# A MIDDLEWARE ARCHITECTURE FOR DYNAMIC RECONFIGURATION OF AGENT SERVICE SPACE IN IN-DOOR LOCATION-AWARE SYSTEM

Tae-Hyon Kim, Hyeong-Gon Jo, Seol-Young Jeong and Soon-Ju Kang
*College of IT, Kyungpook National University, 1370 Sankyuk-dong, Buk-gu, 702-701, Daegu, Korea*

Keywords: Service Coverage Reconfigurable Middleware, Service Agents based Decentralized Architecture, Middleware for Ubiquitous Service.

Abstract: Recently, various ubiquitous services interacting with the surrounding physical environment have been attempted using wireless sensor network (WSN). However, these services have still remained in a centralized service architecture in which it is difficult to support real-time response for a large number of mobile sensor nodes. To address this problem, this paper proposes a dynamic reconfigurable agent space (DRAS), which is a dynamically reconfigurable middleware of service agents. The DRAS is designed for indoor WSN which is collaborating with stationary nodes and mobile nodes simultaneously. In the DRAS, service agents can dynamically expand and contract their service areas according to the location of service consumers. To demonstrate the feasibility and performance of the DRAS middleware, the voting service which is an archetypical service was implemented under the DRAS. The results show that the proposed architecture can support fast response for a large number of mobile nodes by effective network traffic distribution and appropriate service processing.

## 1 INTRODUCTION

Recently, wireless sensor network technology has enabled complex associations between humans and physical objects or between physical and virtual environments. Interaction with the surrounding physical environment and personalized services are increasingly in demand for ubiquitous applications such as mobile asset management (Bardram, 2004) or audience response systems (Lu et al., 2010). However, these services are still implemented today in client/server-based centralized architectures, which have difficulty in supporting services requiring real-time response over a large number of mobile nodes. The reason for this is that messages generated in these services need to be delivered to the centralized server using multi-hop communication, and therefore traffic bottlenecks arise on the paths to the server.

To solve these problems, this paper proposes a dynamic reconfigurable agent space (DRAS), which is a special middleware architecture based on service agents that can be distributed over the service area. The service agents can dynamically expand and contract their service areas according to the location of service consumers. This approach guarantees better service performance with a large number of mobile nodes by effective distribution of network traffic and service processing around the service area.

The main contributions of this paper are as follows:

- We designed and implemented a reconfigurable service agent based middleware. Service agents can be dynamically distributed over the service area according to the request of the service consumers.
- We evaluated the performance of the middleware. We measured the service expansion and the contraction time. Also we evaluated the elapsed time of the voting service in the simulation and the real environment.

The contents of this paper are organized as follows. Section 2 examines the domain description and related research. Section 3 describes the conceptual design of the DRAS, and Section 4 describes the implementation of the DRAS. Section 5 presents the testbed and performance evaluation. Finally, Section 6 summarizes with conclusions.

Kim T., Jo H., Jeong S. and Kang S..
A MIDDLEWARE ARCHITECTURE FOR DYNAMIC RECONFIGURATION OF AGENT SERVICE SPACE IN IN-DOOR LOCATION-AWARE SYSTEM.
DOI: 10.5220/0003813002490253
In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems* (PECCS-2012), pages 249-253
ISBN: 978-989-8565-00-6
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)
249

## 2 DOMAIN DESCRIPTION AND RELATED RESEARCH

### 2.1 Domain Description: A Case Study of U-hospital

Figure 1 shows opportunistic location-based services under dynamic changes of service area in a hospital environment. In this environment, the following assumptions have been made. First, the whole environment can be divided into small spaces like rooms or floors. Such a space is called a unit space and is the basic unit for location awareness. Second, communication devices are divided into stationary nodes and mobile nodes according to their functionality and features. Mobile nodes can be attached to people or medical devices in the form of small tags. A stationary node can be installed in the ceiling or walls in every unit space to provide location reference functionality and a communication access point for mobile nodes.

In this environment, a location-based distributed architecture is more suitable than a traditional centralized one. A centralized architecture has difficulty in supporting services that require real-time response with a large number of sensor nodes. The reason for this is that messages generated in the services need to be delivered to the server using multi-hop communication, and therefore bottlenecks arise on the paths to the server.

In this environment, requests of particular services should be processed by the agent which is in the same location with the service customers (mobile nodes). In figure 1, the tracking service for the wheelchair (mobile node m9) is processed, not by the central server, but by the agents in the S9 and S3 stationary nodes. Due to the location-based distributed agent architecture (based on the location of each mobile node), fast response can be provided despite the large number of mobile nodes. To implement a location-based distributed agent architecture, one possible solution is to have all service agents statically running in all stationary nodes, but this approach is too inefficient to implement. To solve this problem, a location-based reconfiguration architecture was used for the service agents. As shown in figure 1, if necessary, any stationary node can activate any service agent and expand its service area by creating a clone agent (meaning a new instance of the same service agent) to handle location-based service request from mobile nodes. By this, all mobile nodes can freely move anywhere without restriction. Of course, if the mobile nodes' locations are reduced, then the service area also shrinks.

