Development of IoT Based Smart Parking System: A Solution to Unbalanced Parking Demands

Branch - CSE

Semester - 7th

Abstract

Smart-parking system helps users automatically and a free parking space at the least cost based on new performance metrics to calculate the user parking cost by considering the distance and the total number of free places in each car park. This cost will be used to offer a solution of anding an available parking space upon a request by the user and a solution of suggesting a new car park if the current car park is full. The simulation results show that the algorithm helps improve the probability of successful parking and minimizes the user waiting time. We also successfully implemented the proposed system in the real world.

0.1 INTRODUCTION

In the development of traffic management systems, an intelligent parking system was created to reduce the cost of hiring people and for optimal use of resources for car-park owners. Currently, the common method of finding a parking space is manual where the driver usually finds a space in the street through luck and experience. This process takes time and effort and may lead to the worst case of failing to find any parking space if the driver is driving in a city with high vehicle density. The alternative is to find a predefined car park with high capacity. However, this is not an optimal solution because the car park could usually be far away from the user destination. In recent years,research has used vehicle-to-vehicle and vehicle-to infrastructure interaction with the support of various wireless network technologies such as radio frequency identification (RFID), Zigbee, wireless mess network and the Internet. Smart parking system aimed to provide information about nearby parking spaces for the driver and to make a reservation minutes earlier using supported devices such as smartphones or tablet PCs. Furthermore, the services use the ID of each vehicle in booking a parking space. However, the current intelligent parking system does not provide an overall optimal solution in finding an available parking space, does not solve the problem of load balancing, does not provide economic benefit, and does not plan for vehicle-refusal service.

To resolve the before forementioned problems and take advantage of the significant development in technology,

the Internet-of-Things technology (IoT) has created a revolution in many fields in life as well as in smart-parking system (SPS) technology . The present study proposes and develops an effective cloud-based SPS solution based on the Internet of Things. Smart parking based on IOT constructs each car park as an IoT network, and the data that include the vehicle GPS location, distance between car parking areas and number of free slots in car park areas will be transferred to the data center. The data center serves as a cloud server to calculate the costs of a parking request, and these costs are frequently updated and are accessible any time by the vehicles in the network. The SPS is based on several innovative technologies and can automatically monitor and manage car parks. Furthermore, in the proposed system, each car park can function independently as a traditional car park. It also implements a system prototype with wireless access in an open-source physical computing platform based on Arduino with RFID technology using a smartphone that provides the communication and user interface for both the control system and the vehicles to verify the feasibility of the proposed system.

0.1.1 OVERALL VIEW

First, this algorithm describes a mechanism to search car parks at the least cost. Secondly, this mechanism is used for forwarding the vehicles to another car park if the current car park is full. It is considered that a network of

car parks such that each park is a node in a network. Each node obtains the information from the neighboring node, thus ensuring smooth movement of vehicles at low cost and increasing the probability of finding a free parking space.This mechanism evaluated the performance of the system through simulation and implementation. The results of the simulation are close to the mathematical models and achieve better performance than the other systems, so that it reduces the number of vehicles failing to find a parking space and minimizes the costs of moving to the car park. The cost defined here is the time that the user must wait for the service, thus helping users save time and money and reducing environmental pollution.

0.2 ARCHITECTURAL VIEW

0.2.1 SYSTEM OVERVIEW

The SPS uses the WSN consisting of RFID technology to monitor car parks. An RFID reader counts the percentage of free parking spaces in each car park. The use of RFID facilitates implementation of a large-scale system at low cost. The system provides a mechanism to prevent disputes in the car park and helps minimize wasted time in looking for a parking space. After logging into the system, the user can choose a suitable parking space. Information on the selected parking location will be confirmed to the user via notification. Then, the system updates the status of the parking space to "pending" during which time the system will not allow other users to reserve it. If after a certain period of pending time the system determines that no car is parked in that space, then it changes the status to "available." The system will update the status from the WSN node (the status of car park spaces) when a new car joins in the system. Therefore, the status of the overall parking system is always updated in real time. The system will help plot the parking time for each parking space in real time and can support the business with hourly parking charges.

0.2.2 SYSTEM ARCHITECTURE

Elements in the system:

1. Cloud-Based Server: This is a Web entity that stores the resource information provided by local units located at each car park. The system allows a driver to

search and find information on parking spaces from each car park without the need to directly access the local server node by directly accessing the cloud-based server.

