Qualitative Mapping of Ambient Intelligence Characteristics to Operating System Features in Smart Environment

Young-yeol Choo
Department of Computer Engineering
Tongmyong University, Busan, Korea

ABSTRACT

The goal of Ambient Intelligence (AmI) is to build a smart environment for users where they are supported in some of their activities by many interaction mechanisms. The diversity of AmI characteristics requires special support from Operating Systems (OSes). In this paper, in order to support a conscious choice of an operating system for any specific AmI application, features requested by AmI systems were characterized and defined considering various applications. Then, characteristics of existing Operating Systems have been investigated in the context of AmI application support to relate their key characteristics to the typical requirements of AmI systems. Qualitative mapping table between AmI characteristics and OS features has been proposed with an illustration of how to use it. As no OS completely covers the range of characteristics required by AmI systems, challenging issues are summarized for the development of a new OS and a product line of OSes.

Keywords: *Ambient Intelligence, OS, Ubiquitous Computing, Sensor Network, Embedded Systems.*

1. INTRODUCTION

Ambient Intelligence (AmI), which is similar to Ubiquitous Computing in United States, is a challenging research launched by the European Commission [1][2]. The features of AmI application are presented in a document entitled Scenarios for Ambient Intelligence in 2010 which was written by the Information Society Technology Advisory Group (ISTAG) for the Fifth European Framework Programmed (1998-2002)[1]. AmI is a summary term for a large variety of different applications. It is characterized by the European Research Consortium for Informatics and Mathematics (ERCIM) [2] as follows (see Fig. 1):

- Intelligent User Interfaces
- Ubiquitous Communication
- Ubiquitous Computing

According to this definition, Ambient Intelligence systems are supposed to provide access to information at any time and anywhere through multi-modal user interfaces. Applications of AmI cover from essential goods of daily life to entertainment, home automation, healthcare, administration of warehouses, transportation, manufacturing industries, and so on. In addition, they can be applied in many different contexts: e.g., aircraft maintenance is highly regulated; some production shop environments are very difficult from a communication point of view; some contexts make it nearly impossible to work with traditional visually-oriented I/O; sometimes absolutely low-power operation is required, etc. Accordingly, technological span to support AmI is very wide: semiconductor, MEMS (Micro Electro Mechanical Systems), wired and/or wireless communication, Computer Graphics, Computer Vision, Software Engineering, sensors, chemistry, etc.

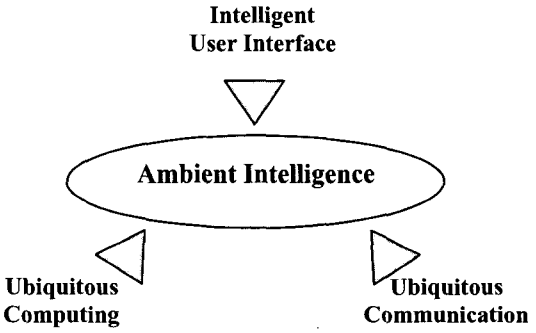


Fig. 1. Technological components of AmI.

Applications of Ambient Intelligence comprise various devices and systems with diverse hardware and software specifications. The goal of AmI may boil down to constructing a *smart environment* in our daily life. In spite of prevalent usage of the words "Ambient Intelligence" or "Ubiquitous Computing", its specific features and requirements are not

* *Corresponding author. E-mail: yychoo@tu.ac.kr*
Manuscript received Sep.15, 2006 ; accepted Sep. 29, 2006

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

specified yet as a whole since its applications are of wide range and each systems are quite unlike in performance. Some devices in the smart environment are highly mobile and have poor capability in terms of computing power and memory size. On the other hand, servers such as inventory server or game server may be immobile and have fast processors and large memory. In addition, communication bandwidth requirements range from few bps (bit per second) to several Giga bps and power consumption requirements from microwatts to watts. Therefore, AmI systems have various requirements and distinct characteristics depending on their applications.