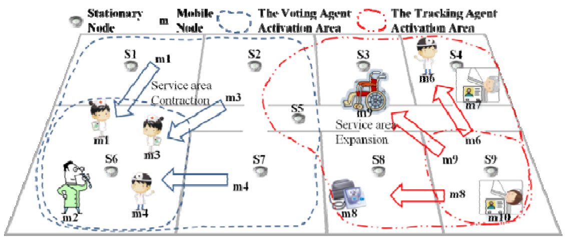

Figure 1: Dynamic changes of service area in a U-hospital.

### 2.2 Related Research

Recently, opportunistic computing or opportunistic networking has become an important concept in the service computing area, driven by the rapid growth of mobile computing and ad hoc networks. The main concept of opportunistic computing, that "when two devices come into contact, it provides a great opportunity to match services to resources, exchange information, cyberforage, execute tasks remotely, and forward messages" (Conti and Kumar, 2010), is highly suitable for location-based services and social network service applications because communication and computing are processed by means of social relationship and collaboration among communication nodes. In this paper, based on the concepts just described, a middleware architecture is proposed which can create communication opportunities and provide exchange services between stationary nodes and mobile nodes which approach each other.

There are many researches related agent based middleware in WSN environment. TeenyLIME (Costa et al., 2006) is a tuplespace-based application middleware which is designed for a WSN environment without base station. TeenyLIME is an extension of LIME. In this paper, a new tuplespace implementation which can provide event-driven asynchronous read/write on top of TinyOS has been provided. Agilla (Fok et al., 2009) is a mobile-agent-based service middleware for a WSN environment. In this middleware, an agent can be moved or copied among mobile nodes without losing its internal contexts. Communications among agents is accomplished by the abstract functions of a tuplespace and an internal neighbor list. However, these studies are also based on multi-hop communications and need to deal with complex calculations on the mobile-node side, and therefore it is difficult to adapt their proposals to provide the services needed for real-time response with a large number of mobile nodes and also for network stability.

## 3 CONCEPTUAL DESIGN OF THE PROPOSED ARCHITECTURE

In the DRAS, a service is provided by service agents which are distributed to stationary nodes in the form of processes and connected to each other by P2P connections. Unlike a server-based centralized architecture, network traffic and service processing are distributed naturally to the multiple stationary nodes. Collaborations with the service agents are a key consideration in the DRAS.

The proposed service architecture uses only one-hop ad hoc communication to communicate between stationary nodes and mobile nodes because of the desire to minimize network congestion and delay problems. To send a message to another mobile node, a mobile node first sends a message to a stationary node. Then the stationary node finds a destination stationary node close to where the destination mobile node is located and sends the message to this stationary node. Finally, the stationary node sends the message to the target mobile node.

Figure 2 shows the proposed network architecture of the DRAS, which is composed of two tiers. The stationary tier is composed of stationary nodes which manage unit spaces, and the backbone network which connects the stationary nodes. The mobile tier is composed of mobile nodes which can be attached to objects or be used for service users, and the one-hop ad hoc network. Communication among stationary nodes is performed in a P2P mode rather than in a traditional server-client mode. A service request coming from a mobile node can be handled by a service agent running in a stationary node. Otherwise, it will be served by collaboration among agents distributed in several stationary nodes (drawn as a service cluster in Figure 2).

To communicate between stationary and mobile nodes, this study used the LIDx/LAMD one-hop ad hoc protocol, which was proposed in a previous study (Kim et al., 2011). LIDx is an abbreviation of "Location-ID Exchange protocol," which is used for location awareness between stationary and mobile nodes. LAMD is an abbreviation of "LIDx-based Asynchronous Message Delivery," which is used for exchanging asynchronous messages among mobile nodes. In this paper, LIDx and LAMD are used as the one-hop-based location determination and message delivery protocols between mobile nodes and stationary nodes to implement the proposed DRAS architecture.

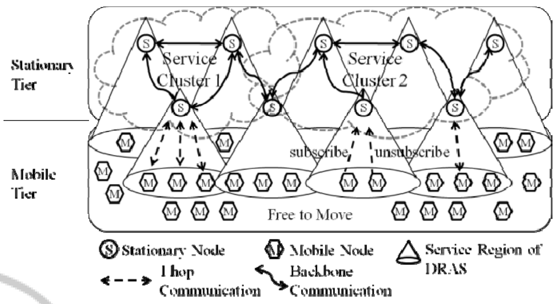

Figure 2: Two-tier service architecture in the DRAS.

