ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

주차정보시스템에서 메시지 지원을 위한 선택통지 프레임 워크☆

Filtered-Push Notification Framework for Messaging Support in Parking Information System

로미오 마테오* 이 재 완**
Romeo Mark Mateo Jaewan Lee

요 약

중요한 통지를 받기 위해 주차관리에서 스마트 폰이 사용될 수 있다. 이러한 접근 방법의 내부구조에서는 통지 시스템에서 제공되는 메시지 지속성을 고려해야 한다. 그러나 브로드케스팅 방법은 클라이언트에게 통지할 때 많은 오버헤드가 발생한다. 본 논문은 불필요한 메시지 오버헤드를 줄이기 위해 필터링 방법을 사용한다. 휴리스틱 방법을 사용하여 통지 테그 값과 유사한 선호 값을 이용하여 이동 클라이언트를 선택한다. 시뮬레이션 결과 제안한 필터링 기법이 다시 연결하는 과정에서 더 빠르고, 전통적인 메시지 프레임 워크와 비교하여 메시지 오버헤드가 낮은 것으로 나타났다.

ABSTRACT

Smart phones can be used in parking management to receive important notifications. An infrastructure based from such approach should consider message persistence which is currently supported through push notification systems. However, the broadcast method produces high overhead when sending notifications throughout the clients. This paper uses a filtering method to prevent unnecessary message overheads. The heuristic approach selects the mobile clients with corresponding preference values based on the notification tag values. The simulation result showed that the proposed filtering method was faster in re-establishing connections and had lower message overhead compared to conventional messaging frameworks.

☞ keyword : Push notification(통지), wireless mobile messaging(무선 모바일 메시지), WSN-based parking management(WSN 기반 주차관리), parking guidance system(주차안내시스템)

1. Introduction

In a parking management system, providing users with fast and efficient service is a primary concern. Socioeconomic implications are also considered in imposing policies for choosing appropriate parking locations as well as maintaining parking areas [1] to prevent an underpriced and overcrowded facility. A parking information system is a well-known technique used by drivers to quickly find parking spaces and to inform them of critical events in the parking area such as collisions [2]. Commonly, a parking lot uses screen monitors to display a map of the parking lot and identify parking areas with empty slots [3, 4]. Markings or labels in each parking area are made visible for it to be easily identified by a driver. Others provide additional devices like light/led indicators to indicate the vacancy of slots. In [3], the system consists of a WSN-based vehicle detection and management system where it gathers information on the availability of each parking lot and guides the driver to the available parking slot by led monitor displays. However, these traditional approaches are costly to implement, and in the long run, devices within the parking lot will be hard to maintain.

In this paper, we present a messaging system to guide drivers in parking their cars and inform them of critical events in the parking area through their smart phone and at the same time using interactions from RFID and wireless sensors. We propose a parking guidance system using

* 정 회 원 : 군산대학교 전자정보공학부 박사과정
rmmateo@kunsan.ac.kr
** 종신회원 : 군산대학교 정보통신공학과 교수
jwlee@kunsan.ac.kr (교신저자)
[2012/03/26 투고 – 2012/05/03 심사 – 2012/07/24 심사완료]
☆ This work (Grants No. 00047659) was supported by Business for Cooperative R & D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2011.

sensors wherein nearby cars in the parking area are detected based on the radio signal strength. The parking sensors cooperate by sending their radio signal strength information to decide the current location of a car. After parking, cars are monitored for possible collision events by a sensor which was placed inside the car. In implementing the messaging of parking information system to clients, the infrastructure for message delivery should be considered. Socket network programs can be used to stream messages but message persistence is not supported in this technique and frequent re-connections are needed which is not appropriate in mobile environment. In our proposed framework, the messaging system of parking information system uses a push notification framework and considers message persistence. The message will keep on updating drivers and provide timely information. However, the current push notification system is inefficient in sending information where messages are broadcasted throughout the registered clients. In the proposed framework, the efficient message delivery is provided by filtering messages from information servers. Prior to broadcasting messages to clients, their individual properties and preferences will be used to decide whether to send the message to that client. A heuristic approach is used in the proposed filtering method wherein property values are transformed into numeric values to be calculated in a range function. These properties can be personal preferences or affiliations to a sender. The proposed filtering method is integrated in the messaging mechanism of the parking information system which supports applications such as smart phone-based park guiding system and collision notification.