2.Local Unit: This unit is located in each car park and stores the information of each parking space, as shown in Fig. The local unit includes the following:

(i)Control Unit: This is an Arduino module, which is connected using an RFID reader. The card reader authenticates the user information and then displays this information on the screen. If the information of the RFID tag or card is correct, the Arduino module will control the opening of the door for the vehicle to enter. The Arduino module connects with the cloud server through an Internet connection to transfer data from the local car park to the cloud server database.

(ii) Screen: This displays information on the capacity of the local car park, the total current percentage of free spaces, the status of the RFID tag check, the user card when entering, and a mini map of the local car park.

(iii)RFID Tag or ID Card: This is used to check and authenticate user information and calculate the percentage of total free spaces in each car park.

3. Software Client: This is an application software system. Running on Android operating system, the users will install it on their smartphones and use it to reserve parking spaces. The users access the system via 3G/4G mobile connections.

0.2.3 NETWORK ARCHITECTURE

In general, the term "user" when referring to the driver or vehicle and the term "resources" when referring to the parking spaces.

1) Parking Network

It uses the car park network (CPN) architecture infrastructure/ backbone. The architecture , where the dashed lines indicate wireless link and the solid lines indicates wired link. This type of parking network includes routers that form as the infrastructure for connected clients. The CPN infrastructure/backbone can be built to allow sensor networks to connect using wireless radio technologies. The routers form a self-configuring and self-healing link network. Routers can be connected to the Internet by gateway functionality. This approach, also referred to as infrastructure meshing, provides the backbone for conventional clients and enables integration of CPNs with existing WSNs through gateway/bridge functionalities in the routers. Conventional clients with the same radio technologies as the routers can directly communicate with the routers. We have assumed that each car park is a node in a CPN. The deployment network in a real environment where each car park is labeled.

(a) P1 is car park number 1; N1 is the total parking spaces in P1.

(b)P2 is car park number 2, N2 is the total parking spaces in P2.

(c)Pn is car park number n, Nn is the total parking spaces in Pn. The total capacity of the system is

$$N = N1 + N2 + N3 + ... + Nn(spaces) \qquad (1)$$

. D is the real distance between two nodes in the network. Dij is the distance between nodes Pi and Pj. Each node has a neighbor table to maintain information on the current status of the network and a queue with predefined length. The neighbor table for each node contains information on the neighboring nodes directly linked to it. On the other hand, the queue is used to control the number of vehicles forwarded to the node, which aims to prevent overloading in the number of vehicles beyond the capacity of the node. In our proposed system, each node will broadcast a message to its neighboring nodes after a new node joins or leaves it. This message includes information on its total free resources. The neighboring node that receives this message will update its neighbor tables.

Let's if N1 = 100 spaces, N2 = 120 spaces, N3 = 200 spaces, N4 = 100 spaces, N5 = 120 spaces, N6 = 120 spaces, N7 = 100 spaces; D12 = 1.2 km, D13 = 1.6 km, D23 = 2 km, D27 = 1 km, D34 = 1.5 km, D37 = 1.8 km, D45 = 1.2 km, D56 = 0.8 km and D67 = 1.2 km. Let's assume that the total free spaces in N1 = 20, in N2 = 60, in N3 = 60, in N4 = 70, in N5 = 60, N6 = 30 and in N7 = 60. To increase the performance of finding a free parking resource, the neighbor table in each node contains information on the current number of free parking resources in the neighboring nodes. Our idea is to use the number of total free parking resources in each node to calculate the cost for choosing a car park.

2) Constucting the neighbour table of nodes

Consider a function named

$$f(\alpha, \beta) \tag{2}$$

to calculate the cost between the nodes in the network.

$$f(\alpha, \beta) \tag{3}$$

) is a function that depends on the distance between two nodes and the number of free parking spaces in the destination node.

$$f(\alpha, \beta) \tag{4}$$

is considered to be a weighted link between two nodes in the parking network. If two nodes are not directly linked, then

$$f(\alpha, \beta) \tag{5}$$

) D 1. If the vehicle comes into a node and that node is full, the vehicle will be forwarded to the next node, which is a neighbor of this node with the smallest value of

$$f(\alpha, \beta) \tag{6}$$

in the neighbor table.Calculate the cost function

$$f(\alpha, \beta) \tag{7}$$

from node Pi to node Pj, i.e.,

$$fij(\alpha, \beta) = \alpha * (dij/Dup) + \beta * (tj/Tup) \tag{8}$$

where

$$\alpha \tag{9}$$

is a coefficient that depends on the length of the path between two nodes and

$$\beta \tag{10}$$

is a coefficient that depends on the number of free slots in the destination node.