Considering a software aspect of AmI application, each and every AmI systems may be based on an Operating System (OS). The diversity described above will be reflected in a similar diversity of OSes for AmI systems. The general purpose OSes cannot be used as is due to resource restrictions and lack of functionalities for AmI systems.

In this paper, in order to support a conscious choice of an operating system for any specific AmI application, features requested by AmI systems were characterized and defined considering various applications. Then, characteristics of existing Operating Systems have been investigated in the context of AmI application support to relate their key characteristics to the typical requirements of AmI systems. Qualitative mapping table between AmI characteristics and OS features has been proposed with an illustrative example of how to use it. In this analysis, focus is how to support the everyday selection task that developers of AmI systems face. As no OS completely covers the range of characteristics required by AmI systems, challenging issues are summarized for the development of a new OS and a product line of OSes. We understand the selection of the appropriate OS is not a trivial task because of the various requirements of AmI systems and applications.

The remainder of the paper is organized as follows. In chapter 2, I describe the characteristics of AmI systems focusing on the distributed, embedded, mobile, and heterogeneous nature of AmI systems. The relevant operating system features to support AmI systems are presented in chapter 3. In chapter 4, the relation between AmI characteristics and OS attributes is addressed in a mapping table along with an illustration of how to use it. Chapter 5 contains summary of this paper and some remarks on the development of a new OS for AmI applications.

2. AmI ATTRIBUTES

2.1 Small and embedded

Various types of devices such as cameras, head phones, data gloves, special sensors, smart phones, Augmented Reality glass, Active Badge, actuators, etc. participate in the operation of AmI systems to make smart environment [3-10]. The environment may be peppered with a variety of networked sensors. Hence, these devices will adopt MEMS or nano-technology so that they take small space. AmI devices will be placed in remote a location, which causes constraints in power consumption [11]. In these cases, the life time of each AmI system may be determined by the battery life. Consequently, they should be implemented in the form of small size devices using SoC or nano technology. Inevitably, embedded system support is a basic requirement for the development of AmI systems [12].

2.2 Distribution

Inherently, AmI systems are working on the premise that all participating nodes are distributed in the environment and collaborate with others through networks, especially mobile networks. The distribution involves not only hardware and software, but also information and service. Hence, collaboration should be supported on two levels: communication protocol and application service. Especially, a service for a mobile system should be elaborated to dynamically integrate and configure relevant information located at different systems and layers.

2.3 Mobility

The nodes in AmI applications and communication networks are of high mobility [7-11]. When an AmI system moves to be faced with a new ambience, it joins ad-hoc networks and/or wireless sensor networks and exchanges information and service with the nodes of the ambience. Hence, self-configuration and re-configuration functions are important features to permit seamless addition and deletion of a node in a network as well as to cope with its vulnerability to failure.

2.4 Heterogeneity

AmI nodes include numerous types of devices, which pervade the "smart environment" [1]. They may be from various vendors, be based on various platforms including different hardware, operating systems, communication protocols, and application software, and provide different interfaces. Thus, the heterogeneity of AmI nodes should be worked out in all aspects.

2.5 Real-time

The applications for real-time computing include process control, intelligent vehicle highway system, telemedicine systems, telematics, multimedia system, etc. All of the above applications involve AmI implementation. Intensive research has been done on real-time computing [13-15]. Real-time systems can be defined as follows: "A real-time system is a system in which the correctness of the computations depends not only upon the logical correctness of the computation but also on the time at which the result is produced [15]."

AmI applications have various real-time properties to be supported by the system level and the application level. In the case of manufacturing application of AmI, real-time processing is necessary for the following fields:

- Real-time control
- Alarm and event processing
- Real-time communication between control devices

2.6 Proactivity and context-awareness

Proactivity is the ability to act in anticipation of a future problem or need of a user by combining knowledge in system and environment. To make a proper decision based on proactivity, an AmI system continuously tracks human intent and has sufficient knowledge of the context related to the current circumstances (internal state + context information). Context is defined as follows:

"Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user [16]."