2. Related Work

2.1 Parking Information System

A parking information system eliminates parking difficulties by making it easier to find available parking slots. Firstly, events are sensed by sensor devices and then processed these data to be sent as information to clients. Wireless sensors are used in the parking management systems to provide smart monitoring of events. Low-cost wireless sensors are deployed in a parking area which detect and monitor vacant parking slots. The events detected by sensor nodes are reported periodically to the management system. The database is accessed by an upper layer component to perform various management functions such as finding vacant parking lots, auto-toll, security management, and statistic reports. A work in [4] implements a shortest path algorithm to determine the nearest entrance of the selected parking lot. A star (A*) shortest path algorithm is used to find the least-cost path from a given origin node to the destination node. In [5], collaboration of cars is studied for efficient dissemination of information about parking spaces. The reports from wireless sensor nodes in [5] are passed from car-to-car in order to achieve scalable dissemination of information regarding parking spaces. However, a simple display monitor cannot update the drivers while driving inside the parking area. We solve this problem by providing services in a smart phone to guide the driver inside the parking area. In this paper, the use of smart phone together with interactions with wireless sensors to guide the car owners in parking is presented. We propose a parking guidance system using the information from wireless sensors to locate a parking area. The system also supports a notification system for events such as collisions.

2.2 Messaging in Wireless Mobile System

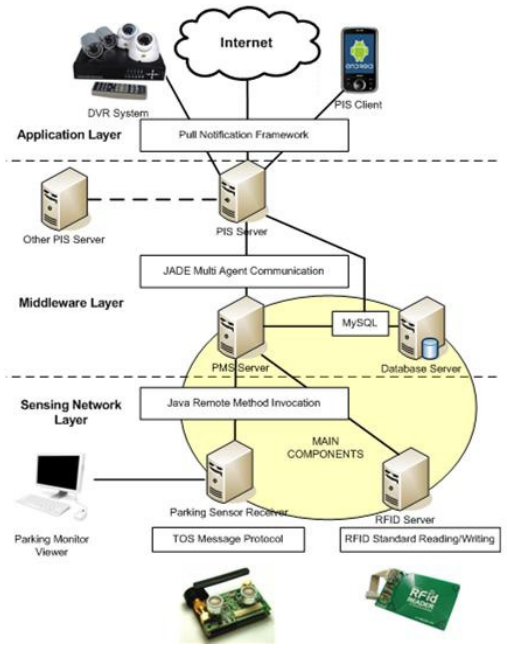
The earlier messaging technologies like Short Message Service (SMS) and Wireless Application Protocols (WAP) use messaging infrastructure built within telecommunications companies. While SMS was limited to simple messages, WAP supported access to the Internet for mobile web browsers but still limited of data transfers. Nowadays, the infrastructure of telecommunications is expanding to support fast Internet access like WiMax (USA, Japan) and WiBro (Korea). These technologies are designed based on a client-server model. Technology in mobile phones also exploited the use of peer-to-peer (P2P). There are three main traditional P2P architectures for mobile P2P networks: centralized P2P, decentralized P2P and semi-centralized P2P [6, 7]. Most studies use the Session Initiation Protocol (SIP) as the underlying signaling protocol for mobile peer-to-peer [6, 8, 9]. SIP is an application-layer control protocol that allows a host attached to a network to establish, modify, and

terminate multimedia sessions such as Internet telephony, Internet conferencing, event notification and instant messaging. Similarly, a study in [7] described a Plug-and-Play Application Platform (PnPAP) to enable dynamic selection between diverse P2P and session management while preserving the best network connectivity.

Nowadays, smart phones support high bandwidth connections such as WiFi and are readily designed for Internet use. Smart phones are popularly used to get recent information from connected peers and social web sites which make use of push and poll frameworks. Push system is a type of communication in which information is “pushed” by the server towards the browser, contrary to the normal Pull principle whereby the browser initiates the request for information. For an application to operate in Push mode, a Push channel is required through which the data is fed from the server to client. Polling is a mechanism whereby a request is sent by the client to the server at regular intervals. In return, the server updates the status of each connected client. There are various push notification frameworks to choose from like Extensible Message and Presence Protocol (XMPP) and Android Cloud to Device Messaging Framework (C2DM) which impose message persistency. XMPP is a protocol for sending XML formatted messages [10] which is a popular in mobile messaging system. The XMPP is customizable where techniques are more flexible in processing messages. Also, Google introduced the Android C2DM to provide the public users their P2P sharing model [11]. Usually, this technology produces message overhead because of the broadcast method. In our paper, we provide a filtering method to choose only users who have interest with or specific to the notification message.

3. Intelligent Distributed Parking Management System

In this paper, the Intelligent Distributed Parking Management System (IDPMS) is used as the over-all infrastructure for the parking information system illustrated in Figure 1. IDPMS, which is an improvement from [2], is classified into 3 layers which are the sensing network layer, middleware layer and application layer. Each layer defines a distinct interaction from its components. In the sensing network layer, event sensing and processing of sensor data values are done to transform these values into information which will be processed in the middleware layer. Standard communication protocols for wireless sensors and RMI communications are considered in this layer. The data are gathered from devices like RFID readers and parking sensors within the parking lot. After transforming this into information, it will be forwarded to the PMS server using Java Remote Method Invocation (RMI) as shown at the bottom of Figure 1.



