유비쿼터스 헬스케어 시스템에서 이동에이전트 기반 균형화 클러스터링☆

Balanced Clustering based on Mobile Agents for the Ubiquitous Healthcare Systems

마테오 로미오* 이 재 완** 이 말 레***
Romeo Mark A. Mateo Jaewan Lee Malrey Lee

요 약

유비쿼터스 헬스케어에서 지능형 의사결정지원 및 빠른 진단결과를 제공하기 위한 자동진단은 일반적으로 에이전트 시스템에 의해 수행된다. 본 연구에서는 이동에이전트기술을 사용하여 저 부하 노드에 효율적으로 프로세스를 이주시켜 부하를 분산시키도록 유비쿼터스 헬스케어시스템을 설계하였다. 또한 실시간 자동진단시스템을 지원하는 이동에이전트 중심의 유비쿼터스 헬스케어 기술을 위한 프레임워크를 제시하며, 효율적인 자원활용을 고려하여, 노드들 내에 있는 프로세스의 부하분산을 위한 균형화된 클러스터링을 제안한다. 제안한 알고리즘은 시스템의 부하분산이 최소화될 때까지 과부하된 노드를 선택하여 프로세스를 가까운 노드에 이주시킨다. 제안한 균형화 클러스터링은, 가까운 노드에 이주시킴으로써 메시지오버헤드를 감안할 때, 효율적으로 프로세스를 모든 노드에 분산시킨다.

ABSTRACT

In the ubiquitous healthcare, automated diagnosis is commonly achieved by an agent system to provide intelligent decision support and fast diagnosis result. Mobile agent technology is used for efficient load distribution by migrating processes to a less loaded node which is considered in our design of a ubiquitous healthcare system. This paper presents a framework for ubiquitous healthcare technologies which mainly focuses on mobile agents that serve the on-demand processes of an automated diagnosis support system. Considering the efficient utilization of resources, a balanced clustering for the load distribution of processes within nodes is proposed. The proposed algorithm selects overloaded nodes to migrate processes to near nodes until the load variance of the system is minimized. Our proposed balanced clustering efficiently distributes processes to all nodes considering message overheads by performing the migration to the near nodes.

☞ KeyWords : 클러스터링, 부하균형,유비쿼터스 헬스케어, 이동에이전트, Clustering, load balancing, ubiquitous healthcare, mobile agents

1. Introduction

Ubiquitous healthcare system is a popular research to apply wireless sensors and intelligent agents for real time monitoring of patients [1] and accurate diagnosis for decision support [2], respectively. Software agent guided by rules in performing a task is another consideration in ubiquitous computing. Agent technology is mostly used to automate tasks in ubiquitous environment [3]. In designing a system in ubiquitous environment, we must identify the

* 준 회 원 : 군산대학교 전자정보공학부 박사과정
rmmateo@kunsan.ac.kr
** 종신회원 : 군산대학교 정보통신공학과 교수
jwlee@kunsan.ac.kr (교신저자)
*** 정 회 원 : 전북대학교 영상정보신기술 연구 센터, 컴퓨터공학부 교수
mrlee@chonbuk.ac.kr
[2009/05/27 투고 - 2009/06/19 심사(2009/10/19 2차) - 2009/12/29 심사완료]
☆ This research was supported by grant R01-2006-000-10147-0 from the Basic Research Program of the Korea Science and Engineering Foundation

constraints like limitation of resources. Processing a large amount of information or adding additional features can be also limited. Another challenge that needs lots of considerations is designing a transparent ubiquitous system. These limitation is tackled in some researches [4,5] by their proposed middleware for mobile and ubiquitous environment. This approach provides an abstraction between applications and underlying network infrastructure.

Quality of service is mostly obtained by providing fast responses on requests and this can be achieved by replicating the services. In replication schemes, the client requests are distributed to replicates which minimize queues of requests providing a faster response. The cloning and migration scheme of mobile agents [6] have same concept as replication. Like replication, mobile agent technology provides quality of service by cloning agents but agents can move to a different node to process its task. This technique is also used in load balancing [7,8]. Most of the researches focus on the mechanism triggering the migration based on thresholds that indicates overloading and finds another least loaded node to deploy the agent. Having no knowledge about the network topology, a mobile agent can be deployed to a far node which produces message overhead in communication. The migration of mobile agents that considers the link communication from node source of agent to its destination node is significant in providing efficient load distribution and QoS to clients.

