



Fakultät für Elektrotechnik und Informationstechnik  
Lehrstuhl für Kommunikationsnetze  
Prof. Dr.-Ing. Wolfgang Kellerer

## Final Report - Vacant Parking Space Detector

**Octavio Rodríguez,  
Jure Zdovc**

WSN Final Report

Semester: SoSe 2014  
Group: 04  
Authors: Octavio Rodríguez  
          Jure Zdovc  
Advisors: Prof. Dr.-Ing. Wolfgang Kellerer  
             M.Sc. Arsany Basta  
Date: July 13, 2014



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Project Description</b>	<b>2</b>
2.1	Requirements . . . . .	2
2.2	Topology . . . . .	2
2.3	Routing . . . . .	3
2.3.1	Rumor Routing . . . . .	3
2.3.2	Dijkstra's algorithm for best path selection . . . . .	4
2.3.3	The rumor-Dijkstra hybrid . . . . .	5
<b>3</b>	<b>Implementation Details</b>	<b>7</b>
3.0.4	Network Discovery . . . . .	7
3.0.5	Parking Status Propagation . . . . .	7
3.0.6	Packet Format . . . . .	7
3.0.7	Hardware used . . . . .	7
3.0.8	Graphical User Interface . . . . .	7
<b>4</b>	<b>Summary</b>	<b>11</b>
<b>A</b>	<b>List of Figures</b>	<b>12</b>
<b>B</b>	<b>Bibliography</b>	<b>13</b>

# 1 Introduction

An important part of city planning involves estimating the amount of traffic a certain street or city district will support. Cars going to a certain destination will ultimately have to find a parking space. This means that obtaining information on parking space usage could greatly contribute to improvement of urban spaces and regulate parking pricing.

For instance, a study conducted between 1927 and 2001 [1] in several cities around the world showed that between 8 and 75% of the traffic on the streets in the study areas were looking for parking. The average time spent cruising for a parking space in urban public parking areas varies from 3.5 to 12 minutes, with distance traveled ranging from 1 km to over 1.5 km. When multiplied by the number of cars looking for a parking space, this translates into a considerable amount traffic, wasted fuel, and unnecessary air pollution.

A parking space monitoring system could help alleviate such problems. However, current vacant parking space systems are almost exclusively used in multi-story car parks. The reason behind it is that the typical sensor-based parking space detection system are costly and require a considerable amount of time and effort for installation. Intrusive sensors (e.g. pneumatic tubes, weight-in-motion sensors, piezoelectric cables) need to be embedded in floor and non-intrusive sensors are wired to the ceiling or walls (e.g. ultrasonic sensors, CCTV-based detection).

In this project we want to demonstrate a cost-effective alternative to such systems with a prototype using interconnected wireless sensors. By using Crossbow MicaZ motes, the sensors could be used in parking lots without roof, or even on the streets for curb parking slots. The statistics are then fed to a web server that could be accessed, for example, by the parking lot operator or by smart devices on cars. This means that such a system could provide benefits for both parties:

Benefits for operators	Benefits for drivers
Near-real-time parking lot occupancy information. More efficient and safer traffic flow. Parking usage trend analysis. Improved space utilization. Valuable statistics.	Guided assistance in finding a free space. Reduced carbon dioxide emissions. Lower fuel consumption. Cruising time savings. Less frustration.

Table 1.1: Examples of benefits for a cost-effective Vacant Parking Space Detector system.

## 2 Project Description

### 2.1 Requirements

In order fulfil the basic needs of any parking detection system, our proposed solution will feature the following:

- Low power consumption – The system will focus on event-driven updates in order to conserve as much energy as possible and prolong time between sensor battery replacements.
- Ease of installation – With a wireless sensor network approach, this system will avoid cabling installation. Also, dynamic routing makes it possible to install sensors in a plug-and-play fashion.
- Reliability – The routing protocol focuses on reaching the control center keeping several alternate paths ready in case of connectivity loss.
- Error detection – By keeping track of each device's power level on every message, sensors running low on battery can be quickly identified and get a replacement before losing connectivity.
- Fast reaction times – The system will notify parking status changes on time for the next car looking for a parking spot.