(Figure 1) Intelligent Distributed Parking Management System.

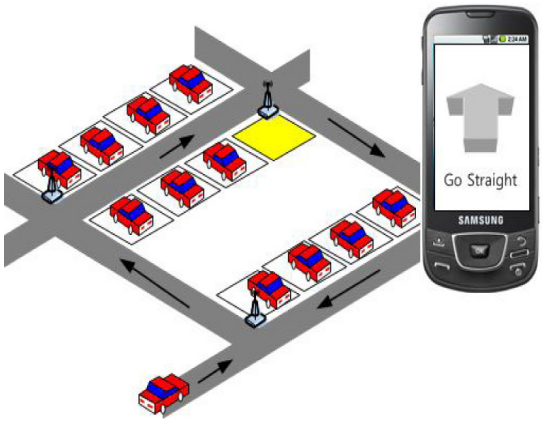
The PMS server provides an RMI server interface to update the information gathered from sensors and other devices. The components from this layer only gather sensor data but it does not have intelligence to analyze information. Also, one-way communication is implemented in which the PMS server only receives information from the sensors and RFID readers. The middleware layer is provided with intelligence to process information. The components in this layer are assigned with distributed tasks and use a more

complex messaging system. The components are represented by agents and have specific tasks where the means of communication is more on conversational. Components used in parking management such as PMS server, database server and PIS server use multi-agent communication in the middleware layer. Information from sensing network and middleware layers are secured and not shared to the public. In the application layer, client communication is considered and information is accessed publicly. This layer provides a representation of information from the sensors and information from the parking management. The Parking Information System (PIS) is processing between the clients and the parking management system. Smart phones and some components that can be accessed in public are using the notification messaging.

The parking flow is described by the following: 1) a driver picks up a collision sensor, which is called *ParkSENSE*, from the gate before entering the premises of a parking lot. The driver manually seeks for parking lot with the guidance of PIS client installed in his/her smart phone. 2) After choosing an empty slot, a car that has parked will be detected by a parking sensor and it will process the data to determine its location area. The parking sensor sends a message to the PMS server to record the parking event and to store the data which includes parking time, area ID and identification of car. A car owner should activate the *ParkSENSE* for collision sensing after parking and leaving his/her car, and 3) before exiting the parking area, the collision sensing of *ParkSENSE* should be deactivated. *ParkSENSE* is controlled by the PIS client in the smart phone of the driver. Also, a digital video recorder captures the video of parking events inside the parking area which is also used in collision verification method.

3.1 Parking Guidance System using Smart Phones

The Parking Guidance System (PGS) is classified in application layer while its method of communication is processed in the middleware and sensing network layers. The smart phone of a car owner is installed with a PIS client which includes the PGS program in its module. A *parking slot* is an area for a single car to park and a *parking area* consists of several parking slots. The numbers of parking slots in each parking area vary and this is adjusted by an administrator of the parking management. An establishment that owns a *parking lot* has several parking area. IDPMS is used by the establishment to manage the parking lot. Each parking area has a sensor used for PGS and these sensors are called *parking sensors*. PIS server informs a car entering the parking lot about the availability of a parking area and the possible routes to a vacant parking slot with the information from parking sensors. When the car approaches near a parking area, a parking sensor is queried by PIS server if there is a vacant slot. The location of a car is determined by the radio signal strength (RSS) where the location area is decided if the RSS from a collision sensor is higher on that area. If there is an available slot then the PGS will command the driver to GO INSIDE, or else, the PGS will find a route for the next available slot of a parking area. *R* contains the route table of the next parking area and this is sent to the PGS. A single route contains commands, GO STRAIGHT, TURN RIGHT and TURN LEFT. The commands are associated to a parking area led by a route which is manually configured by an administrator or route experts. These commands are executed accordingly when a driver is near a parking sensor. Figure 2 illustrates the guidance system using parking sensors with a number of cars in the parking lot. A yellow box represents a vacant slot and this information is sent to



(Figure 2) Parking Guidance System using parking sensors in parking area.

the smart phone of a client entering the parking lot. The client then uses its smart phone as a guide to that parking slot.

