A Hybrid Middleware for RFID-based Parking Management System using Group Communication in Overlay Networks[*]

Louie F. Cervantes
Institute of Information and Communications Technology
West Visayas State University
Luna St., Lapaz, Iloilo City, Philippines

Young-Seok Lee, Hyunho Yang, and Jaewan Lee
School of Electronic and Information Engineering
Kunsan National University
68 Miryong-dong, Kunsan City, Chonbuk, 573-701, South Korea
{lfcervantes, leeys, hhyang, jwlee}@kunsan.ac.kr

Abstract

A hybrid middleware for RFID-based parking management system is presented in this paper. The publish-subscribe approach is combined with group communication in overlay networks in the design of event monitors and application service connectors to comprise the core of a hybrid middleware. In this paper, group communication relevant to P2P networks is the focus of middleware technology development. A group of peer nodes provide an efficient abstraction to handle events from RFID readers and vehicle identity detectors to be processed by services and applications that run in the conventional IP networks. Programming abstractions for publish-substribe messaging (multicast and single) from the reference P2P systems are used to build new extensions. The hybrid middleware provides an efficient communication transport between the peer-based networks and existing application services and mobile communication infrastructure that require conventional IP-based networks. The simulation results show that the approach improved the performance of the P2P network. This paper provides an implementation model that can significantly lower the infrastructure cost for building an electronic parking management system.

[*]This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce Industry and Energy of the Korean Government.

1. Introduction

Application-level multicast has gained popularity as an active research area. Various algorithms and systems for scalable group management and reliable group communication can be found in the literature. Radio frequency identification (RFID) systems are emerging as practical means of auto-identification in a wide variety of applications from access control to product tracking. The widespread adoption of RFID is already happening and many organizations have begun using the technology in innovative ways.

Parking is an ever growing challenge in cities and town everywhere. Parking demand is routinely high at airports, downtown areas, and around transit and ride facilities. In this paper we present our design of an electronic parking management system based on RFID technology. We designed a middleware based on group communication in P2P networks to handle the communication needs of a large number of RFID readers and presence detectors distributed across a wide area in several parking facilities.

2. Related Works

2.1. Peer-to-peer Middleware

The term peer-to-peer (P2P) refers to a class of systems and applications that employ distributed resources to perform a function in a distributed manner. The resources encompass computing power, data storage and content, network bandwidth, and presence (computers, humans and other resources) [1]. The distributed function can be computing, content, and collaboration or platform services. De-

0-7695-3006-0/07 $25.00 © 2007 IEEE
DOI 10.1109/IPC.2007.112

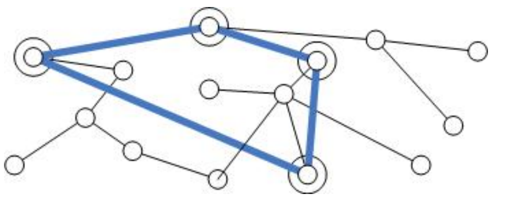


Figure 1. Peer nodes and Group Overlay

centralization may refer to data, algorithm, meta-data, or all of them.

P2P applications such as the office collaboration suite Groove [2] and telephony service Skype [3] are widely used. However, few applications beyond file sharing are being built to take advantage of the capabilities of P2P networks [4]. Messaging based middleware like CORBA, EJB, COM+ and SOAP ease the development of distributed applications but their centralized model do not fully utilize the resources in decentralized environments. The first significant peer-to-peer middleware and platform is JXTA [5]. JXTA provides a general middleware useable for a variety of applications. Other P2P platforms include Microsoft's Peer-to-Peer SDK [6], Pastry [7], Chord [8] and CAN [9]. The current work is heavily influenced by Scribe [10].

Our approach focuses on group communication especially relevant to P2P applications. The strength of the P2P approach lies in the collaboration of several peers, allowing each peer to profit from the others. Collaboration in a group requires communication among its members, thus the need for group communication.

2.2. Peer-to-Peer Group Communication

A *group* consists of several peers in a peer-to-peer network having a common interest in a specific topic. These peers are subscribers of a topic in a publish/subscribe domain. When a message of interest is sent, the peer-to-peer network can handle the delivery to each interested recipient. However, this approach can be inefficient even if the routing is optimal. The bandwidth of non-subscribed peers is utilized even if they are not interested in the message. A group overlay network is created on top of the P2P network to provide a more efficient communication service independent of the underlying network. We choose the ring topology to build the overlay network. Rings are scalable in terms of bandwidth, because the workload of the nodes is independent of the total number of nodes. Backup and repair procedures can be easily included due to the manageability and simplicity of the ring topology. Figure 1 shows the group overlay created on top of the P2P nodes.