A service can dynamically modify its service coverage by changing service agents. Each of the service agents is managed by a process in the DRAS. A service agent can exist in one of the following states: creation, expansion, processing, contraction, and destruction.

## 4 IMPLEMENTATION OF THE DRAS

Figure 3 describes the DRAS software block for the stationary nodes. The proposed middleware which manages the DRAS services can be divided into three components: the service process manager, the repository manager, and the communication manager. The service process manager takes charge of the lifecycles of service agents. The repository manager handles service contexts, including location of mobile nodes and environmental data. The communication manager deals with communication between stationary and mobile nodes. Each of these components contains several processes such as message loops, worker processes, controllers, and others. Message passing is the only way to communicate among processes in this middleware architecture. This means that communication transparency between local and remote processes can be provided throughout the services and that the influence on other processes of the sudden failure of one process can be minimized, even though the response time of the services would be increased by message-handling overhead.

This middleware has been implemented with C++ and Erlang (Armstrong, 2003). Erlang is a programming language that supports fast process creation, control of large numbers of processes, and fast communication among processes for distributed systems. However, hardware control is very limited. Therefore, the service process manager and the repository manager have been implemented in Erlang because they need to control and communicate with many processes. The communication manager has been implemented mainly in C++, but partly in Erlang, because it controls multiple communication protocols for mobile nodes, including hardware access. All messages based on mobile-node communication protocols will be converted to TCP/IP-based messages in the communication manager, and TCP/IP-based messages will be converted before transmission to a mobile protocol depending on the target mobile node.

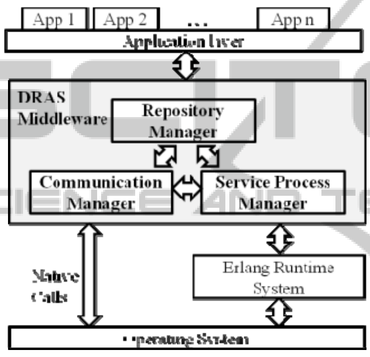

Figure 3: A block diagram of the DRAS middleware.

## 5 TESTBED AND PERFORMANCE EVALUATION

To verify the proposed DRAS middleware and evaluate its performance, a test-bed was set up consisting of five stationary nodes, one PC, and one Ethernet hub. In order to evaluate the performances of the middleware on a large number of mobile nodes, a simulation program was also developed. It can simulate the behavior of mobile nodes, including communication based on LIDx and LAMD (except retry and delay), location movement, and service processing. In the simulation, each Erlang process will simulate the behavior of mobile nodes. Generated messages are routed directly to the stationary node, which is regarded as the current location of the mobile node. Even though TCP/IP rather than the one-hop ad hoc protocol was used for communication between a stationary node and a mobile node, this experiment was intended to evaluate the performance of the services and the middleware itself, which is not limited by the communication medium. In a real environment, the performance could be highly influenced by the limitations of WSN protocols such as traffic congestion or long delay problems. The stationary node is consists of an Arm Cortex 800-MHz A8 MCU with 512MB SDRAM and Linux 2.6.28 kernel. The mobile node has a 8bit MCU with 8KB SRAM and 256KB flash. The PC has an i5-520 CPU with 4 GB memory and Linux 2.6.30 kernel.

Figure 4 shows the evaluation results for the voting service. The response time was evaluated by varying the number of stationary nodes between one and five and of mobile nodes between 10 and 50 in the real environment (Fig.4(a)), between 100 and 1000 in the simulation (Fig.4(c)). The mobile nodes were uniformly distributed over all stationary nodes. This result indicates that the middleware handles traffic distribution well. The decrease in response time with increasing number of stationary nodes is due to the distribution of traffic through multiple stationary nodes with expansion of service coverage. We can observe that the response time was rising significantly when only one stationary node and above 30 mobile nodes were used in the real environment because the traffic of mobile nodes flowing into a stationary node, the traffic was beyond the communication capacity of the ad-hoc node attached to the stationary node.

Service performance was also measured by changes in service-processing overhead and in the number of stationary nodes needed to distribute service processing. The results of this experiment are shown in Fig. 4(b) for the real environment and in Fig. 4(d) for the simulation. To vary service-processing overhead, a series of Fibonacci numbers was just calculated. It takes 32ms to call a function fib(23), 84 ms for fib(25). Similarly to the results for traffic distribution, the performance increases by increasing the number of stationary nodes. Due to the retry and delay problem in the real environment, the result between the real environment and the simulation has differences (retry delay is 100ms in the real environment). However we can recognize the common tendency of the result by the changes of the number of the stationary nodes or mobile nodes.