A user's context may consist of the following attributes [17][18]:

- Location
- Time
- State of system

Once the current context is seized by an AmI system, it is used to control the environment or adjust the system to the circumstance. According to the detection of context change, an AmI system may behave as follows:

- The AmI system automatically adapts to the new context by changing its activity.
- It reports the new context to the user by various Human Computer Interface (HCI) devices or stores the context for later retrieval.

Besides the contexts mentioned above, personal history, behavior pattern, and physiological state such as body temperature, heart rate, etc. could be contexts requested by a specific application. These might be collected by appropriate peripheral devices.

Examples of proactivity and context-awareness are as follows:

- In the application of "Bicycle Team Training [7]," the AmI system suggests optimal speed, formation pattern, and the position of each cyclist at each timing section based on track profile, speed and direction of wind, physical condition of all cyclists, and so on.
- In the scenario of inter-car communication [9], to guide a user to his/her destination quickly and safely, the AmI system needs to collect and combine data for proper suggestion of the route. The data may be gathered from communication with nearby car as well as traffic information servers and filtered to prevent false suggestions.

2.7 Adaptivity

The information and the services that an AmI system can get from environmental nodes are unstable regarding both quantitative and qualitative aspects. Adaptivity is the ability that an AmI system can adjust its operation without human intervention to cope with the change of circumstances or resources. The resources can be bandwidth in wired/wireless communication, battery power, memory space, Central Processing Unit (CPU) capacity, and so on. To decide resource allocation or future system behavior, the following three strategies can be considered [19]:

- AmI entities guide the application to use less of a scarce resource.
- AmI entities can ask the environment to guarantee a certain level of resources.
- AmI entities suggest a corrective action to the user when a good decision is not possible or the result of a bad decision entails high cost.

In these strategies, the quality of service (QoS) is an important feature to consider. Although an application faces a new circumstance, AmI system should maintain the minimum QoS required to fulfill the mission of the application or reduce gracefully the service quality even in the worst case. QoS of an application needs to be considered in all levels of AmI systems including hardware, OS, middleware, application, etc.

Communication in smart environment includes several adaptivity issues to cope with. The sensors deployed in an environment may fail to operate properly due to an unexpected accident or depletion of battery power. The AmI system should adapt to these failures and continue to provide its service without human intervention such as replacement of devices and maintenance operation. In particular, the following adaptive services are necessary to support seamless communication among AmI devices and/or systems.

- Self-organization of ad-hoc network or sensor network
- Self-healing capability
- Self-management
- Reconfiguration of network when the network components are changed
- QoS support: To satisfy this requirement, AmI systems adapt to the change of network states and resources. This feature is closely related to proactivity.

2.8 Spontaneity

An AmI system interacts with its environment *spontaneously* and *seamlessly* [20]. As mentioned above, when an AmI system is faced with a change of circumstances, it should adapt to the new ambience without human intervention. That is, joining the new network, exchanging information with other nodes, and collaborating with the environment to fulfill the mission of the system should happen without explicit commands. This feature may be closely related to the domain of software quality or application services

2.9 Open Systems

AmI systems may consist of heterogeneous devices, which are connected with various network protocols. Therefore, open system technology is indispensable for the seamless integration and cooperation among AmI devices. I define open systems in three aspects:

- Interoperability: An application collaborates with other applications on local and remote systems.
- Portability: A software/hardware should adapt to other software/hardware so that it will function in a different

computing environment than the one for which it was originally written.
- Connectivity: An application can exchange information with other applications without modification of communication facilities.

2.10 Dependability

To avoid ambiguity, I will use the definition of dependability as follows [21]: "Dependability is the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behavior as it is perceived by its user(s)."