2.3. Parking Management System

Electronic parking management system (ePMS) maintains a real-time parking space inventory across a set of participating facilities. The data can be used to generate parking availability advisory presented to the public using electronic display boards or similar means. ePMS helps drivers find parking lots quickly, thereby reducing frustration and enhancing their overall driving experience[11].

The main motivation for using RFID technology in our design of ePMS is the significant lower infrastructure cost of implementation compared to other approaches. An electronic parking system (EPS) is presented in [12]. Their design focused on automated ticketing and a charging system that integrates with a toll collection facility called Electronic Road Pricing (ERP). The system performance was reported as reliable and their work has been marketed as a mature product. In this work, we seek to achieve comparable or better performance results in parking management while avoiding the use of costly infrastructure such as the case of the previous study. In another paper [13], wireless sensors were used for the dual role of presence and identity detection. Their approach relied on IP multicast using flooding. The results obtained indicated limited success due to the significant communication cost incurred in transmitting data to the sink node.

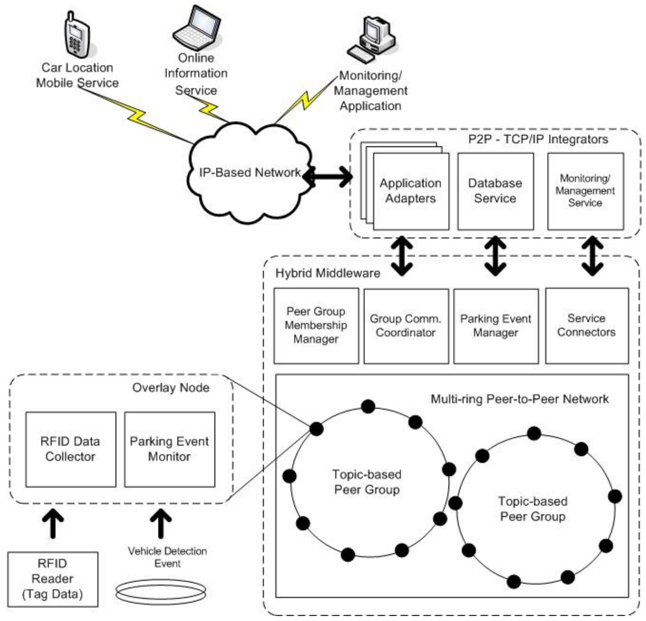


Figure 2. Architecture of ePMS

3. Architecture of Hybrid Middleware for RFID-based ePMS

Our hybrid middleware sits between the RFID hardware and the enterprise applications or conventional middleware. The main goal of this middleware is to process the data from tags collected by the readers deployed in the RFID infrastructure. Figure 2 shows the architecture of our hybrid middleware. In this work, group-based communication is achieved using the topic based publish-subscribe approach in Scribe [14]. Scribe is a large- scale decentralized event notification infrastructure built upon Pastry [7], a scalable, peer-to-peer location and routing substrate.

3.1. Hybrid Middleware Components

Peer Group Membership Manager - this node is used to create all the groups. All group related tasks including to create/destroy a group or to change group membership a corresponding notify message is send to this node. The purpose is to enhance the management of group membership more efficient.

Group Communication Coordinator - this node controls the rate of heartbeat messages that sends a multicast message to other member nodes. A heartbeat message ensures that changes to parking status such as when a slot becomes vacant after a vehicle has left, will be robustly detected with minimal delay even when the vehicle presence detector fails to detect the event. This behavior is based on the routing tree repair mechanism but in this case it is used to ensure the accurate detection of the parking space status.

Parking Event Manager - this node acts as the sink node for all parking events. The details for sending the parking event data to the database service is handled by this node.

Service Connectors - Specific nodes are tasked to perform specialized roles in the peer groups such as provide management control of the individual nodes or a subset of the overlay group.

3.2. Overlay Node

Each RFID reader is attached to a host. The detection region of the reader can span several parking slots. The presence detector in each slot provides the trigger that fires the parking event (entering or leaving) and activates the reader which obtains the identification data from the RFID tag. Each parking slot is associated with a peer node thus a parking event fired up by the presence detector is handled as a message by this node which acts as a parking event handler. All nodes within a single parking space form a group and share a common groupID.

Our approach describes two types of parking events: ENTERING event occurs when a vehicle comes into the parking slot and LEAVING event occurs when it vacates the parking slot.

