A Context Management Architecture for Large-Scale Smart Environments

Yoosoo Oh, Jonghyun Han, and Woontack Woo, Gwangju Institute of Science and Technology

ABSTRACT

Context-aware architecture collects various context data from heterogeneous sensors and provides an intelligent service by exploiting the collected data. In this article we explain the generalized context-aware software architecture for heterogeneous smart environments. The proposed architecture integrates large-scale contexts from multiple heterogeneous sensors, and makes a semantic decision by fusing and reasoning about the collected contexts. Moreover, we discuss a designed architecture that manages communities between large numbers of heterogeneous information entities and enhances intelligence abilities.

INTRODUCTION

A context-aware architecture has the capability to manage large numbers of distributed heterogeneous information sources in large-scale smart environments. In particular, context-aware architecture collects various contexts from heterogeneous information sources and makes a decision to provide an intelligent service. Large-scale heterogeneous environments are human beings (e.g., wearable/physiological sensing environments), real environments (e.g., smart home/office/car), and virtual environments (e.g., Second Life and Google Earth). To ensure a seamless connection between those distributed environments, we need to have a generalized architecture that manages dynamic situations in large-scale smart environments. In this article we use the widely accepted definition of context in the area of context-aware computing [1], and we explain the context that describes and interprets the situation as it is perceived by the system [2].

There is much prior work related to context-aware architecture/middleware [3–5]. In addition to these architectures, there are many tools [1, 6]. However, developing a generalized context-aware architecture that is applicable to large-scale smart environments is still an issue to be addressed. Actually, related research did not simultaneously use human, real, and virtual environments; they should reconsider the connections among domain-free entities in heterogeneous large-scale environments. Therefore, there is a need for a versatile, generally context-aware architecture that can be applied to the full lifecycle in large-scale smart environments.

Through analysis of the literature, we found the following requirements to be necessary in a generalized context-aware architecture:

- Context flow in architecture should be clear and of minimal complexity in order to efficiently manage a system.
- A seamless relationship/network among heterogeneous context sources (sensors/services) should be formed.
- Architecture should support full lifecycle management of a system, such as adding, deleting, modifying, or varying functionality, capacity, or platform in runtime.
- Contextual data that is used in various context-aware systems should be synchronized.
- Architecture should be easy to use across a wide range of applications.
- Architecture should have social intelligence to manage multiple users' behaviors.

In order to achieve these requirements, we need a generic solution that provides for seamless connection of domain-free entities among users and real/virtual environments. Also, we need to resolve issues concerning how to improve useful intelligence ability in a context-aware architecture. In a large-scale environment, issues such as extracting multiple users' attention or intention and providing personalized services to each individual user in large crowds should be discussed by designing a hierarchical context-aware architecture. Therefore, we should consider how to build such a context-aware architecture for heterogeneous large-scale environments.

In this article we explain a generalized context-aware software architecture for heterogeneous smart environments and their services. The proposed architecture is a hierarchical architecture that collects large-scale contexts from multiple heterogeneous sensors and makes a semantic decision by fusing and reasoning about the collected contexts. Moreover, we develop user-centric community composition and management methods, and we design an architecture that enhances intelligence abilities. Finally, we suggest a service provisioning method that enables easy and rapid development of context-aware services.

0163-6804/10/$25.00 © 2010 IEEE

	SOCAM	JCAF	CAMidO	The proposed architecture
Context fusion and reasoning	Various inferences, real-time rule update	Simple inference, real-time rule update	Adoption rule, real-time rule update	Adoption rule, real-time rule update
Context interpretation	Logical-reasoning-based	Entity-structure-based	Interpretation-policy-based	(Logical) context-interpretation-based
Context representation	Ontology (OWL)-based	Object-oriented	Ontology (OWL)-based	Ontology (5W1H model)-based user-centric approach
Service provisioning	X	X	X	Service provisioning for both developers and users
Community management	X	X	X	Community composition and management
Implemented application	A dining room application (playing music, adjusting lighting conditions)	Context-aware medical application (interactive hospital bed)	An e-commerce shopping application (mobile payment service)	Context-aware media application (content recommendation service)
Domain	Real (or user) env.	Real (or user) env.	Real (or user) env.	Heterogeneous env. (user, real, and virtual env.)

Table 1. *Functional comparison; env: environment.*

Architectures for Context-Aware Application Platforms

A context-aware architecture is an important research issue in the ubiquitous computing field. The architecture- or middleware-related research is as follows: Context-Toolkit [1], SOCAM [3], JCAF middleware [4], CAMidO [5], TEA context acquisition architecture [6], and ubi-UCAM 1.0 [7]. This article describes the recent research activities among these tools.