$$f(\alpha, \beta) \tag{11}$$

is inversely proportional to the distance between two nodes and directly proportional to the total free slots in the destination node. Depending on which parameter we consider to be the more important of the two parameters, i.e., the distance or the free slots, we can adjust

$$\alpha, \beta \tag{12}$$

to achieve better network performance.

$$\alpha, \beta \tag{13}$$

are parameters derived from the experiment, and their value is [0, 1]. If

$$\alpha = 0 \tag{14}$$

, only consider the number of free spaces to calculate the cost to the user. If

$$\beta = 0 \tag{15}$$

, only consider the distance between two nodes to calculate the cost to the use

Let us calculate the cost function based on the distance between two nodes and the percentage of free parking spaces at each node. We use the upper bound of the distance between two nodes and the upper bound of the capacity for parking in each car park. dij is the distance between nodes Pi and Pj, Dup is the upper bound of the distance and is a global parameter, tj is the number of spaces that are occupied at node Pj, and Tup is the upper

bound of the capacity of the overall parking network and is a global parameter. We assume a network with seven nodes and calculate the value of function F with

$$\alpha = 0.2 \tag{16}$$

$$\beta = 0.8 \tag{17}$$

D = 2 km, T = 200 spaces; F12 = 0.36; F13 = 0.72; F21 = 0.44; F23 = 0.76; F27 = 0.34; F31 = 0.48; F32 = 0.44; F34 = 0.27; F37 = 0.42; F43 = 0.71; F45 = 0.36; F54 = 0.24; F56 = 0.44; F65 = 0.32; F67 = 0.36; F72 = 0.34; F73 = 0.74; F76 = 0.48. The neighbor table of each node with the

$$f(\alpha, \beta) \tag{18}$$

shows that the new neighbour table for each node follows Eq. We will use this routing table in choosing the next node where to forward the user when a car park is full.

0.3 ALGORITHM AND MATHEMATICAL MODEL

0.3.1 ALGORITHM

1) SYSTEM OPERATIONS

When a user wants to find a parking slot, he must login to our system. After successful login, a request message is sent to search for a free parking slot. Then, the system will send back a response message containing the information, including the car park address and the directions to reach it. The choice of the car park is based on the function which is

$$f(\alpha, \beta) \tag{19}$$

calculated based on the current location of the vehicle and the location of the car park. The system will forward the vehicle to a car park with a minimum

$$f(\alpha, \beta) \tag{20}$$

value if the current car park is full. When the user arrives at the car park, he must be authorized to enter. This authorization is achieved via the RFID technology or by scanning the user card. This mechanism is simple but economical. If the information is correct, the user is allowed to park. If the current car park is full, the system will send a suggestion message that includes information on a new car park, including the address and new directions, with a minimum cost. The new car park will be selected based on the neighbor table of the current car park (the first node in the neighbor table).

(i)Reservation Process:

if the user is looking for a free parking space,he will send a request message to the system (1), which is done using a smartphone. When the system receives this request, it will find car park P1 with the least cost [minimum value of

$$f(\alpha, \beta) \tag{21}$$

] and forward this message to the user. In this case, the least cost is the minimum value of function

$$f(\alpha, \beta) \tag{22}$$

. The value of

$$f(\alpha, \beta) \tag{23}$$

is calculated as the distance (between the vehicle and car parks) and the number of free spaces in each car park. If this car park has free parking slots, it will send a response message to the user . The response message includes the address of car park P1 and its directions. Because we use the percentage of total free spaces in suggesting a new car park, a high probability of success exists in finding a free parking space.

In our proposed system, we use RFID technology to calculate the percentage of total free parking spaces in each car park. In each car park, an RFID reader is installed at the entrance. We use a variable named "Count" to calculate the total number of vehicles in the car park. Count D Count C 1 when a vehicle enters, and Count D Count 1 when a vehicle leaves. When Count D Ni, car park i is full. The process of updating the neighbor table is described as follows: when a change in the value in the Counter occurs, which changes the percentage of

the total free parking spaces at this node, this node will send a message containing updated information to the cloud-based server. The cloud-based server will update (ii) Entering Process:

if a user enters car park P1; he must be authorized using an ID or an RFID card . If authorized, the door is opened, and the count will increase by one. The system will send a response message to the user to notify successful parking . If the car park is currently full, it will send a response message suggesting an alternative car park, including relevant information on new car park P2, with the least cost.