This paper presents a design of a ubiquitous and intelligent middleware that mainly supports the transparency of interaction of components in ubiquitous computing. Autonomous agents are included in the middleware framework to represent the intelligent application components. The framework is applied for the ubiquitous healthcare for senior citizens to monitor health status and to provide necessary healthcare services. The cloning scheme is done by healthcare agents for the QoS of the system. To distribute efficiently these mobile agents within the nodes, a balanced clustering is proposed. The balanced clustering minimizes the variation of loads in the system by deploying agents from loaded node to its near nodes that have fewer loads. We compared our approach to the common load balancing schemes and the proposed algorithm was more efficient in distributing loads.

2. Ubiquitous Middleware

Ubiquitous computing concepts have emerged not only on providing information wirelessly but also automate services based on the context information from user profiles and the environment. Current middleware technologies for mobile and ubiquitous support are limited in providing transparency of services to mobile users which are tackled by some researches [4, 5]. The middleware for mobile computing based on mobile agents (MA) is proposed by Belevista et. al [4]. With the use of autonomous mobile agent, users can access services even if a terminal disconnected because an agent delivers the results upon reconnection. The HOMEROS is proposed [5] which allow high flexibility by adopting a hybrid-network model and dynamically configurable reflective object request broker (ORB) to provide flexibility on wireless applications. In our proposed middleware, agent mobility and scalability to integrate additional components is considered.

3. Ubiquitous Healthcare using Intelligent Distributed Framework

The target application of our proposed middleware is the ubiquitous healthcare for senior citizens. The proposed middleware supports the specifications of devices like sensors that are implemented in the spontaneous network. The ubiquitous environment specifications are different from a classical distributed environment. In our proposed middleware, mobility of programs and flexible on adding components are supported shown in Figure 1. It also supports agent protocols for the execution of application and efficient services interaction to serve agents through their tasks in a transparent manner.

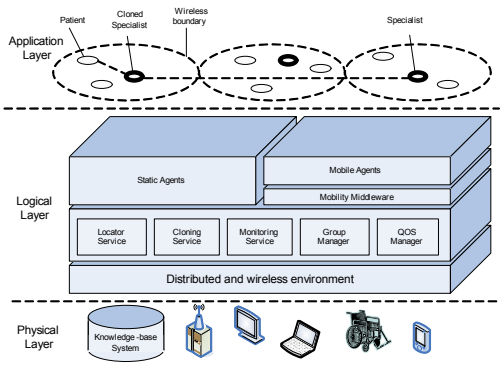


Fig. 1 Ubiquitous and intelligent framework is a middleware that implements distributed task by agent technology and gathering by sensor devices. Services specialized for the ubiquitous devices supports the system communication.

The framework design is divided into three layers. First, application layer is consisted of agents that apply the role of a healthcare system. Specialists or physician agents monitors an individual through its patient agent. A patient agent contains information about the individual it represents. Moreover, applications layer considers the requirement of logical layer. Logical layer serves between the application layer and physical layer, and its executions are transparent to those layers. Physical layer consists of different hardware like mobile devices, ubiquitous wheel chairs, computers and others. Services from the logical layer are the following:

Locator service – locates the appropriate physician to monitor a patient using mobile devices. The classification method is used to choose the appropriate physician agents for a patient. This also coordinates with the monitoring service in performing the proposed balanced clustering.

Cloning service – performs the cloning of agents and deployment of mobile agents through base stations. The cloning service communicates to locator for request of service.

Monitoring service – monitors the activities of every services and agents. All events are monitored and recorded by the monitoring service.

Group manager – manages the grouping of agents on all base stations. The grouping manager decides the migration based on the proposed balanced clustering.

QOS manager – implements the efficient coordination of the services to perform quality of service.

4. Balanced Clustering using Mobile Agents

Agents in our framework are components of a healthcare system, specifically, physicians and patients. In our previous work [9], an expert mobile agent (EMA) classifies data of a patient using the neuro-fuzzy algorithm. Similarly, we followed the

design patterns from the previous work. The physician agent is used to monitor and diagnose an illness of a patient while the patient agent representing a real patient provides necessary data to physician agent. Moreover, the physician agents are cloned for faster response and increase throughputs. In our scheme, we assumed the number of clones ($A= \{a_1, a_2, \cdots, a_n\}$) is equal to number of request ($Q = \{ q_1, q_2, \cdots, q_n \}$ *where* $A=Q$).