More precisely, a dependable system should satisfy four attributes:
- Availability: to be ready for usage
- Reliability: to continue a service without error
- Safety: to avoid catastrophic consequences
- Security: to prevent unauthorized access and/or handling of information.

Devices in an AmI system have the characteristics relevant to dependability as follows.
- Accurate diagnostics
- Fault tolerance

2.11 Scalability

Scalability is defined as the ability of an AmI application or device (hardware or software) to continue to function well when it (or its context) is changed in size or volume in order to meet a user need.

2.12 Security

Because embedded systems in AmI often rely greatly on cost effectiveness and power efficiency, CPU and other resources are restricted in performance. For example, an 8-bit CPU cannot be enough to encrypt/decrypt data with a long cryptographic key [22]. This is one of the important challenging issues in general embedded system applications as well as in AmI applications. Requested security services in AmI applications are as follows:
- Confidentiality: means that only authorized people/users can see/access protected data.
- Authentication: protection against fabrication
- Availability: protection against denial of service
- Integrity: means that only authorized people/users/processes can modify the data in acceptable ways.

2.13 User Interface

AmI systems, just like conventional systems, involve some sort of interface schemes. Perhaps, the traditional interfaces such as PDAs and mobile phones might be still used. However, AmI interfaces often differ widely from traditional interfaces. Audio equipments, haptic interfaces, or augmented reality can be exploited to access services of an AmI system. Hence, a large range of adaptation is expected for context awareness in an application in terms of user interfaces (UI). The domain of adaptation can be categorized as follows:
- Device: AmI UIs should be able to integrate and adapt to various input/output devices exploiting voice and gestures seamlessly.
- Function: If additional functions become available in a dynamic system, the UI should be able to represent them and to effectively accommodate them as well.
- Context: If the brightness of a light or the volume of a sound change, the interface should adapt itself to the changes.
- User: Depending on the skills and preferences of the user, the interface should be self-adapting as well.

3. OS CHARACTERISTICS

Nowadays, there is no common definition of the architecture of how an OS should look like. Depending on the specific application from an embedded system to a main frame server, each OS vendor has its own architecture, and more over, it has its own definition of what makes up an OS [23]. However, there are many common modules and features offered in an OS. In the following sub-sections, the most important ones are listed with some definitions.

3.1 Real-time

A real-time computing environment is an environment where (some) task are executed (completed) within a predefined time deadline from start to completion. As stated in Chapter 2, the real-time property exists in two formats [24]:
- Hard real-time property: hard real-time property is said about deterministic systems where tasks are fulfilling the real-time requirement all the time.
- Soft real-time property: the soft real-time property means that the system is tolerant of missing some of the timing requirements. Probabilistic approaches mostly belong to this category.

3.2 Scalability

The concept of scalability in an OS is the same as the one described at Section 2.11.

3.3 User Interfaces (UIs)

UIs are interaction methods with users supported by an OS. The most notable ones are voice interfaces, graphical user interfaces in all varieties. Recently, robotic interfaces or input via gestures are studied intensively. In this paper, UIs of an OS are classified into three categories:
- Graphical User Interface (GUI) for the OS
- Voice recognition facilities
- I/Os

3.4 Modularity

Modularity supports the possibility of being well customized for particular application in terms of volume and function of an OS. A modular OS may allow the use of modules that are only needed for this kind of applications such as no mouse, no keyboard and so on.

3.5 Resources

Platforms on which OSes run are characterized by a big diversity in terms of resources. The resources may differ in the CPU architecture; they may use different memory technologies, etc. In this paper, I will make distinction between four categories of platforms in terms of resources:

- Tiny: this category contains those systems with CPUs using less then 8 bits, and memories with ranging from few bytes to few Kilo-bytes.
- Small: this category contains those systems with CPUs using from 8 bits to 16 bits, and memory ranging from few kilo bytes to few mega bytes.
- Medium: this category contains those systems with CPUs using from 16 bits to 32 bits, and memory ranging from few mega bytes to hundreds of mega bytes.
- Large: this category contains those systems with CPUs using more than 32 bits, and memory with more than hundred of mega bytes.