Figure 3 shows the pseudo code of routing a parking area with an available parking slot (u_t) in all parking areas (U). The routes (R) that are sent to PGS are the possible shortest route to a parking slot. Every time the PIS server determines a car's location (c_t), a command will be imposed to PGS. These commands are stored in the recommended route (r). These routes were identified by an administrator and also accompanied with the shortest route which is set at the PIS server.

```
// locating a vacant slot
u_t = find Vacant(U)
// determine the parking area
c_t = find ParkingArea(u_t,C)
// selecting the routes
R = select Routes(c_t,C)
r = select Recommend(R)

// instructions of PIS server to PGS
 for each c_i ∈ r do
      if c_i = c_t
    instruct c_i     with GO INSIDE
  else if c_i+1 STRAIGHT
    instruct c_i     with GO STRAIGHT
      else if c_i+1 RIGHT
    instruct c_i     with TURN RIGHT
      else if c_i+1 LEFT
    instruct c_i     with TURN LEFT
  end if
  i++
 end for
```

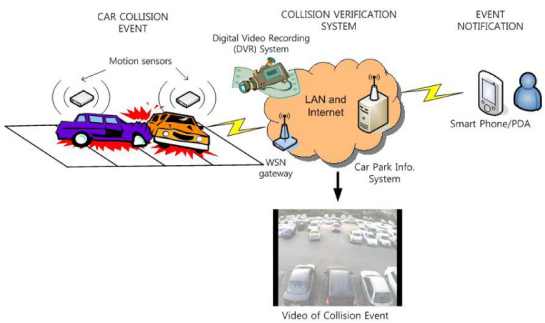
(Figure 3) Pseudo code of finding and selecting routes to a vacant slot. Each route consists of commands associated to parking areas.

Equation 1 calculates each route of all parking areas. PGS will only consider the best route which is informed by PIS server about the routes to a parking area with a vacant slot. If there are two or more selected routes then it will choose at random. While parking inside, PGS will always be updated by the PIS server of the available slots after a car drives near the parking area.

$$\min(R) = R = \sum_{i=1}^{I} \sum_{j=1}^{J} route(Area_i, Area_j) \quad (1)$$

3.2. Car Collision Notification System

In the collision detection, the collision analyzer agent in [2] was used. The collision analyzer agent handles the detection and verification of collisions that occur inside the parking area. After the verification, the PMS server that receives messages from the collision analyzer agent informs the parking information system. There are three steps for the car collision detection which are: 1) *collision sensing*, 2) *collision verification* and 3) *car owner notification* illustrated in Figure 4. After the collision analyzer agent verified the collision and identified the cars that collided, it reports the car collision event to the PMS server and then the PMS server will send the information to PIS server. The collision information includes identification of cars affected by the collision.



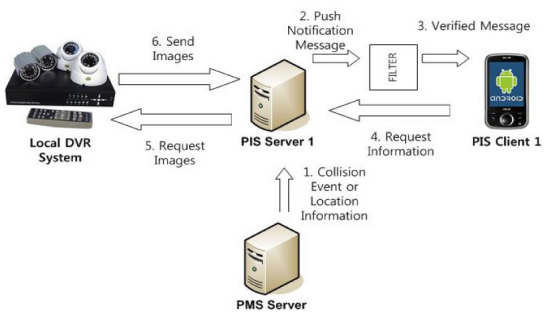
(Figure 4) Car collision detection, verification and notification methods.

4. Filtered-Push Notification Framework

The proposed filtered-push notification framework for PIS is illustrated in Figure 5. The push notification is done in the application layer by its components. The information processed in the middleware layer is forwarded to the application layer and then the dissemination of information will be managed by PIS server to be sent to clients. Messages that are processed in this layer are in XML format. Also, smart phones are provided with libraries to support processing of XML messages which is already included in the PIS client program. A driver has to be

registered to PIS server before it can receive notification messages or use other applications of the parking management system. PIS applications such as parking guidance system and collision notification use the push notification framework for its messaging system. The procedure of push notification system illustrated in Figure 5 is described by the following:

1. PMS server sends event information to PIS server and then PIS will process notification events. Examples of these events are collision notification and location information requested by the PIS client.
2. PIS server processes filtering before sending notifications to clients. If the notification is for a specific person/driver then filtering will be used to select clients.
3. Clients, which were selected by the filtering, are sent with the notification message.
4. Registered PIS client sends a request for information to the PIS server.
5. PIS server requests for images from the DVR System.
6. DVR system sends images to the PIS Server. The images that were sent will be viewed by PIS clients that were notified by a collision event.



(Figure 5) Procedures of the push-notification system for the PIS.

The filtering method uses the input properties from PIS client, and tags from notification message. The input properties from a driver is represented by $n = \{u_1, u_2, \cdots, u_n\}$ are compared to the notification tags represented by $t = \{t_1, t_2, \cdots, t_n\}$. The notification message tag is set by a sender and this is compared to each u of a client n where all clients in session will be processed in the filtering. These tags are initially configured by a PIS administrator. Compound statements are used when comparing with two or more client properties. The OR statement is used if one of the properties is true while AND statement is used if the search requires two input properties in the search, e.g., u_1 OR u_2 returns all services that satisfies either u_1 or u_2 while u_1 AND u_2 returns all services that contains both u_1 and u_2. The properties with outcome values from a notification and input properties are compared. An outcome value can be represented by a string or a numerical value. The system matches properties and notification tags to process filtering. For example, if notification tag is *relative* of driver A, obviously, the filtering will query the clients who are relatives of driver A. All n that satisfy the condition for filtering in Equation 2 are gathered where 0 indicates there is no similar properties from tags while 1 indicates there is at least one similar property in sending the notification.