4.1 Balanced Clustering

Classical clustering methods [10 11, 12] perform grouping based on the nearest property of a given value or providing center values of a group. A typical objective function is shown in Equation 1 where the main parameters are vectors u_k and cluster centers c_i.

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k=1, \mathbf{u}_k \in C_i} m_{ik} \left\| \mathbf{u}_k - \mathbf{c}_i \right\|^2 \right) \quad (1)$$

The J_i is minimized by several iterations and stops if either the improvement over the previous iteration is below a certain tolerance or J_i is below a certain threshold value. In our proposed framework, a clustering technique that balances current processes on each node or base station using mobile agents to maximize the contribution of all nodes within the process. The total loads of each base station must be approximately equal to other base stations. In the proposed framework, a static network topology of base stations and several agents that can migrate to base stations are assumed. Each agent has its original source node that is basis of cloning. The setting of an agent is mainly configured by a physician but the clone's iterinary is processed by the proposed clustering method. To provide the quality of service within the system, agents in different base station are deployed considering the least number of links that can delay the communication to clients and original source of agent. We assumed that the network topology is stationary and input variables are moving clients and agents migrating to another base station. The mean and standard deviation is used to determine the current load distribution from all base stations ($N=\{n1, n2, \cdots ni\}$). The loads of a single base station are calculated by summing up all agents with its processing time (x) that are currently in base station. Equation 2 shows the mean of all processing time in a node.

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n \quad (2)$$

Equation 3 is the load variance based on the standard deviation of total loads where μ is the mean load and x is the load of a single base station. We process all load values in Equation 3 and get the value of σ for the load analysis to be used in agent migration.

$$\sigma = \frac{1}{N} \sum_{n=1}^{N} \left\| x_n - \mu \right\| \quad (3)$$

We determine the minimum value of σ to terminate the process which means that it provided the optimal value of the process. In the first run, the algorithm finds the highest load from the base station and determines the candidate agents needed to be deployed. The selected highest loaded base station deploys its excess agents to the next neighbor base stations. The overload is calculated by subtracting current load of base station to mean load. To determine the agents which needed to be

deployed, excess agents are determined by selecting the agent to be collected in *D*, while total loads from *D* is approximately equal to overload. The collection of these agents to be deployed is shown in Equation 4.

$$D = \sum_{n=0}^{N} candidate(a_n) \tag{4}$$

The function *candidate()* collects the index of an agent until the loads from D is approximately equal to the overload. After determining the candidate agents, the proposed algorithm selects neighboring base station to deploy the agents. The selection of candidate neighbor base station is determined by degree links (*d*). We set the degree of link before processing deployment. If the load is less than mean load then nearest node with available resources are processed by deployment first. The deployment of agents is shown in Equation 5 where procedure includes the looping of candidate agents to be deployed and all neighboring base station. The number of neighboring base station is indicated by *L*. The f(*x*) is a function to determine if the base station has available loads while operating the loop of deployment.

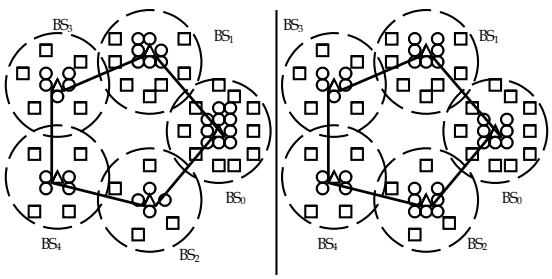
$$J = \sum_{l=linkindex}^{L} \sum_{agentindex}^{D} f(x_a) \tag{5}$$

Equation 6 represents the function of determining the available resources from the base station indicated by *aload*. If aload < 0 then it migrates a candidate agent from *A* and subtract the load of that agent to current load or aload.

$$f(x_a) = \begin{cases} aload - agentload \;\&\; migrate(x_a, bs_l), if\ aload < 0 \\ 0, else \end{cases} \tag{6}$$

Deployment procedure from Equation 5 will continue if all agents from A are not deployed. If not all agents are deployed within its first degree neighbors, the next degree neighbors is used to process the Equation 5. The degree neighbor is determined by the hop links from source node to destination of deployment. This method continues until all candidate agents are deployed and it reaches maximum degree neighbor represented by *d*. After deploying all candidate agents, the algorithm compares again the distribution of loads. If σ is minimal then the process is terminated. The summary steps of the algorithm are shown:

1. Determine the number of agent clones base on the number of request.
2. Calculate the load distribution that includes total loads, mean and standard deviation.
3. Base on step 2, determine the loaded node and candidate agents needed to deploy.
4. Deploy the candidate agents to nearest neighbor links.
5. If candidate agents are not all deployed then use the next degree links of loaded node.
6. Calculate load distribution in step 2, if minimum then stop, if not then continue the deployment process.