2) CALCULATING THE TOTAL FREE PARKING SPACES AND UPDATING THE NEIGHBOR TABLE

In the system, we use RFID technology to calculate the percentage of total free parking spaces in each car park. In each car park, an RFID reader is installed at the entrance. We use a variable named "Count" to calculate the total number of vehicles in the car park. Count = Count + 1 when a vehicle enters, and Count = Count - 1 when a vehicle leaves. When Count = Ni, car park i is full. The process of updating the neighbor table is described as follows: when a change in the value in the Counter occurs, which changes the percentage of the total free parking spaces at this node, this node will send a message containing updated information to the cloud-based server. The cloud-based server will update the neighboring tables of its neighboring nodes.

0.3.2 MATHEMATICAL MODEL

Let us create a parking planning strategy. We let P denote the set of all vehicles with parking queries in the queue. We let S denote the total of all available car parks.We letWdenote the set of wij, where wij is the cost between vehicle pi (pi

$$\epsilon \tag{24}$$

P) and car park Sj (Sj

$$\epsilon \tag{25}$$

S). We can achieve W by calculating the distance from the vehicle to the car park (GPS address) and the number of free spaces in car park Sj. We let M and N be the size of P and S, respectively. Therefore, the size of W is M * N. By assuming that vehicles are jobs and parking places are servers, Wij is the cost for server Sj to do job Pi. We save the solution in X, where xij

$$\epsilon \tag{26}$$

X, i.e.,

xij = 1; if Pi will park at Sj 0; if Pi will not park at Sj

We let C be the total cost for all vehicles in P to go to the parking places assigned to them by the SPS, i.e.,

C=

$$\sum_{i=1}^{m} \sum_{j=1}^{n} wij * xij \tag{27}$$

In our study, we use

$$f(\alpha, \beta) \tag{28}$$

as the cost; thus, we have a new total cost. C=

$$\sum_{i=1}^{m}\sum_{j=1}^{n} fij(\alpha,\beta) * xij \tag{29}$$

To decrease the cost to the user, we will choose the minimum value of

$$f(\alpha,\beta) \tag{30}$$

in (30). We aim to make C minimum on the condition that each vehicle obtains exactly one parking resource and each car park space can be assigned to only one vehicle, i.e.,

C=

$$\sum_{j=1}^{n} xij <= 1 \tag{31}$$

indicates that any user in the queue may be assigned at most one car park but may also fail to get an assignment. On the other hand, xij = 1 still guarantees that each user in the queue maintains a car park assignment. In our proposed system, if a vehicle does not find a free parking space on arrival at a full car park, forwarding it to a different car park will be suggested. We let h denote the number of forwarded vehicles. Each car park can be assigned to kj vehicles (kj is the total free slot in Sj; C=

$$\sum_{i=1}^{m}\sum_{j=1}^{n} wij * xij \tag{32}$$

The time complexity of our algorithm is O(n*k). We will try to reduce the time cost for each vehicle in finding a free parking resource. Our mathematical model has reduced the total cost and reaches a better solution in distributing the users to the overall network resources.

0.3.3 QUEUE MODEL

We modeled the system into a service queue. It includes all users entering each car park. The entering process at each node is considered to be a first-in first out (FIFO) queue and a Markov process. The mathematical model for the entering process can be described as follows: M/M/1/K/FIFO. The first "M" denotes that the distribution of the arrival process is Markovian (Poisson distribution), the second "M" stands for the servicetime distribution, which is also Markovian (exponential distribution), "1" denotes the server, and K is the number of lots.

$$\mu A \tag{33}$$

is the inter-arrival time between two users, and

$$\mu S \tag{34}$$

is the service time (parking time). By assuming that

$$\mu S > \mu A \tag{35}$$

, the queue does not explode. The average waiting time in the queue (expected from a long time) in the M/M/1 queue is expressed as

Ta =

$$\mu S^2/\mu S - \mu A \tag{36}$$

The average waiting time in the queue (expected from a long run) in the M/M/k queue is expressed as

Ta =

$$\mu S^2/k(\mu S - \mu A) \tag{37}$$

The average waiting time of the system is

$$\bar{Ta} = \sum_{i=1}^{n} Tai/N \tag{38}$$

where N is the total number of parking spaces.