$$n(t) = \{0,1\} = \sum_{n=1}^{N} \sum_{i=1}^{I} f(u_n, t_i) \tag{2}$$

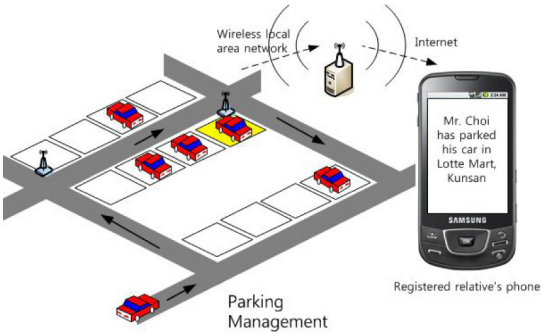
$$rx = \frac{t_{\max} - t_{\min}}{m} \tag{3}$$

$$\begin{aligned} &f(u,t) = \{x_{\min} <= t < x_{\max} \mid \\ &x_{\max} = t_{\min} + (rx \times cat(u)), x_{\min} = x_{\max} - rx, \\ &if\, cat(u) = m \quad then \quad x_{\max} = t_{\max} \;\&\; t <= x_{\max}\} \end{aligned} \tag{4}$$

Equation 2 is the filtering function where all clients are processed by filtering and then selected clients are sent with notification message. The u and t are compared using the function $f(u, t)$. If the value of either u or t is categorical and other is numerical then categorical values will be transformed into numerical values. Equation 3 and Equation 4 are used to transform categorical values from u referring to numerical values from t. In Equation 3, the rx represents the length of each categorical value to transform into numerical value. The maximum value from a property u represented by t_{max} is subtracted to the minimum value represented by t_{min} and the result is divided by the number of categories represented by m. In Equation 4, the filtering function will choose all clients that satisfies the condition $x_{min} <= t < x_{max}$ where x_{min} is the starting range value of u and x_{max} is the ending range value of u. The $cat(u)$ is a

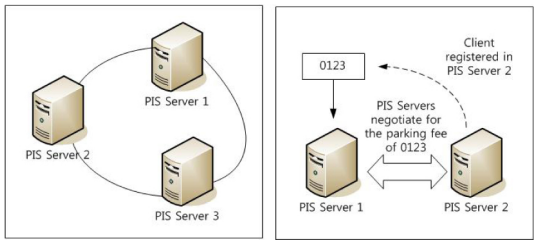
function that determines the index category of a preference. If m=3 then indexes are $cat(u)$={1,2,3}. If u is the last indexed category from the selection then x_{max} will be equal to t_{max} and t will be filtered using the condition x_{min} <= t <=x_{max}. The results from filtering are listed to *selected_clients* and each client will be sent a notification message.

The notification message will be sent to PIS clients after processing the filtering method. In Figure 6, a final result from using the filtering algorithm is illustrated where relatives of a driver are sent with notifications after that driver parks in an establishment. The configurations of properties are set manually by the driver who wants to receive notification messages.



(Figure 6) A notification message is sent to the relative of a driver after filtering.

PIS servers also form P2P network to other PIS servers. The P2P network allows PIS servers to interact and negotiate using an agent approach. The negotiation includes allowing the use of services from other parking systems. In Figure 7A, a P2P network of PIS servers is illustrated where logical links over the Internet are established. Negotiations to use parking services to different establishments are supported in the peer linked PIS servers. An example of negotiation is illustrated in Figure 7B where a registered client in PIS server 2 is allowed to park using PIS server 1 and use its account through negotiations from PIS server 1 and PIS server 2. After negotiations, the PIS server 2 will pay the parking fee to PIS server 1. The client does not need to register to other PIS servers but the system will automatically negotiate and handle transactions.

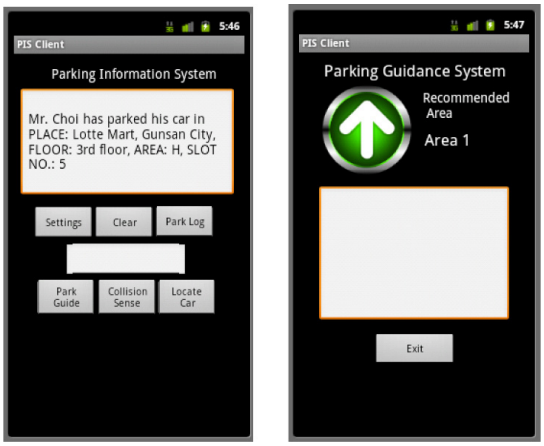


(A) (B)

(Figure 7) Peer links of PIS servers in A are used to perform negotiation within different establishments illustrated in B.