Table 1: The elapsed time for changing service coverage.

| The Number of Stationary nodes (ea) | 1->2 | 2->3 | 3->4 | 4->5 |
|---|---|---|---|---|
| Expansion Time(ms) | 29.4 | 28.0 | 29.5 | 29.0 |
| - | 5->4 | 4->3 | 3->2 | 2->1 |
| Contraction Time(ms) | 12.8 | 11.7 | 11.9 | 11.9 |

Table 1 shows the elapsed time for the dynamic change of service coverage. It shows that the change

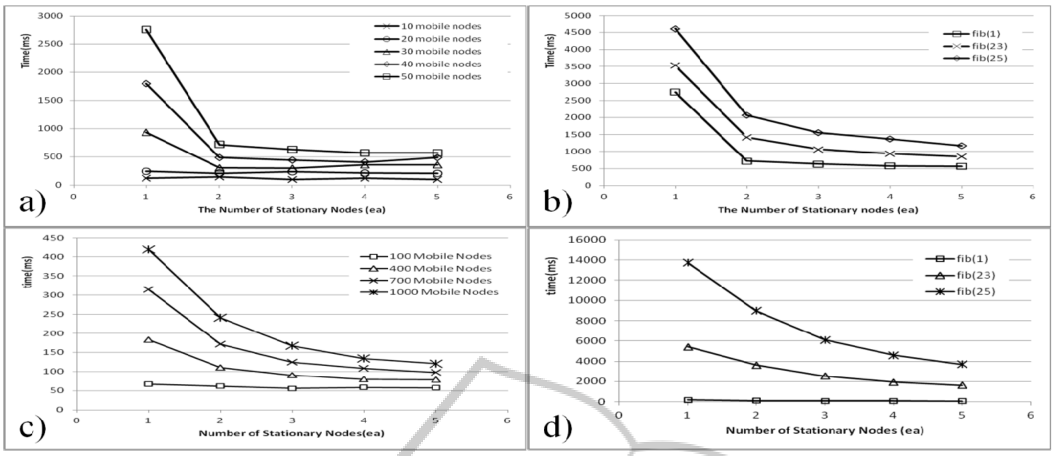

Figure 4: Test results for the voting service: a, c) Variation in response time with number of stationary and mobile nodes (average of 10 runs); b, d) Variation in response time with processing overhead and the number of stationary nodes (with 50 mobile nodes for b, 400 mobile nodes for d, average of 10 runs). a, b) the real environment. c, d) simulation.

of service coverage takes under 30 ms for expansion and under 13ms for contraction.

## 6 CONCLUSIONS

In this paper, a dynamic reconfigurable agent space (DRAS) has been proposed, which is a special middleware architecture based on service agents. The agents can be distributed over an actual service area in the form of processes in the stationary nodes and can provide real-time response by distribution of network traffic and service processing. The DRAS can expand and contract the service area dynamically by generation and destruction of service agents in the stationary nodes around the mobile node.

To verify the DRAS middleware and to evaluate its performance, the voting service was implemented. The response time was evaluated in the real environment and in the simulation. We showed that the traffic and the processing were well distributed over the multiple stationary nodes.

We are developing the DRAS continuously and applying it to self-organizing applications such as robot swarms which need fast response and complex collaboration among communication nodes. It can be expected that the characteristics of the proposed DRAS, such as effective traffic distribution and service processing and dynamic service-coverage reconfiguration, can be of great assistance in that service domain.

## ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0003387)

## REFERENCES

Armstrong, J., 2003. Making Reliable Distributed Systems In The Presence Of Software Errors. Phd Thesis, *The Royal Institute Of Technology.*

Bardram, J. E., 2004. Applications Of Context-Aware Computing In Hospital Work: Examples And Design Principles. *Proceedings Of The 2004 Acm Symposium On Applied Computing.*

Conti, M. and Kumar, M., 2010. Opportunities In Opportunistic Computing. *Computer,* 43**,** 42-50.

Costa, P., Mottola, L., Murphy, A. L. and Picco, G. P. 2006. Teenylime: Transiently Shared Tuple Space Middleware For Wireless Sensor Networks. *Proceedings of the International Workshop on Middleware for Sensor Networks.*

Fok, C. L., Roman, G. C. and Lu, C., 2009. Agilla: A Mobile Agent Middleware For Self-Adaptive Wireless Sensor Networks. *Acm Trans. Auton. Adapt. Syst.*

Kim, T., Jeong, S., Cho, H., Kang, S. and Lee, J., 2011. A Location-Aware Asynchronous Message Delivery For Indoor Wireless Sensor Network Applications. *Acis/Jnu International Conference On Computers, Networks, Systems, And Industrial Engineering.*

Lu, J., Pein, R. P., Hansen, G., Nielsen, K. L. and Stav, J. B. 2010. User Centred Mobile Aided Learning System: Student Response System (Srs). *Computer And Information Technology (Cit).*