3.6 Protection and security

Security in computing can be defined as guaranteeing the properties such as confidentiality, integrity, authentication, and availability.

3.7 Reliability

In OS, reliability is the fact that the OS operates consistently with its specification [25]. In theory, a reliable OS is totally free from technical errors; in practice, vendors express this by a percentage, and weight affected to OSes' tasks/components.

3.8 Open systems

This characteristics shares its definition with the one described at Section 2.9.

3.9 POSIX compliant

Portable Operating System Interface for UNIX (POSIX) is a standard from the IEEE and ISO communities that defines an interface between programs and OSes [26]. Most OS vendors are trying to certify their OSes as compliant with the POSIX specification.

3.10 Multi-tasking

Multi-tasking is the ability to enable the CPU in a computing environment to execute more than one task at a time. This requires OS to implement a process scheduling mechanism.

3.11 Distribution

Distribution in operating systems is the concept of interconnecting many computing nodes through communication networks to be presented to the OS users as one computing system [23]. In fact, distribution can take many formats; here is a list of distribution forms with a few examples:

- **Distributed processing:** Distributed processing is the ability to distribute processes over many processing nodes in a way that is transparent to users [27]. In this paper, the SUN RPC (Remote Procedure Call) is stated as an example. In order to implement system distribution in processing, many features are required. These include distributed mutual exclusion, clock synchronization, distributed deadlock detection, and agreement protocols.

- **Distributed file systems**: Distributed file systems mean the ability to implement a common file system that can be shared by all computing nodes in the distributed system [27]. The Network File System (NFS) is an example. Many proprieties such as distributed mounting, distributed caching, encryption (for security proposes), availability, and consistency are required to implement distributed file systems.

- **Distributed shared memory:** Distributed shared memory means the ability to use some/all of the memories present in computing nodes in a distributed system, as one logical memory. Examples include IVY (Integrated Shared Memory at Yale), which was implemented in the Apollo DOMAIN environment, Mirage, which was implemented at the University of California Los Angles, and Clouds, developed at the Georgia Institute of Technology. This property is supported by many other concepts like granularity, paging algorithms, etc. (for details, refer to [27]).

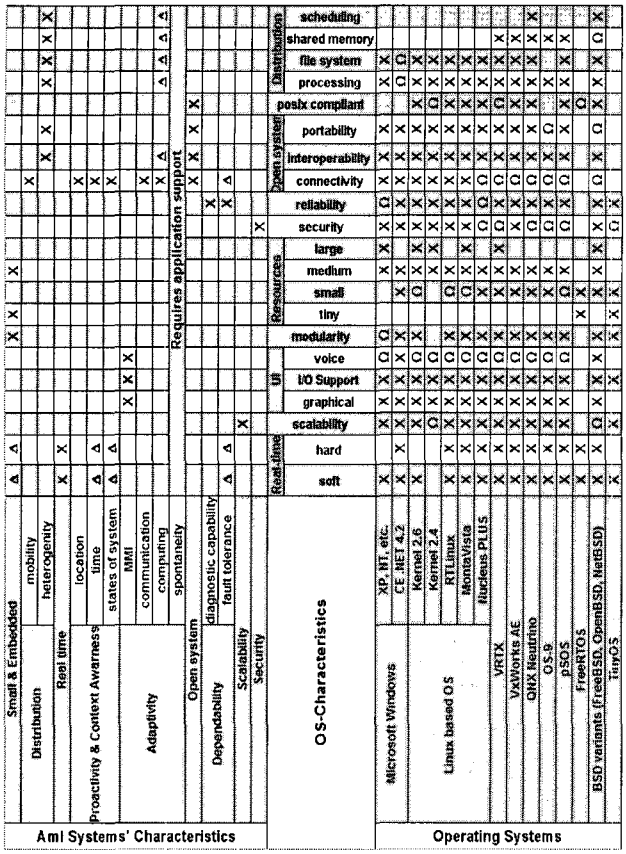
- **Distributed scheduling:** Distributed scheduling deals with process scheduling in distributed systems. It deals with load balancing, load sharing, task migration, etc. In distribution, many other subjects are of relevance, such as fault tolerance, security, failure recovery, and concurrency control. In theory, in order to classify an OS as a distributed OS, all aspects stated should be fulfilled. However, only some of these features are present in practice.