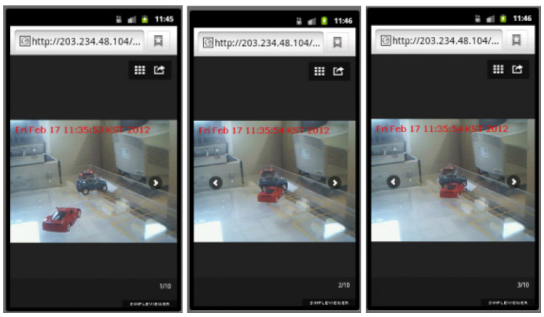
5. Implementation of IDPMS

The IDPMS was developed in Java and different class libraries were used which are, 1) RMI for parking sensors and RFID to communicate with PMS server, 2) JADE Framework for interactions of components in the middleware layer and 3) XMPP application server for PIS server. The PIS client used Android OS with libraries for XMPP to communicate with PIS server. Wireless sensor motes with Zigbee specification were used for parking sensors to process the proposed guidance system. The HBE-SM5-S4210 installed with Android OS was used to evaluate the filtering technique for messaging performance. In Figure 8, the PGS program (B) was built as a sub form of PIS client (A). To communicate with the wireless sensors inside the parking lot, the application layer requests information to PMS server. Every time a vacant slot is determined by parking sensors, PIS server informs PGS of the availability of area through the messaging system of IDPMS. In Figure 9, the images of collision event viewed using PIS client program are shown. The image viewer is executed after a collision notification was sent to the smart phone. In Figure 9A, a car is about to approach another car, in Figure 9B, the car is very near to another car and in Figure 9C, the car collides and collision sensor senses this event which triggers message sending to PMS server.



(A) (B)

(Figure 8) PIS main interface in A opens the PGS activity in B after clicking the Park Guide command button of PIS client program.



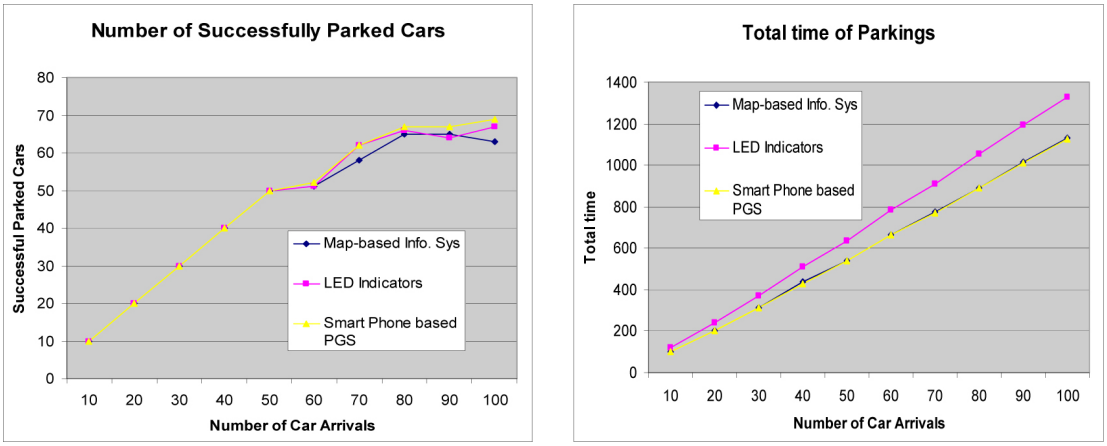
(A) (B) (C)

(Figure 9) Images of collision event viewed by using the PIS client. In A, a car is about to approach another car, in B, the car is very near to another car and in C, the car collides and collision sensor sends collision notification.

5.2. Performance Evaluation of Messaging System

In the performance evaluation, a discrete event simulation of car arrivals and car parking was performed. The total time of parking inside a parking lot and the number of successful parked cars were observed. In the simulation, 3 parking areas were set and each area has 10 parking slots. The arrival time of cars was set in random. Also, a parking time from each car was set randomly. The parking lot is only one way driving. A car will go to the next parking area if there are no available parking slots. If a car passed the last parking area then it will exit after passing through that area and labels this as unsuccessful parking. Two conventional methods were compared to the proposed method which is the map-based information system and use of LED indicators. In the map-based information system a driver is informed of vacant slots by looking on a display screen before it parks while the LED indicator method uses post light beneath a parking slot to inform a driver of its vacancy. In LED indicators, a driver will be guided by LED indicators where the driver needs to drive slowly while inside the area because of unfamiliar place. In a map-based information system, a driver will search for vacant slot by looking at the map on a monitor screen. This will give him prior knowledge to map the vacant slot before he or she enters the parking lot. In this case, he can drive faster compared to using LED indicators to search vacant slots inside the parking area. However, vacant slots might be occupied immediately because some cars that already entered the parking lot might have found it first and parked in that slot. This can be solved by combining the two methods, however, expensive to implement. The PGS works like combining the methods ignoring the cost of installing additional devices (LED indicator) and only smart phone and PIS software are used. In evaluating our method, the proposed method was also set with the same environment with map-based as well as LED indicator. The PGS and maps based methods were configured with 20% faster than LED indicator method. There are two performance metrics; these are the total time of cars parked and the total number of successfully parked cars. The arrival time and period time of parking are generated randomly. The arrival time is selected between 0 and 100 seconds and also the period of parking is between 0 and 100 seconds.