We define the overlay node as a data collection module and parking event monitor. To make communication with the group more efficient, the group is create corresponding to the set of detectors and RFID readers that are co-located in a particular parking space. This optimizes the physical proximity of group members ensuring the minimum number of hops for multicast messages.

```
deliver(msg, key)
  switch (msg_type)
  case CREATE:
    topics.add(msg.topic)
    invokeNotifyGroupManager(nodeID)
  case JOIN:
    topics.children.add(msg.source)
    invokeNotifyGroupManager(nodeID)
  case MULTICAST:
    for each node in topics.children
      send(msg, node)

    if subscribed(msg.topic)
       invokeEventHandler(msg.topic, msg)

  case LEAVE:
    topics.children.remove(msg.source)
invokeNotifyGroupManager(nodeID)
    if topics[msg.topic].children = 0
      msg.source = thisNodeID
      send(msg, topics[msg.topic].parent)
```

Figure 3. Implementation of deliver

3.3 Group Communication Model

In this paper, the design of a hybrid middleware proposes novel extensions to Scribe [10]. The group communication approach in Scribe provided a suitable base for the design of the core functionality of our hybrid middleware. Our innovation is focused on extending these mechanisms to create adapters between the P2P network and a centralized database server and other application services. This abstraction of peer nodes into a group permits a service to interact with the group as if it is a single object. On the other hand, peers within the group view the service as if it is another member peer.

The following relevant APIs in Scribe were modified and extended with new functionality to satisfy the requirements of this system:

create(credentials, groupID) creates a group with the specified groupID. Unlike the implementation in Scribe which allows any node to create a new group, we assign a node to handle group creation and maintain a record of group membership. *join(credentials, groupID, eventHandler)* causes the local node to join the group. All messages for this group are received by this member node and events are handled by the specified event handler. *leave(credentials, groupID)* causes the local node to leave the group and be removed from the group membership record. *multicast(credentials, groupID, event)* causes the event message to be delivered to all the members nodes that are subscribed to this group. *send(msg, nodeID)* causes the local node to route the message to the node specified by nodeID.

The *deliver function* handles message delivery events in each node. Figure 3 shows the implementation of deliver in ePMS.

4. Implementation of RFID-based Electronic Parking Management System

The electronic PMS have the following basic hardware features:

1. The system should be able to detect occupancy of the individual parking slot, i.e., determine if the slot is empty or occupied.

2. Two types of events should be clearly distinguishable: ENTERING - the vehicle has moved into the slot, and LEAVING - the vehicle is exiting the slot. Proper detection thresholds should be configured to avoid false detections when vehicles momentarily pass the detection region.

3. Distinct mechanism for two types of detection is supported: presence detection and identity detection. Various mechanisms of the practical implementation of presence detection is presented in the literature [12] [13]. This work is an implementation of parked vehicle identity detection using RFID as the underlying technology.

4. Each authorized vehicle is provided a passive RFID tag attach using a tamper proof mechanism in the front section of the vehicle placed in a location for optimal detection. Authorized vehicles are those that belong to faculty, staff and students. Similar tags are issued to visitors but can be mounted temporarily.

5. RFID readers are distributed in the parking spaces, in a manner that ensures adequate coverage of all parking slots.

6. Inexpensive mechanical presence detectors are installed per individual parking slot.

Our design of the ePMS is based on the needs of the parking management in a university campus. Our system aims to provide the following application services for:

1. Real-time monitoring to support enforcement of parking space privileges for faculty and students;

2. An on-demand search utility to locate a vehicle using mobile phone or a web interface;

3. A system for a campus-wide security log of parking space occupancy; and

4. Automated ticket and parking fee collection is a future feature goal. However, the current implementation anticipates this need by providing appropriate data fields compatible with a parking fee collection application.

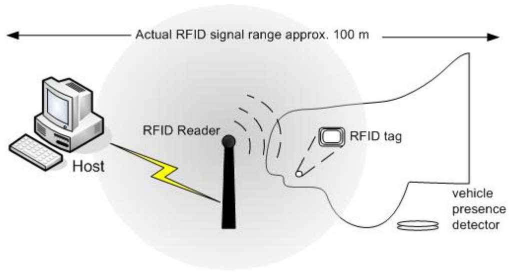

Actual RFID signal range approx. 100 m
RFID Reader
RFID tag
Host
vehicle presence detector

Figure 4. Parked Vehicle Identity Detection

As designed, ePMS consists of RFID sensors, vehicle presence detectors, host computers, communication/power infrastructure, and Internet based applications for system monitoring and user queries. Figure 4 shows our implementation of identity detection using RFID technology in ePMS.