(A) (B)

Fig. 2 Graphical result of balanced clustering. In *A* represents the initial deployment of mobile agents and *B* is the final deployment.

Figure 2 shows a graphical illustration of deploying mobile agents in base stations during the process of proposed algorithm. In Figure 2A shows a static network topology of base station with 5 nodes and having the corresponding links: bs0★bs1, bs0★bs2, bs1★bs3, bs2★bs4 and bs3★bs4. There are 30 agents and initially distributed within base stations: bs0=10, bs1=7, bs2=4, bs3=5 and bs4=4. These agents are determined by number of request in the system to perform cloning and migration. There are 5 classes of physician agents and also differ in load. Figure 2B is the final distribution of mobile agents to all base station. Moreover, the migration of an agent chooses the nearest neighbor to prevent delay communication to its source node. In Figure 3, we show the pseudo codes of the algorithm.

```
N collection of all nodes, d maximum link degree
for each request do
        CloneAgent(request)
end for
function LoadDistribution()
X→loads(N)
m→Average(X)
std→StandardDev()
end function
while not min(std) do
loaded→CandidateNode(N)
agents→CandidateAgents(loaded), a ∈ agents
DeployAgents(loaded, agents, degree)
        LoadDistribution()
end while
function DeployAgents(node, agents, degree)
L→neighbors(node), degree++
for each l ∈ L where x→load(l) do
 if x < m then
    while aload < 0 do
                migrate(a, l), next a
                aload→TotalLoad(l)
    end while
          end if
         end for
         if agents is not equal to 0 and degree is
        not equal to d then
    for each l ∈ L
                DeployAgents(l, agents, degree)
                     end for
          end if
end function
```

Fig. 3 Pseudo code of the proposed balanced clustering.

Figure 3 shows the pseudo-code of the proposed balanced clustering. The load analysis codes is stated at *LoadDistribution()* function while deployment of agents using the values on load analysis is stated at *DeployAgent()* function. In executing agent deployments, we need to provide a constant value on maximum link degree. Supposedly, we found out that α is the appropriate value for the maximum link degree and increasing d will not affect the agent migration because it always terminates at α . Setting d >α provides a low load variation because it may deploy all overloads from the loaded node to its neighboring nodes and thus lowering the value of σ . However, if an agent was deployed far from the original node then it needs several hops to communicate from deployed node. Setting d <α will limit hop counts but may stop while the overload from the loaded node is not well distribute which is a tradeoff.

5. Experimental Evaluation

The proposed system is simulated using Jade Framework, a Java implementation of multi-agent system, where the components and mobile agents were designed. We used 10 computers, running in Pentium 4 processors, to host agents and serve as base stations. Wireless sensors and embedded devices used for ubiquitous healthcare in a local wireless area are managed by these base stations. Middleware services are implemented in Java codes and integrated to Jade to acquire the functionality of different services in our proposed framework. Mobile agents are used in load balancing techniques [7,8]. To efficiently distribute the load processes of agents in the framework, we used our proposed balanced clustering. Moreover, the load distribution performance of proposed balanced clustering is

compared to least load selection, random load distribution and no distribution method.

5.1 Environment

We configured 10 base stations (bs) and having the corresponding links: bs0↔(bs1, bs2, bs3), bs1↔(bs0, bs4, bs5), bs2↔(bs0, bs6, bs7), bs3↔(bs0, bs8, bs9), bs4↔(bs1, bs5, bs6), bs5↔(bs1, bs4, bs9), bs6↔(bs2, bs4, bs7), bs7↔(bs2, bs6, bs8), bs8↔(bs3, bs7, bs9) and bs9↔(bs3, bs5, bs8). We assumed the initial distribution of mobile agents is based on the number of requests (Q=100) within the base station: bs0=20, bs1=8, bs2=9, bs3=10, bs4=8, bs5=2, bs6=23, bs7=7, bs8=10, and bs9=3. There are 10 classes of physician agents and differ in processing time of a request: a0=2000ms, a1=3000ms, a2=2000ms, a3=3000ms, a4=1000ms, a5=2000ms, a6=3000ms, a7=2000ms, a8=3000ms, and a9=1000ms. Using these parameters, the performance of our proposed algorithm and other traditional algorithms [13] in balancing method was compared.