### 2.2 Topology

The two scenarios where this solution could be applied would be mainly in parking lots, and multi-story car parks, but also curb parking spaces. In order to communicate the status of the parking spaces, wireless motes are programmed with one of three roles (see figure 2.1):

1. Sensor - Mote running on batteries equipped with a light sensor and encased in a protective shell that is fixed to the floor or pavement. It is responsible for monitoring parking space occupancy and inform its immediate gateway of a status change.
2. Gateway - Mote connected to a power source in an elevated position (e.g. a street lamp). It provides network access to the sensors by forwarding their status messages towards the control center. It also builds a distance-based routing table that consists of neighbor gateway addresses and their distance to the control center.

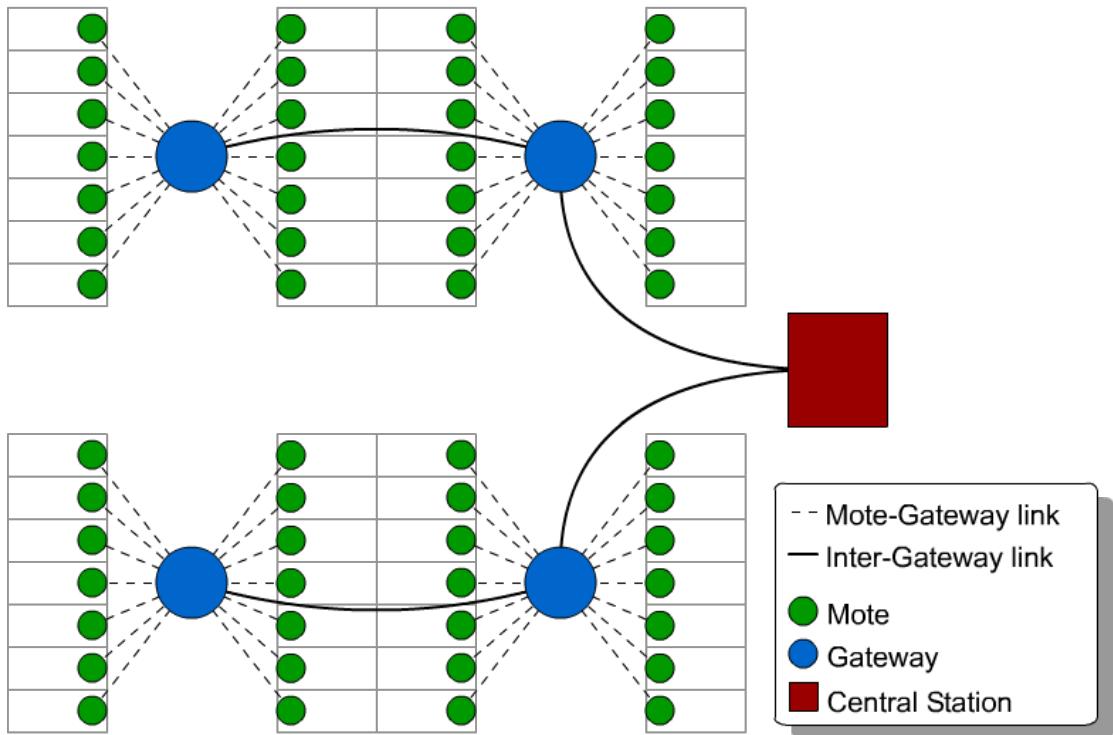


Figure 2.1: Example topology of a small parking lot

3. Central Station (CS) - Mote connected to a computer or web server. This device will receive and process all status messages from every gateway. It also provides statistical information and GUI for the parking lot operator and information to be printed in an outdoor display for the drivers.

The complete space detection and status forwarding is illustrated in figure 2.2.

## 2.3 Routing

The routing protocol proposed is dynamic and meant to ease installation in a plug-and-play fashion. The objectives of our implementation is to develop a simple, yet effective and fast route selection. It should provide quick reaction times and backup path switch-over in case of gateway saturation or connectivity loss. Lastly, it should allow new sensors or gateways to be added to the network with as little configuration effort as possible, if any. We decided a protocol mix of Dijkstra and rumor routing could meet all these requirements.