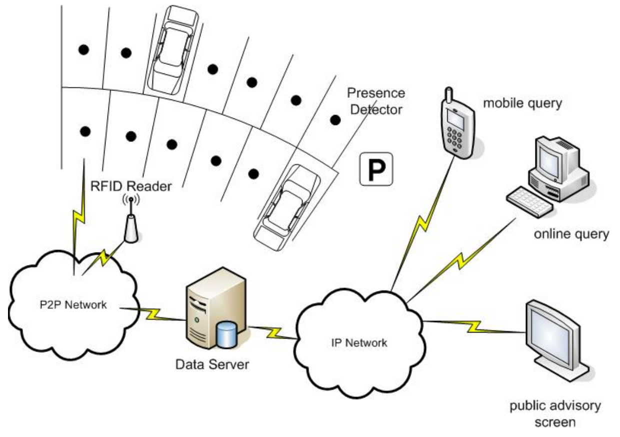
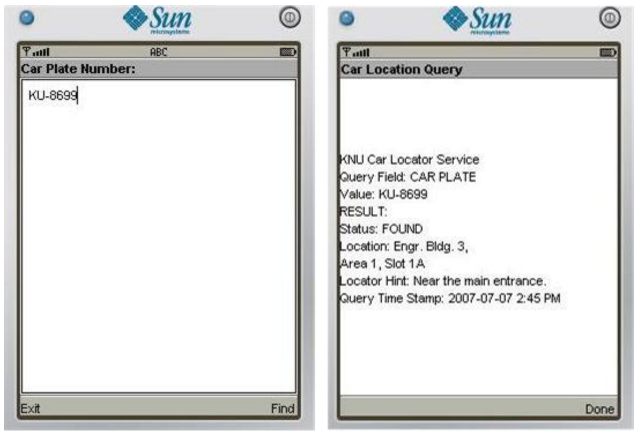


Figure 5. Implementation Model

The focus of the current work is on the hybrid middleware to support group communication among peer nodes that handle parking event messages and connect with conventional services. The implementation diagram is shown in figure 5. The P2P network and the overlay group of sensor nodes provide the abstraction for the individual parking slots and a mechanism to efficiently manage the delivery of parking events data generated when cars enter and leave parking slots.

The mobile car location service is one of the key services of this system. Figure 6 shows the user interface of a J2ME/MIDP mobile phone emulator as it connects to the car location service. The car location service itself is a Java servlet that runs on a server connected to an IP-based network. A similar interface is provided for the web browser, thus making the car location service accessible to an Internet kiosk in a designated place within the facility. A global monitoring service provides real-time information about parking space availability for the entire system. This system can be easily extended to include online reservation and payment collection if later required.



Car Plate Number:

KU-8699

Car Location Query

KNU Car Locator Service
Query Field: CAR PLATE
Value: KU-8699
RESULT:
Status: FOUND
Location: Engr. Bldg. 3,
Area 1, Slot 1A
Locator Hint: Near the main entrance.
Query Time Stamp: 2007-07-07 2:45 PM

Figure 6. GUI of Mobile Car Location Service

5. Experimental Results

In this section we present the experimental results quantifying the performance of modified Scribe (mScribe) and standard Scribe. The results were obtained using a Scribe simulation using 10 groups with varying group size of 10 to 100 member nodes.

The unmodified Scribe uses Pastry to manage topic or group creation, subscription and to build a per-topic multicast tree used to disseminate the events published in the topic. This approach is fully decentralized, all decisions are based on local information, and each node has exactly identical capabilities as the others. This means that every node can be a publisher, a root of a multicast tree, a subscriber to a topic, a node within a multicast tree and any combinations of these. mScribe assigned certain nodes with extended roles and all other nodes are either nodes in a group multicast tree (message forwarder) or message senders (origin of send messages).

In mScribe, topic creation tasks were performed using one node (the Group Membership Manager) which then selects certain nodes as the multicast tree roots (Group Communication Coordinators) and maintains the record. This means that each group has a rendezvous node just like in the original design of Scribe but now a global record of all roots is maintained to avoid the cost of routing across many nodes to locate the root. We also define one node to act as the sink node for the group to receive parking event messages from the member nodes. The sink node is any node other than rendezvous node of the group. This ensures that communication from member nodes to the sink node will always be single hop. Otherwise the node will require multi-hop Pastry routing to send the message to the sink node.