5.2 Simulation Result

The mean represents an ideal value of a load in each base station while the standard deviation evaluates distribution performance of a load balancing technique. We get the proportion (r) of standard deviation to its mean value shown in Equation 7 where the value closer to 0 means that the distribution technique is efficient. Number of hops or communication overhead (λ) produced by the algorithm is also evaluated. This is the number of agent's hops in moving to another base station where the original base station or source node stores data. It is assumed that after deployment of an agent to another base station, it still needs to communicate through its source node and thus causes message overheads. The comparison of algorithms using r and λ is shown in Table 1 where A is the number of agents deployed in a node and L is the total loads.

$$r = \frac{\sigma}{\mu} \tag{7}$$

The least load selection scheme distributes excess load processes from highest loaded node and migrate these to the least loaded node. Like our proposed algorithm, this technique provides a delay cost in analyzing the loads and deciding the migration of agents. While this procedure can distribute resources efficiently, it does not consider the communication overheads. This overhead is determined by the hop links when an agent has migrated to another host. In the random distribution, migration procedure is done randomly which has no delay cost in analyzing but can have a high load variations. The no distribution method is also compared which has no cost in analyzing loads and communication overhead in migration but has the highest r result compared to all algorithm in performing load distribution. Using a 100 request which will generate 100 clones, the result from Table 1 shows that our algorithm is better of 19 hops compared to the least load selection which is also a factor in having fast response time. Also, it shows a good load distribution compared to random and no distribution methods. Our proposed balanced clustering distributes efficiently loads and considers minimal hops. The number of hops to communicate from deployed agents is less because it chooses the nearest base station in processing migration unlike in the least load selection scheme.

Table 1. Load distribution performance of balanced -clustering, least load selection, random distribution and no balancing method

BS	Balanced clustering		Least load selection		Random distribution		No distribution method	
	A/L(ms)	λ	A/L(ms)	λ	A/L(ms)	λ	A/L(ms)	λ
0	13/ 26000	18	13/ 26000	21	5/ 10000	30	20/ 40000	0
1	8/ 24000	0	8/ 24000	0	14/ 36000	9	8/ 24000	0
2	11/ 25000	3	12/ 21000	0	17/ 46000	13	9/ 18000	0
3	9/ 27000	10	9/ 27000	3	8/ 23000	9	10/ 30000	0
4	14/ 25000	0	13/ 25000	6	8/ 19000	16	8/ 8000	0
5	9/ 25000	0	9/ 24000	0	18/ 44000	2	2/ 4000	0
6	9/ 27000	19	9/ 27000	30	9/ 21000	36	23/ 69000	0
7	11/ 26000	0	9/ 20000	0	4/ 9000	6	7/ 14000	0
8	9/ 27000	1	9/ 27000	1	11/ 26000	5	10/ 30000	0
9	7/ 14000	0	9/ 25000	0	6/ 12000	2	3/ 3000	0
	r=0.091	42	r=0.076	61	r=0.44	128	r=0.57	0

In Table 1, only 100 clients were shown because more than 100 clients produce a very small value of r using the proposed algorithm and it is enough to use for comparison. Figure 4 shows the message overhead that is produced by number of hops of deploying agent from its original node to destination node for balanced clustering (BC), least load selection (LSS), and random distribution (RD). The number of hop links is generated by requests and as the number of requests increases, also the number of hops increases. The graph shows that BC is better of 54 hops in average than the LLS and outperformed the RD. The result concludes that our algorithm provides a low communication overhead.

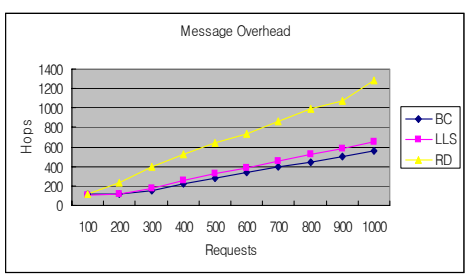


Fig. 4 Communication overhead determined by number of agent hops generated by requests.