SOCAM [3] is based on OWL, and its context interpreter provides logical reasoning services and fuses multiple low-level contexts into high-level contexts. Furthermore, it manages context reasoning and knowledge base consistency; for instance, it controls home appliances and reasons about the family member's activity. JCAF [4] is a framework that supports the development of context-aware applications. It is a distributed service-oriented Java framework for public use; it reacts to changes in contextual information and manages changes in context events. For example, JCAF reacts to changes in actuators and deals with environment monitoring. CAMidO [5] is an extension of component-based middleware that has context-aware functions. It collects and analyzes contexts by communicating with sensors and behaves based on the collected contexts. These activities provide context-aware applications based on their context-aware architecture: smart home, healthcare, and shopping applications. To make those applications understandable, it is important to provide information about why the decision is made and what input factors are responsible for the decision. A context-aware architecture should manage comprehensible system decisions.

Table 1 is a functional comparison of the proposed architecture and other related tools concerning the requirements of context-aware architectures stated in the introduction. The related research studies were not designed for large-scale heterogeneous environments; previous activities needed to connect domain-free entities in heterogeneous large-scale environments under unpredictable dynamic situations. Therefore, we should design a generalized context-aware architecture that can be applied to full lifecycle management systems. Moreover, we should improve social intelligence by exploiting a community approach based on semantic representation, interpretation, fusion, and the reasoning of contexts.

An Approach for a UCAM

In this section we present the proposed context-aware architecture, Unified Context-Aware Application Model (UCAM), and explain its definition, features, composition, and functionalities.

Overview

UCAM is a generalized context-aware architecture for seamless human, content, and environment interaction in large-scale ubiquitous computing environments. It seamlessly connects various entities among user, real, and virtual environments. Actually, we modified previous versions of UCAM [7]; our proposed UCAM particularly supports intelligence such as context reasoning, interpretation, and community management. The proposed UCAM architecture uses a context model that describes context in the form of 5W1H (*who*, *what*, *where*, *when*, *why*, and *how*) [2]. The 5W1H context model represents contexts according to each individual user. This context model maintains context history in a context repository. More precisely, the UCAM architecture has a context memory, and the UCAM makes a decision by using the current collected contexts and recorded contexts at each decision interval. The proposed architecture increases the intelligence level of contexts