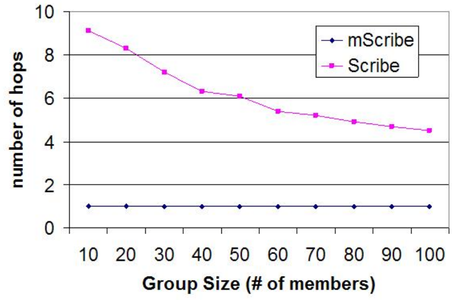


Figure 7. Average hops to reach sink node

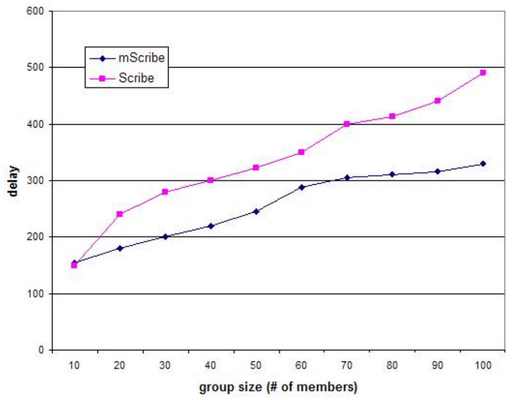


Figure 8. Average roundtrip delay between request and reply

In figure 7 we find that bigger group size helps improve the performance of Scribe yet our approach to use the a single hop send and not route through Pastry greatly reduce latency particularly for small group size.

In figure 8 we measured the round trip delay of a multicast request from group coordinator node and the reply from all node members. For Scribe we used Pastry routing in both directions, and for mScribe only the request had to use standard Scribe multicast and the reply used the single hop message send.

6. Summary and Conclusions

The paper described a hybrid middleware for RFID-based electronic parking system using group communication in overlay networks. The proposed hybrid middleware consists of overlay group of peer nodes that serve as event monitors and service connectors that bridge the P2P and conventional IP networks. Our simulation results indicate that network performance in terms of reduced message hops and round trip delay is improved by the proposed approach. Our future work will focus on improving the communication performance of our hybrid network and performing additional simulations to measure the efficiency of our approach in comparison to other related systems. We foresee interesting research in the development of large-scale multi-agent systems using the P2P networking. We continue the development of our platform to define additional functionality of hybrid middleware for general application areas.

In conclusion, while our work is preliminary and much improvement of the platform remains to be done, we have demonstrated that a fresh approach, such as what was presented in this paper, can spur more research in the design and deployment of hybrid middleware to combine the strengths of both P2P networks and the conventional IP network.

References

[1] D. Melojicic. Peer-to-Peer Computing, *Technical Report HPL-2002-57.* Available at http://www.hpl.hp.com/techreports/2002/HPL-2002-57R1.pdf, 2002

[2] Groove Networks. *Groove Networks Website.* Available at http://www.groove.net, 2007

[3] Skype. *Skype Homepage.* Available at http://www.skype.com, 2007

[4] M. Junginger and Y. Lee. Peer-to-Peer Middleware, *In Proc. of Second International Conference on Peer-to-Peer Computing.* pp.49-56, 2002

[5] Sun Microsystem. *Project JXTA Homepage.* Available at http://www.jxta.org, 2007

[6] Microsoft. *Microsoft Windows XP Peer-to-peer Software Development Kit (SDK).* Available at http://msdn.microsoft.com/library/default.asp?url= downloads/list/winxppeer.asp, 2003

[7] A. Rowston, and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large Peer-to-peer Systems *In Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware),* pp. 329-350, November 2001

[8] I. Stoica, R. Morris, D. Karger, M.F. Kashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.*In Proc. SIGCOMM2001,* August, 2001

[9] S. Ratsanamy, P. Francis, M. Hadley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. *In Proc. SIGCOMM2001,* August, 2001

[10] A. Rowston, A.M. Kermarrec, M. Castro, and P. Druschel. Scribe: The Design of a large-scale event notification infrastructure. *In Proc. NGC'01,* November, 2001

[11] Federal Highway Administration. Advanced Parking Management Systems: A Cross-Cutting Study. *US Department of Transportation,* January, 2007

[12] T. Obota, H. Ono, Y. Miyazaki, and M. Ando. Electronic Parking System for Singapore. *Mitsubishi Heavy Industries, Ltd. Teahnical Review,* 40-3, June, 2003

[13] J. Benson, T. O'Donovan, P. O'Sullivan, U. Roedig, and C. Screenan. Car Park Management using Wireless Sensor Networks.*In Proc. of the 1st IEEE Int'l. Workshop on Practical Issues in Building Sensor Network Applications (SENSEAPP2006),* November, 2006

[14] M.Castro, P.Druschel, A.M. Kermarrec and A. Rowston.Scribe: A large-scale and decentralized application level multicast infrastructure. *IEEE Journal on Selected Areas in Communications,* 20-6, October, 2002