works are surveyed in [23].

5.3 Bluetooth and Mesh Networking

Our initial mesh deployment mimicked the Bluetooth SIG BLE mesh specification [3] and other Bluetooth controlled flooding algorithms [19], however such a design quickly became unusable for our purposes as we needed to minimize our use of BLE advertising channels as they were the foundation of our localization solution. This drove us to use Bluetooth *classic* (EDR) in an attempt to minimize the impact on these channels.

Mesh networks are too large of a topic for this section, however we can group mesh network research into two main groups, those that use message flooding [1,15,30] and those that establish fixed or dynamic routes [11,21,25,34]. In contrast, flooding approaches are generally easier to implement and are often broadcast based but produce a lot of duplicate messages while routed approaches are (sometimes) more complicated to implement and often produce some degree of network overhead due to the exchange of routing information between network nodes. Our solution avoids some of this message overhead by producing routes prior to the use of the network, based on a boot-time RSSI measurement of nearby nodes. This prevents the need for network overhead, and remains viable since our sensor network never moves (e.g. the RSSI measurements are network construction will remain the same). Additional routed mesh protocols can be found in [37], and Bluetooth-specific mesh protocols surveyed in [6,36].

5.4 Random Forest Classifiers

Machine learning techniques are used to find patterns and solve real world problems from malware detection [9] to detecting oils spills in satellite imagery [20]. Any well understood problem that requires a solution involving the determination of patterns within data that are not clearly accessible by manual examination are prime candidates for machine learning algorithms. Random forest classifiers are a type of machine learning algorithm referred to as *ensemble* techniques as they combine multiple models together to form more accurate detection (many other examples of use are surveyed in [2]). Classifiers of this type are easy to interpret and tune, which was out primary driver in considering their use (outside of experimental validation conducted in past work [31].

6 Conclusions

In this work we present an evolution of out Bluetooth based outdoor smart-parking localization solution. We perform a series of validation experiments to produce a prediction model that balances efficacy with simplicity of deployment. With *zone-based* smart parking solutions like ours, outdoor lot owners can leverage new technologies to make parking management more efficient without requiring significant investments in technology.

Acknowledgement. © 2019 The MITRE Corporation. ALL RIGHTS RESERVED. The author's 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 19-2570. Distribution Unlimited.

References

1. Baert, M., Rossey, J., Shahid, A., Hoebeke, J.: The bluetooth mesh standard: an overview and experimental evaluation. Sensors **18**(8), 2409 (2018)
2. Belgiu, M., Dragut, L.: Random forest in remote sensing: a review of applications and future directions. ISPRS J. Photogram. Remote Sens. **114**, 24–31 (2016)
3. Bluetooth SIG Mesh Working Group: Bluetooth mesh profile specification, July 2017
4. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
5. Daniay, F.S., Cemgil, A.T.: Model-based localization and tracking using bluetooth low-energy beacons. Sensors **17**(11), 2484 (2017)
6. Darroudi, S.M., Gomez, C.: Bluetooth low energy mesh networks: a survey. Sensors **17**(7), 1467 (2017)
7. Fabian, H.: A public parking management system for Zurich. Master's thesis, University of Zurich (2015)
8. Faragher, R., Harle, R.: 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, pp. 201–210, September 2014
9. Firdausi, I., Erwin, A., Nugroho, A.S., et al.: Analysis of machine learning techniques used in behavior-based malware detection. In: 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, pp. 201–203. IEEE (2010)
10. Gao, R., Zhao, M., Ye, T., Ye, F., Wang, Y., Luo, G.: Smartphone-based real time vehicle tracking in indoor parking structures. IEEE Trans. Mob. Comput. **16**(7), 2023–2036 (2017)
11. Guo, Z., Harris, I.G., Tsaur, L., Chen, X.: An on-demand scatternet formation and multi-hop routing protocol for ble-based wireless sensor networks. In: 2015 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1590–1595, March 2015. https://doi.org/10.1109/WCNC.2015.7127705
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
13. Hobi, L.: The impact of real-time information sources on crowd-sourced parking availability prediction. Master's thesis, University of Zurich (2015)
14. Holtmann, M., Hedberg, J.: Bluez - official linux bluetooth protocol stack (2018). http://www.bluez.org/