Figure 10 shows the result of successfully parked cars and total time of parking in the simulation where it was observed that there were no vacant slots to park incoming cars when 70 or more cars are currently inside the parking area. The cars in LED indicator implementation were slower



(A) (B)

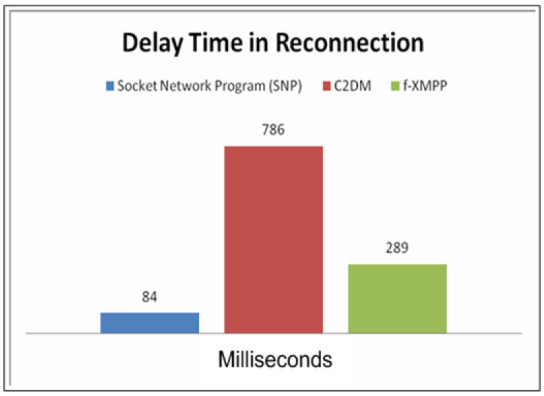
(Figure 10) Number of successful parked cars in A and total time of parked cars in B.

in finding vacant slots while the smart phone-based PGS and map-based information system had almost the same performance in total time, however, in 70 car arrivals, the number of successfully parked cars of the smart phone-based PGS was higher than other two methods.

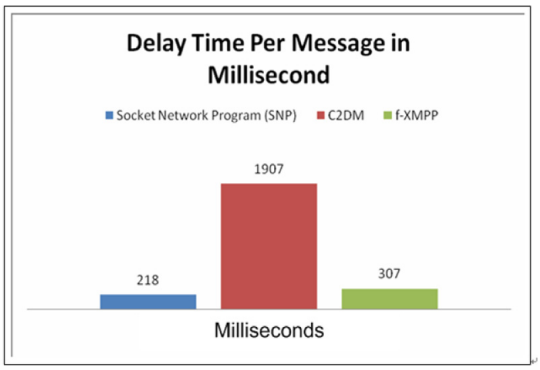
5.2. Message Passing Performance

The messaging performance of the filtered-push notification framework was compared to other methods for evaluation. A server installed with Linux OS for PIS and an Android based smart phone were used. The performance of message persistence was measured in terms of delay time in re-establishing a disconnected session. The socket network program (SNP) and Android C2DM were compared to the proposed filtered-XMPP (f-XMPP). SNP used a handler to reestablish connection which was coded for the purpose of our simulation while C2DM and f-XMPP used the libraries to re-establish its sessions. Moreover, the performance in sending a message was measured by its delay time. The content of each message consisted of 10 words and was increased until 1000 words through the simulation. It was observed that there was no huge difference in sending 1000 words compared to 10 words only. Each case was repeated 10 times and then the total result was averaged for the final values.

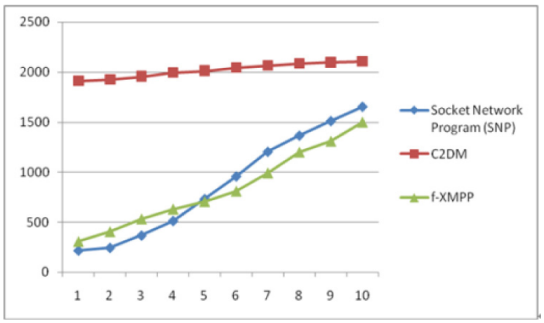
In Figure 11, the result shows that SNP is fastest in a reconnection. Simple reconnections can be managed by SNP but C2DM and f-XMPP can handle more complex issues in re-establishing sessions. It was observed that f-XMPP was faster in re-establishing the connection than C2DM. The C2DM handled the messaging system using its complex system and resulted to more delays. The delay time of sending a message is shown in Figure 12 which has a similar trend from Figure 11. As discussed previously, the content volume of a message had less significance in the delay time of sending a message. Figure 13 shows the messaging performance by increasing the number of messages. Performing this simulation will tells us how fast a framework responds to simultaneous sending of messages. Similar to Figure 12 and 13, the complexity of message passing in C2DM provides high overhead. It was observed that SNP was faster from 1 to 4 messages but f-XMPP was faster from 5 to 10 messages. f-XMPP selected the relevant clients by comparing tags of notification messages and properties from PIS client. In Figure 12, clients to be sent by notification were processed in filtering and few which were irrelevant to PIS clients were not included. This result implies that f-XMPP was better in simultaneous sending of messages than the SNP and C2DM.