### 2.3.1 Rumor Routing

In the original concept of query-based rumour routing [2], each node has a neighbor list and an events table. The events table contains forwarding information to all the events it knows. After a

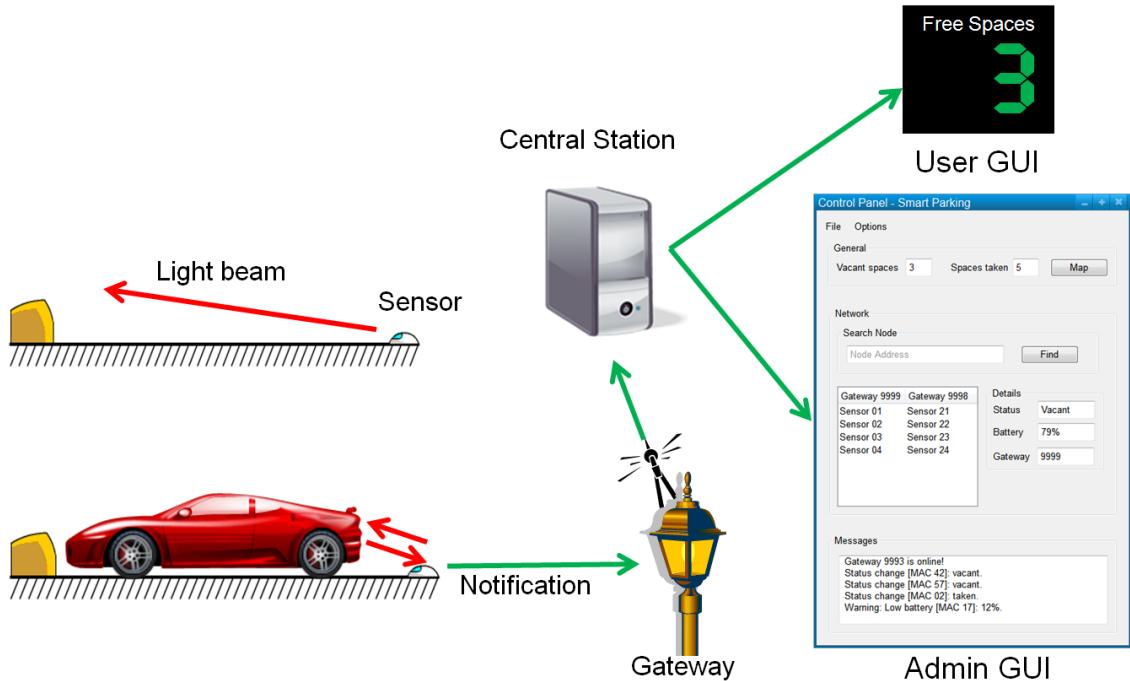


Figure 2.2: System setup: Sensor motes are fixed onto the ground and detect if a car is using the parking space. If the parking status changes, the sensor notifies its gateway. The gateway injects the notification into the network where it reaches the central station. The central station publishes the information via an Admin GUI or a public display on the street.

node witnesses an event, it sends a special long-lived packet to the network known as an agent. The agent travels around the network hopping from node to node creating a path originating from the event.

Once there are events, agents and paths on the network, any node may send a query packet for a particular event into the network. If the query packet reaches a node who knows the route to the event (as told by a passing agent), it will forward the query. Otherwise, the query will be sent and propagated in a random direction until the query reaches a node with a route to the event.

#### Properties

- Power saving, improves network longevity.
- Robust in dealing node failure.
- Tunable, allow trade-off between setup overhead and delivery reliability.

#### Assumptions

- No coordinate system is available so location information is not known by nodes.
- Static topology.

### 2.3.2 Dijkstra's algorithm for best path selection

This algorithm seeks the best path from a source node to a destination node in a given network. It starts by assigning a tentative distance value to every node: set to zero for the initial node and

to infinity for all other nodes. Marks all nodes unvisited. Sets the initial node as current. Creates a set of the unvisited nodes called the unvisited set consisting of all the nodes.

Once all these initial settings are done, an iterative process begins in which each node is evaluated for the shortest distance:

1. For the current node, all of its unvisited neighbors are evaluated and their tentative distances are calculated.
2. It then compares the newly calculated tentative distance to the current assigned value and assign the smaller one.
3. When all of neighbors of the current node has been evaluated, the current node is marked as visited and removed from the unvisited set.
4. If the destination node has been marked visited or if the smallest tentative distance among the nodes in the unvisited set is infinity, the algorithm stops.
5. The next step is to take the unvisited node with the smallest tentative distance as the new "current node" and go back to the first step.

#### Properties

- Clever way to use cost/distance to select a path.
- Focused in looking for one path only (i.e. the algorithm must be run on every device).
- Looking for the shortest path by flooding consumes a lot of energy.
- Flooding increases the chance of collision.