4. A MAPPING TABLE BETWEEN AML ATTRIBUTES AND OS CHARCTERISTICS

The relationship between the AmI characteristics and the OS characteristics is not evident. More over, commercial OSes are provided with only a sub-set of the characteristics described in Chapter 3. In order to understand how we can select which OS to use in an AmI system, it is needed to analyze the relationship between the AmI- and the OS- relevant characteristics. Based

on this analysis, the existing relationship between OS and AmI characteristics is presented in Table 1.

Table 1. Mapping table between AmI characteristics and OS features



The table also provides a set of commercial OSes and their characteristics. The meanings of notations in Table 1 are as follows:

- X: means that there is strong correlation between the corresponding attributes in the table or an attribute is supported by the corresponding OS.
- Δ: means that there is weak correlation between an AmI characteristic and the corresponding OS characteristic.
- Ω: means that an OS does not fully support the corresponding OS characteristic.

To illustrate how Table 1 can be used to select an OS based on the AmI characteristics, an example is provided as follows: The HomeCare company wants to engineer a node of an AmI system in the assisted living domain. The analysis of the node reveals the following characteristics: it is small and embedded, mobile, has to be aware of the position, and its communication should be adapted depending on energy consumption. At first, the person in charge of selecting an appropriate OS identifies the respective rows in the upper part of the (rotated) table. In each of these lines, he identifies the X or Δ s and follows the columns down to the corresponding OS characteristics. X means that designated OS characteristic is requested. Δ means that the OS characteristic may be necessary. In this example, real-time property may be of interest; modularity and resources have to be considered. Furthermore the OS has to provide an appropriate connectivity. With the identified OS characteristics, he can then find the listed OSes that meet these requirements.

5. CONCLUSIONS

AmI is a totally new paradigm shift in information technology. AmI applications need holistic approaches consisting of systems and technologies that are sensitive, embedded, distributed, heterogeneous, highly mobile, adaptive, contextualized, transparent, dependable, and intelligent. In this paper, the characteristics of AmI systems and features of operating systems to support them were presented. The relations between two characteristics were summarized in a table along with support of current commercial OSes for AmI characteristics. Since AmI applications cover many areas, we understand that the selection of the appropriate OS to fulfill a variety of AmI characteristics is not a trivial task. Some AmI characteristics might be handled on the application level. However, others need support on the OS level. As a whole, current OSes are inadequate for supporting most AmI systems and applications because those are based on a new technological paradigm. Hence, a new type of operating system architecture is a challenging issue to support AmI systems and applications. The following attributes of an OS are often required for AmI systems:

- Small size and power management: to be used in sensor nodes and resource-constrained systems.
- Modularity and openness: to be ported in various hardware and software architectures.
- Concurrency: to execute high level processing for users such as proactive behavior, and low level input/output functions such as sensing, concurrently.
- Mobility: to support highly mobile ad hoc networks.
- Direct access to hardware: to control power usage directly in applications.