0.4 SIMULATION

0.4.1 SIMULATION

1) SETUP

To evaluate the performance of the processes, we simulated a network deployment, including the car park architecture mentioned above.We used the network simulation tool Arena to simulate this network. To simulate the mathematical and queuing models, we randomly created vehicles to join the network. The arrival process of the vehicles followed the Poisson distribution in our simulation, which was denoted as POIS(X), where X is the inter-arrival time between successive vehicles arriving at the car park. In this simulation X = 15 and 20 min. We considered the vehicle as the job and the parking space as the entity doing the job. The time for doing the job followed an exponential distribution, which was denoted as EXPO(Y) in this simulation, where Y is the average service time that a vehicle stays in the parking space. We chose Y = 60 min in this case. We simulated a parking network with five car parks as five nodes. We assumed that the network nodes are interconnected, as shown in Fig. 11. We set up each car park with a capacity for four parking spaces as the resources. We also created the same number of random vehicles to arrive

at each car park. In this simulation, the number of vehicles arriving at each car park is 60, 70, 80, 90, and 100. We ran the simulations until all vehicles were serviced. To compare the network performance and provide an optimal solution for our proposed network, we set up a simulation based on various values ofalpha and beta. We simulated all cases of alpha and beta in the range from zero to one. The following are some outstanding values of alpha and beta in the range from zero to one. The value of alpha changed from (0, 0.2, 0.5, 0.8, 1). The value of beta changed from (0, 0.2, 0.5, 0.8, 1). We set up the distance between the network nodes as follows: D12 = 0.6 km, D15 = 1.2 km, D23 = 2.4 km, D24 = 2.0 km, D34 = 1.8 km, and D45 = 1.6 km. We chose the value of the upper bound of the distance as Dup = 2.4 km and that of the upper bound of the capacity as Tup = four spaces.

2) ARENA SIMULATION TOOL

Arena is discrete event simulation and automation software developed by Systems Modeling and acquired by Rockwell Automation in 2000 . It uses the SIMAN processor and simulation language. Arena is good software that simulates many types of real-time systems such as parking systems. The basic building blocks of Arena models are modules. Flowcharts and data objects define the process to be simulated and are chosen from panels in the project bar. Flowchart modules describe the dynamic processes in the model. The types of flowchart modules available are Create, Dispose, Process, Decide, Batch, Separate, Assign, and Record. Other panels may contain many additional types of flowchart module. Data

modules define the characteristics of various process elements such as entities, resources, and queues. They can also set up variables and other types of numerical values and expressions that pertain to the whole model. In particular, in the parking system simulation, Arena supports many random distributions of arrival and service processes, such as Poisson, Normal, Exponential, Triangular, Uniform, Beta, Gamma, Logarithmic, and Weibull distributions. It allows statistical calculations and exports the average values of the parameters used to evaluate the performance, such as the average vehicle waiting time for parking requests and average time the vehicle stays in the parking system. Many previous researchers have shown that the simulation results in Arena are close to those in actual practice. Owing to the abovementioned advantages, we chose Arena as the simulation tool in this study.

3) SCENARIOS

As mentioned in Section IV-(1), we created a simulation network consisting of five nodes. To compare the performance To reduce thewaiting time of vehicles in the system, we use the second network model in planning to solve this problem. This network model . In this network model, when a vehicle arrives at a car park that is currently full, it will be forwarded to a different car park that has free parking spaces. The forwarding is based on the algorithms that we have proposed. We simulated two network models in the Arena simulator, and we compared the average waiting time and average total time that a vehicle resides in each node.

0.5 RESULT AND IMPLEMENTATION

0.6 CONCLUSION

A parking system that improves performance by reducing the number of users that fail to find a parking space and minimizes the costs of moving to the parking space. The proposed architecture and system has been successfully simulated and implemented in a real situation. The results show that our algorithm significantly reduces the average waiting time of users for parking. The results closely agree with those of our proposed mathematical models. The simulation of our system achieved the optimal solution when most of the vehicles successfully found a free parking space. The average waiting time of each car park for service becomes minimal, and the total time of each vehicle in each car park is reduced. In future studies, there is a possibilty of adding queues in the model as well as implement the proposed system in large scales in the real world.