#### 2.3.3 The rumor-Dijkstra hybrid

As we have seen, rumor routing and Dijkstra's algorithm have attractive properties that could provide advantageous to our purposes. In order to provide a practical, simple and fast solution we had to mix the most convenient features to our topology :

- After booting, a new node will send an UPDATE broadcast announcing its MAC address and its distance to the central station (255, the equivalent to infinity).
- Every gateway that receives an UPDATE and is currently available, must reply with an ACK, its MAC address and its distance to the central station.
- The new node will wait five seconds to receive all acknowledgements. After that, the device reads each ACK and selects the five best metrics.
- In the case of sensors, the metric is signal strength (the higher, the better) and for gateways is distance to the central station (the lower, the better).
- After a gateway reads the distance of an ACK, it also checks the signal strength of the packet and adds a weight depending on the RSSI value (+1 if  $> 0$ , +2 if  $> -15$ , +3 if  $\leq -15$ ).

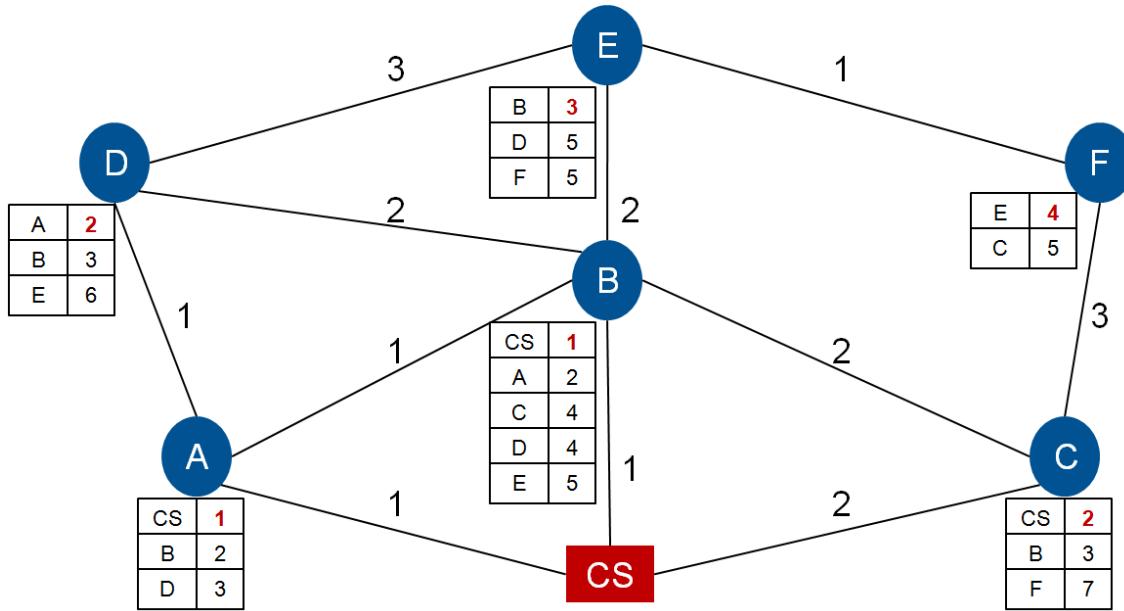


Figure 2.3: Example of best path discovery. The number in the lines connecting the nodes are the weights taken from the RSSI values of each link (1 if  $> 0$ , 2 if  $> -15$ , 3 if  $\leq -15$ ). The blue circles represent gateways and the red square is the central station (CS). Each node keeps a table with five of its neighbors with the shortest distance to the CS. The red number is the distance each node reports in UPDATE packets.

- The replies with the five best metrics are put in the routing table. In the case of gateways, the best distance (+ RSSI weight) in the table becomes its own distance. the device can begin with the tasks specific to its role (gateway or sensor).
- If there was no response at all, the device will repeat the process over and over again sending UPDATE packets until it receives an ACK.

In contrast to the original rumor routing concept, the hybrid does not consider events as information source or an agent traversing the network. Instead, it makes emphasis on routing by exchanging information with neighbors only. Their only interest is about their distance to the central station (or signal strength).

Also, instead of running a Dijkstra algorithm to check every single node for their link cost and update it after every calculation, we took the concept of cost and a step-wise evaluation of the best path to a specific node. This process happens locally on every node and this information will be exchanged with neighbors only.