6. Conclusion and Future Work

This paper presents a middleware for ubiquitous systems that considers efficient load distribution using mobile agents. The framework was implemented in the ubiquitous healthcare environment where the efficient interaction of agents provides necessary healthcare services for senior citizens. For fast diagnosis, a cloning scheme for physician agents was implemented. The proposed balanced clustering was used to balance the distribution of processes within the base stations. The balanced clustering minimized the system load variation by deploying processes from overloaded base stations to less loaded base stations. Experiment evaluation showed that the proposed balanced clustering was efficient in load distribution and provides low communication overhead compared to common load distribution schemes.

The future research will exploit the method to other fields that implements on-demand processes like cloud computing. Also, the proposed algorithm was only simulated in a fixed network topology. The algorithm will be tested on different types of network topology and the configurations will be optimized as its future work.

References

[1] J. Jung, K. Ha, j. Lee, Y.S. Kim and D. Kim, "Wireless Body Area Network in a Ubiquitous Healthcare System for Physiological Signal Monitoring and Health Consulting", IJISP, Vol. 1, No. 1, pp. 47-54, 2008.

[2] B. L. Iantovics, "Cooperative Medical Diagnosis Elaboration by Physicians and Artificial Agents", Understanding Complex Systems, pp. 315-339, 2009.

[3] M. Rodríguez and J. Favela, “Autonomous Agents to Support Interoperability and Physical Integration in Pervasive Environments”. Proc. of AWIC, pp. 307-317, 2003.

[4] P. Bellavista, A. Corradi and C. Stefanelli, “Mobile Agent Middleware for Mobile Computing”, Computer, Vol. 34, No. 3, pp. 73-81, 2001.

[5] S. W. Han, Y. B. Yoon, H. Y. Youn W. D. Cho, “A New Middleware Architecture for Ubiquitous Computing Environment”, Proc. of STFEUS, pp. 117-121, 2004.

[6] O. Shehory, K. Sycara, P. Chalasani, S. Jha, “Agent Cloning: An Approach to Agent Mobility and Resource Allocation”, IEEE Communications Magazine, Vol. 36, No. 7, pp. 63 – 67, 1998.

[7] H.A. Thant, K.M. San, K.M.L. Tun, T.T. Naing, N. Thein, ″Mobile Agents Based Load Balancing Method for Parallel Applications″. APSITT 2005 Proceedings, pp.77 – 82, 2005.

[8] Y.Yang, Y.Chen, X.Cao1, J.Ju1, ″Load Balancing Using Mobile Agent and a Novel Algorithm for Updating Load Information Partially″, Springer Verlag, LNCS 3619, pp. 1243-1252

[9] R. M. Mateo, L. F. Cervantes, H. K. Yang, and J. W. Lee ″Mobile Agents Using Data Mining for Diagnosis Support in Ubiquitous Healthcare″, Springer Verlag, LNAI 4496, pp. 795-804, 2007.

[10] J. B. MacQueen, ″Some Methods for classification and Analysis of Multivariate Observations″, Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, pp. 281-297, 1967.

[11] M. Berthold and D. J. Hand, “Intelligent Data Analysis, An Introduction”, Springer, 1999.

[12] J. C. Bezdek, “Pattern Recognition with Fuzzy objective Function Algorithms”, New York, Plenum Press, 1981.

[13] O. Othman, C. O’Ryan, and D. C. Schmidt, “The Design and Performance of an Adaptive CORBA Load Balancing Service”, IEEE DS Online, Vol. 2, No. 4, 2001

◐ 저 자 소 개 ◑

Romeo Mark A. Mateo

2004 West Visayas State University, Philippines
BS in Information Technology
2007 Kunsan National University, South Korea
Master of Engineering major in Information and Telecommunications
2007 ~ current Kunsan National University, South Korea
Graduate student in Ph.D course
Research interest : 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
1992 ~ 현재 군산대학교 교수
관심분야 : 분산 시스템, 운영체제,실시간 시스템, 컴퓨터 네트워크, 티미디어 등
E-mail: jwlee@kunsan.ac.kr

이 말 례(Malrey Lee)

1998 년 중앙대학교 컴퓨터공학과 박사
1999~2003: 전남대학교 멀티미디어학과조교수
2003~현재: 전북대학교 전자정보공학부 부교수
관심분야 : 인공지능, 로보틱스, 컴퓨터게임,
멀티미디어, 유비쿼터스 컴퓨팅, 헬스케어응용 등
E-mail: mrlee@chonbuk.ac.kr