15. Hortelano, D., Olivares, T., Ruiz, M.C., Garrido-Hidalgo, C., López, V.: From sensor networks to internet of things. Bluetooth low energy, a standard for this evolution. Sensors **17**(2), 372 (2017). https://doi.org/10.3390/s17020372. https://www.mdpi.com/1424-8220/17/2/372
16. Itti, L.: Darknet yolo jevois module (2018). http://jevois.org/moddoc/DarknetYOLO/modinfo.html
17. JeVois Inc: Jevois-a33 smart camera (2018). https://www.jevoisinc.com/pages/hardware
18. karulis: Pybluez - python extension module allowing access to system bluetooth resources (2018). https://github.com/pybluez
19. Kim, H., Lee, J., Jang, J.W.: Blemesh: a wireless mesh network protocol for bluetooth low energy devices. In: 2015 3rd International Conference on Future Internet of Things and Cloud, pp. 558–563, August 2015
20. Leifer, I., et al.: State of the art satellite and airborne marine oil spill remote sensing: application to the BP deepwater horizon oil spill. Remote Sens. Environ. **124**, 185–209 (2012)
21. León, J., Dueñas, A., Iano, Y., Makluf, C.A., Kemper, G.: A bluetooth low energy mesh network auto-configuring proactive source routing protocol. In: 2017 IEEE International Conference on Consumer Electronics (ICCE), pp. 348–349. IEEE (2017)
22. Liniger, S.: Parking prediction techniques in an IoT environment. Master's thesis, University of Zurich (2015)
23. Liu, H., Darabi, H., Banerjee, P., Liu, J.: Survey of wireless indoor positioning techniques and systems. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **37**(6), 1067–1080 (2007)
24. LLC, P.: Parkmobile (2018). https://parkmobile.io/
25. Mikhaylov, K., Tervonen, J.: Multihop data transfer service for bluetooth low energy. In: 2013 13th International Conference on ITS Telecommunications (ITST), pp. 319–324. IEEE (2013)
26. National Coordination Office for Space-Based Positioning, Navigation, and Timing: GPS accuracy. https://www.gps.gov/systems/gps/performance/accuracy/
27. Oguejioforo, S., Okorogu, V., Abe, A., Osuesub, O.: Outdoor localization system using RSSI measurement of wireless sensor network. Int. J. Innov. Technol. Exploring Eng. **2**, 1–6 (2013)
28. Parking Panda. https://www.parkingpanda.com/
29. Passport Labs, INC.: Passportparking. https://www.passportinc.com/
30. Reddy, Y.K., et al.: A connection oriented mesh network for mobile devices using bluetooth low energy. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 453–454. ACM (2015)
31. Seymer, P., Wijesekera, D., Kan, C.D.: Secure outdoor smart parking using dual mode bluetooth mesh networks. In: Proceedings of the 89th IEEE Vehicular Technology Conference (VTC 2019-Spring), April 2019
32. Seymer, P., Wijesekera, D., Kan, C.D.: Smart parking zones using dual mode routed bluetooth fogged meshes. In: Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems, May 2019
33. Silver, O.: An indoor localization system based on BLE mesh network. Master's thesis, Linkoping University (2016)
34. Sirur, S., et al.: A mesh network for mobile devices using bluetooth low energy. In: 2015 IEEE SENSORS, pp. 1–4, November 2015. https://doi.org/10.1109/ICSENS.2015.7370451

35. StarTech: Mini usb bluetooth 4.0 adapter - 50m (165ft) class 1 edr wireless dongle (2018). https://www.startech.com/Networking-IO/Bluetooth-Telecom/USB-Bluetooth-4-Dongle~USBBT1EDR4
36. Todtenberg, N., Kraemer, R.: A survey on bluetooth multi-hop networks. Ad Hoc Netw. **93**, 101922 (2019)
37. Waharte, S., Boutaba, R., Iraqi, Y., Ishibashi, B.: Routing protocols in wireless mesh networks: challenges and design considerations. Multimedia Tools Appl. **29**(3), 285–303 (2006)
38. Yee, H.C., Rahayu, Y.: Monitoring parking space availability via zigbee technology. Int. J. Future Comput. Commun. **3**(6), 377 (2014)