Queries are sent only on the network discovery stage, right after the device booted up. Alternatively, UPDATE packets are triggered if a device had to delete all of the entries in its routing table and in the case the best distance in a gateway's table changes (and its distance with that). This way the gateway informs its neighbors and each of them updates its own table.

## **3 Implementation Details**

**3.0.4 Network Discovery**

**3.0.5 Parking Status Propagation**

**3.0.6 Packet Format**

**3.0.7 Hardware used**

**3.0.8 Graphical User Interface**

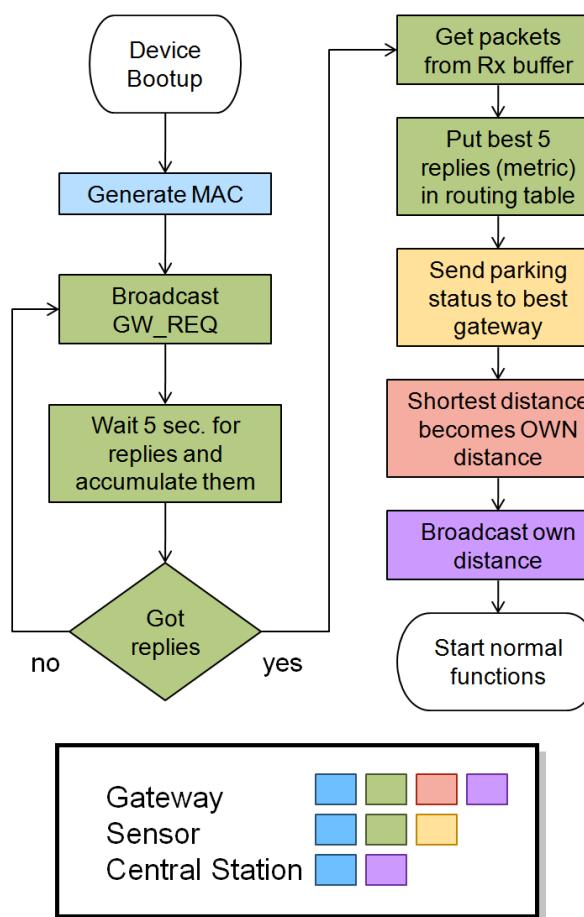
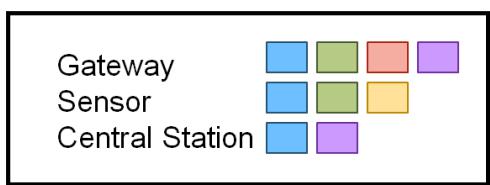


Figure 3.1: Network discovery process for the three roles.



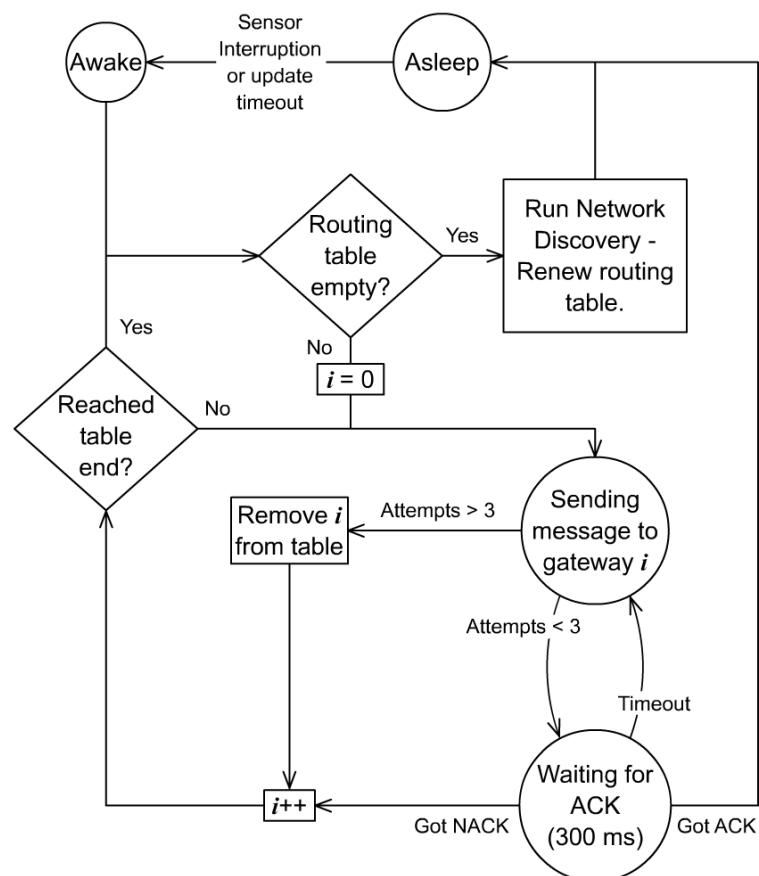


Figure 3.2: Flowchart of the functions of a sensor.