REFERENCES

[1] Paolo Remagnino, Gian Luca Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm," **IEEE Trans. on Systems, Man, and Cybernetics**, Vol. 35, No. 1, pp.1-6, Jan. 2005.
[2] http://cordis.europa.eu/istag-reports.htm
[3] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/chall_in_AmI.pdf
[4] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/
[5] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/
[6] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/HMI_white_paper.pdf, white paper in the area of MMI for the research topic "Ambient Intelligence"
[7] http://agrausch.informatik.uni-kl.de/budapest/szenarien/
[8] http://www.eit.uni-kl.de/ami/en/inhalte.html, "Scenario : Human centered manufacturing"
[9] http://agrausch.informatik.uni-kl.de/budapest/szenarien/
[10] http://agrausch.informatik.uni-kl.de/budapest/szenarien/

[11] Nojeong Heo and Pramod K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," **IEEE Tr. On Systems, Man, and Cybernetics**, Vol. 35, No. 1, pp. 78-92, Jan. 2005
[12] http://www.cordis.lu/ist/istag.htm
[13] Ian Broster, "Flexibility in Dependable real-time system," Ph. D thesis, Univ. of York, Aug. 2003
[14] Hermann Kopetz and Gunther Bauer, "The Time-Triggered Architecture," **Proc. of the IEEE**, Jun. 2002.
[15] Shin K., and Ramanathan P. "Real-time Computing: A New Discipline of Computer Science and Engineering," **Proc. of the IEEE**, Vol. 82, No. 1. Jan. 1994
[16] Guanling Chen and David Kotz, "A Survey of Context-Aware Mobile Computing Research," **Technical Report TR2000-381**, Dartmouth College, Dept. of Computer Science, 2000
[17] M. A. Strimpakou, I. G. Roussaki, M. E. Anagnostou, "A Context Ontology for Pervasive Service Provision," **Proc. of 20th Conf. on Advanced Information Networking and Applications**, Vol. 2, pp. 18-20, April 2006
[18] C. Anagnostopoulos, A. Tsounis, S. Hadjiefthymiades, "Context Management in Pervasive Computing Environments," **Proc. of Int'l Conference on Pervasive Services**, pp. 421 – 424, July 2005
[19] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," **IEEE Personal Communications**, pp. 10-17 Aug. 2001
[20] S. S. Yau and F. Karim, "A Lightweight Middleware Protocol for Ad Hoc Distributed Object Computing in Ubiquitous Computing Environments," **Proc. of 6th Int'l Conference on Object-Oriented Real-Time Distributed Computing**, pp. 172 – 179, May 2003
[21] J. C. Laprie, "Dependability—Basic Concepts and Terminology," Vol. 5 of **Dependable Computing and Fault-tolerant Systems**, Springer-Verlag, IFIP WG 10.4 1992.
[22] Philip Koopman, "Embedded System Security," **IEEE Computer**, Vol. 37, No. 7, pp. 95-97, Jul. 2004
[23] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, **Operating System Concepts**, 7th Edition, John Willy & Son, Reading, Massachusetts, Dec. 2004
[24] Jane W.S.LIU, **Real-Time Systems**, p-28-32; Prentice Hall.
[25] http://whatis.techtarget.com/definition/0,,sid9_gci752461.
[26] http://www.webopedia.com/TERM/P/POSIX.html.
[27] K. Li, "IVY: A shared virtual memory system for parallel computing," **Proc. of Int'l Conference on Parallel Processing**, pp. 94-101, 1988.

Young-yeol Choo
He received a B.S. and an M.S. degree in Control and Instrumentation Engineering from Seoul National University, Seoul, Korea in 1986 and 1988, respectively and a Ph. D. from the Pohang University of Science and Technology, Pohang, Korea, in 2002. From 1988 to 2002, he has worked for posco as a senior researcher. At present, he is an Assistant Professor in the Dept. of Computer Engineering of Tongmyong University, Busan, Korea since 2002. He was invited as a Visiting Scientist by Fraunhofer IESE (Institute of Experimental Software Engineering) from January 2005 to July 2005. His research interests include computer network, Ambient Intelligence, Ubiquitous Sensor Network, real-time systems and network security.