(Figure 11) Delay time in reestablishing a session in milliseconds.



(Figure 12) Delay time in sending message in milliseconds



(Figure 13) Delay time in sending message by increasing the number of messages.

6. Conclusion

A smart parking system helps eliminate difficulties in parking by making it easier and faster for drivers to seek available slots. This includes tools and devices to guide a driver on his way inside the parking area. In this paper, a smartphone-based parking guidance system was proposed to provide drivers with proper directions which uses the information gathered from parking sensors. In our simulation using our proposed method, drivers can efficiently find vacant parking slots. However, the push notification messaging used in the parking information system (PIS) produces high message overhead when sending notifications throughout the clients which causes delay time in updating the car's current location. In our proposed messaging framework, the notification method uses filtering to prevent unnecessary message overheads. This paper showed the implementation and performance evaluation of the proposed framework. It was observed that the complexity of message passing in C2DM provides a high overhead compared than f-XMPP, and f-XMPP was faster in handling simultaneous message sending than the SNP.

References

[1] E. Barata, L. Cruz, J.P. Ferreira, "Parking at the UC Campus: Problems and Solutions," *Cities*, vol. 28, no. 5, 2011, pp. 406-413.

[2] R. M. A. Mateo, Y. S. Lee, J. Lee, "Collision Detection for Ubiquitous Parking Management Based on Multi-agent System," in Proc. *International Conference Agents and Multi-agent System – Technologies and Applications*, LNAI 5559, 2009, pp. 570-578

[3] S.E. Yoo, P.K. Chong, T.H. Kim, J. Kang, D. Kim, C. Shin, K. Sung and B. Jang "PGS: Parking Guidance System based on Wireless Sensor Network," in Proc. *International Symposium on Wireless Pervasive Computing*, 2008, pp. 218-222.

[4] M.Y.I. Idris, E.M. Tamil, N.M. Noor, Z. Razak and K.W. Fong, "Parking Guidance System Utilizing Wireless Sensor Network and Ultrasonic Sensor,"

Information Technology Journal, vol. 8 no. 2, 2009, pp. 138-146.

[5] S. Miura, Y. Zhan and T. Kuroda, "Evaluation of Parking Search using Sensor Network," in Proc. *1st International Symposium on Wireless Pervasive Computing*, vol. 6, 2006.

[6] S. Lui, J. Chen, S. Zhao and F. Na, "Peer-to-peer Application in Mobile Cellular Systems," in Proc. *IEEE 5th International Conference on Information Technology New Generations*, 2008, pp. 366 - 371.

[7] E. Harjula, M. Ylianttila, J. Ala-Kurikka, J. Riekki and J. Sauvola, "Plug-and-play Application Platform: Towards Mobile Peer-to-Peer" in Proc. *ACM 3rd International Conference on Mobile and Ubiquitous Multimedia*, 2004, pp. 63 - 69.

[8] D. Howie, M. Ylianttila, E. Harjula, and J. Sauvola, "State-of-The-Art SIP for Mobile Application Super networking", in Proc. *Nordic Radio Symposium, including Finnish Wireless Communications Workshop*, 2004.

[9] L. Li, and X. Wang, "P2P File Sharing Application on Mobile Phones based on SIP," in Proc. *IEEE 4th International Conference on Innovations in Information Technology*, 2007, pp. 601-605.

[10] Extensible Message Presence Protocol, http://xmpp.org/

[11] Google inc., "Android Cloud to Device Message Framework", http://code.google.com/android/c2dm/

◐ 저 자 소 개 ◑

마테오 로미오 (Romeo Mark Mateo)

2004년 West Visayas State University, Philippines BS in Information Technology
2007년 Kunsan National University, South Korea, Master of Engineering major in Information and Telecommunications
2007년~현재 Kunsan National University, SouthKorea, Graduate student in Ph.D course
관심분야 : Distributed systems, data mining, fuzzy systems, multi-agents, ubiquitous sensor networks, cloud computing
E-mail : rmmateo@kunsan.ac.kr

이 재 완 (Jaewan Lee)

1984년 중앙대학교 이학사-전자계산학
1987년 중앙대학교 이학석사-전자계산학
1992년 중앙대학교 공학박사-전자계산학
1996년 3월~1998년 1월 한국학술진흥재단 전문위원
1992년~현재 군산대학교 교수
관심분야 : 분산 시스템, 운영체제, 실시간 시스템, 컴퓨터 네트워크, 클라우드 컴퓨팅 등
E-mail: jwlee@kunsan.ac.kr