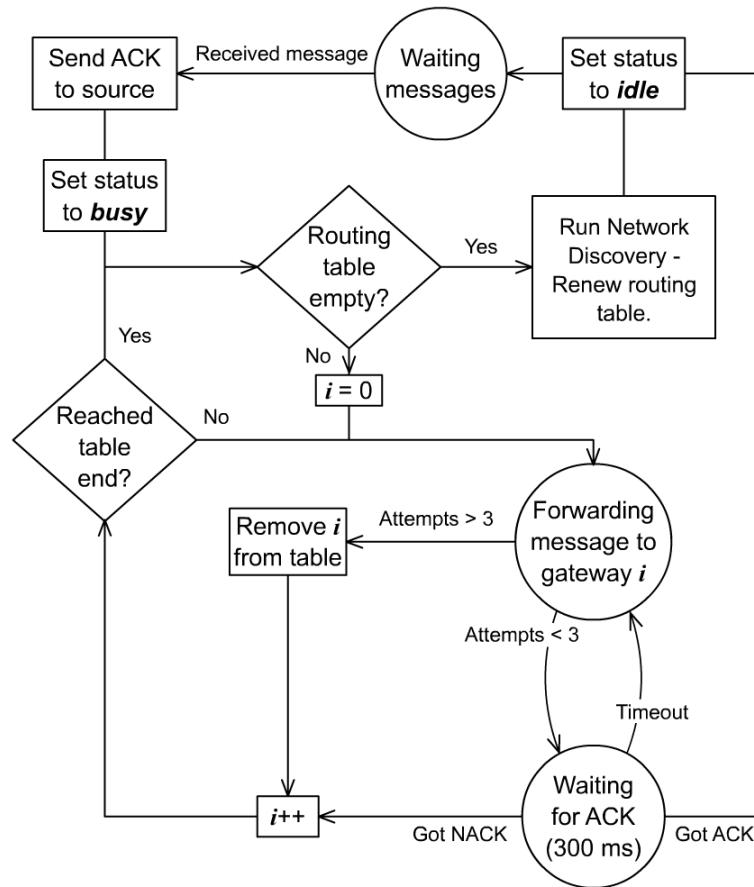


Figure 3.3: Flowchart of the functions of a gateway.

	SRC	ID	CMD	Value
ACK	Gateway Address*	Packet ID Number	ACK	Busy status
Update			UPDATE	Distance
Occupied			OCCUP	Source sensor addr.
Empty			EMPTY	Source sensor addr.
GW Request			GW_REQ	**unused

Figure 3.4: Packet Format. \* In the case of occupied and empty, the SRC field originally contains the sensor's address. \*\* The Packet is fixed length, so the unused value is arbitrarily assigned.

## **4 Summary**

## A List of Figures

2.1	Example topology of a small parking lot . . . . .	3
2.2	System setup and vacant space detection . . . . .	4
2.3	Best path discovery example . . . . .	6
3.1	Network discovery process for the three roles. . . . .	8
3.2	Flowchart of the functions of a sensor. . . . .	9
3.3	Flowchart of the functions of a gateway. . . . .	10
3.4	Packet format . . . . .	10

## B Bibliography

- [1] Donald Shoup. Cruising for parking. *Access Magazine*, 2007.
- [2] David Braginsky and Deborah Estrin. Rumor routing algorithm for sensor networks. In *First Workshop on Sensor Networks and Applications (WSNA)*, pages 1–12, 2002.
- [3] D.B.L. Bong, K.C. Ting, and K.C. Lai. Integrated approach in the design of car park occupancy information system (COINS). *IAENG International Journal of Computer Science*, 2008.
- [4] Amin Kianpisheh, Norlia Mustaffa, Pakapan Limtrairut, and Pantea Keikhosrokiani. Smart parking system (SPS) architecture using ultrasonic detector. *International Journal of Software Engineering and Its Applications*, 2012.