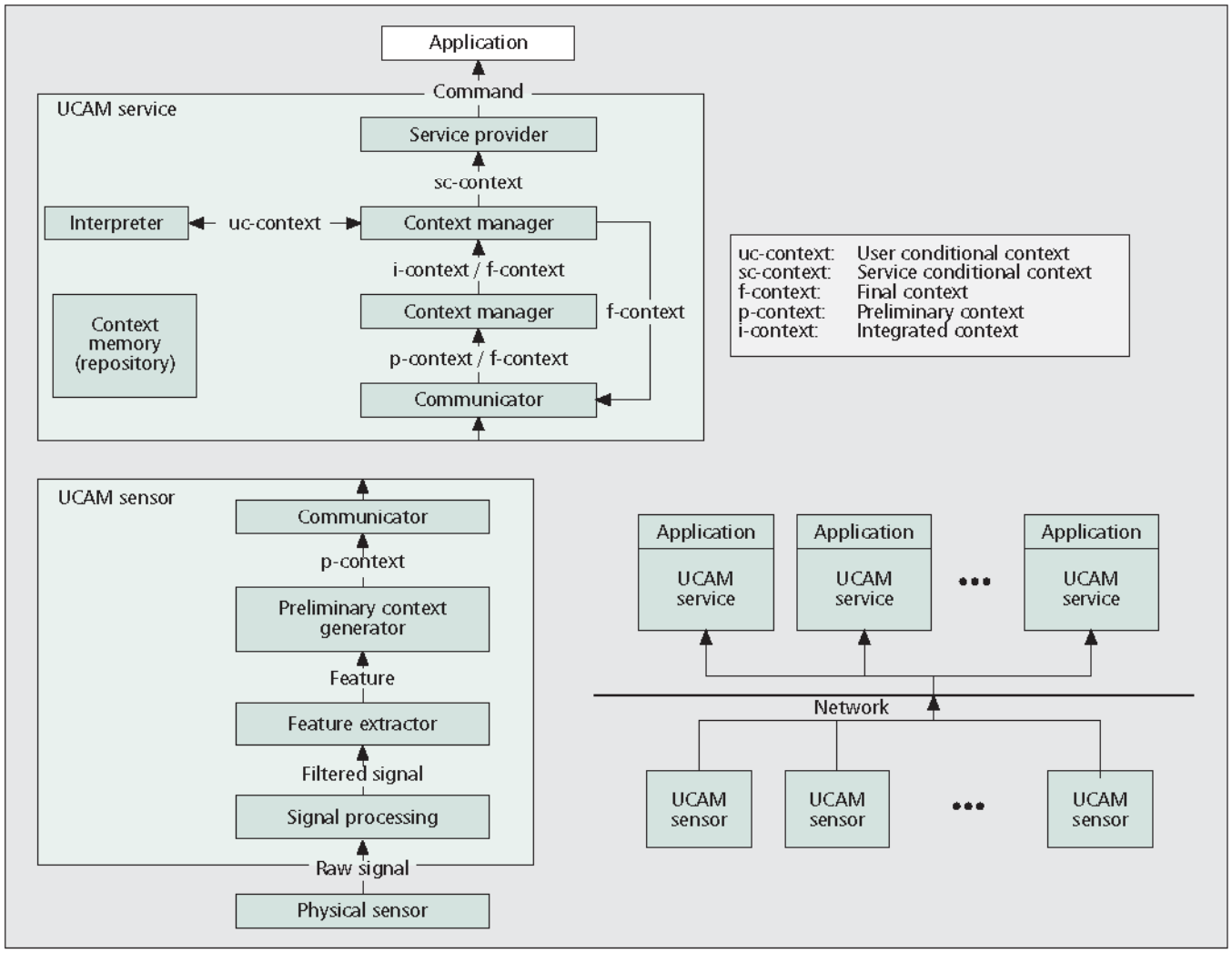


Figure 1. *UCAM architecture.*

because it can interpret and reason about them. The architecture can connect any tool used for the awareness ability as an add-on feature and manages communities among users, sensors, and services.

A context-aware architecture should have a context representation method that fully supports contextual information from sensors and services. In other words, a unified context representation method that adopts changes in an environment should be designed. Context fusion and reasoning mean the integration of contexts in various domains and the enhancement of intelligence ability by extracting high-level contexts. In order to manage complex or heterogeneous environments, methods of community composition and configuration should be considered at an architectural level. The service provisioning in the architecture enables easy and rapid development of context-aware services.

Figure 1 illustrates the UCAM architecture. It is a distributed architecture, which consists of UCAM sensors and UCAM services. An application connects to one or many UCAM services, and a UCAM service is connected to many UCAM sensors. The UCAM sensor is a logical sensor that interfaces with a physical sensor, while the UCAM service is a context processing unit that integrates and manages contexts. The goal of UCAM is to increase context richness, which it accomplishes by introducing new capabilities such as community management, and using richer semantic models in context representation and processing. This intelligence is improved by semantic representation, interpretation, integration, and reasoning of contexts.

The UCAM sensor has a hierarchy that features step-like signal processing, feature extraction, and preliminary context generation. The signal processing module, meanwhile, supports basic signal processing functions such as peak extraction and filtering functions (e.g., Hann, Hamming, rectangular, Bartlett, and Blackman windows). The filtered signal is sent to the feature extractor module, where computation of mathematical features such as mean, standard deviation, first and second order differences, and minimum and maximum intensity, as well as structural features such as time interval of peak to peak and frequency ratio, are provided as basic functions. Finally, the extracted features are converted to the preliminary context in the preliminary context generator module. To achieve this, the UCAM

sensor refers to sensory profile information described in XML. Each XML field has an ID, which represents an identical naming of each sensor device.

The UCAM service exploits users' intentions, which are extracted by the collected contexts. As shown in Fig. 1, the communicator manages and configures communities for efficient management of heterogeneous entities. Only necessary contexts are collected by the communicator. The context integrator, context manager, and interpreter perform the intelligent behavior of a context-aware service by using a unified context representation model. Moreover, those modules support semantic intelligence for community management. The service provider manages a service profile to support various applications. The proposed UCAM can easily utilize various artificial intelligence algorithms as add-on functions. The proposed architecture is thus an open source policy, and connects any context-aware service through a simple service profile in XML format.

Context Model and Processing

UCAM supports the full lifecycle (adding, deleting, replacing, modifying, and debugging entities) for context-aware applications in large-scale smart environments.

Context Representation — Context representation has the crucial role of communicating between the heterogeneous sensors and services distributed in a smart environment. The proposed architecture uses the 5W1H context model as its basis, and this model consists of context element (structured format), context (data), and context memory (repository) [2]. The 5W1H context is a structured format in which the context is expressed in terms of its components: WHO, WHAT, WHERE, WHEN, WHY, and HOW. The 5W1H context model is defined by a set of ontologies, which are an explicit specification of user context reasoning. The *WHO context* field consists of personal profile information such as social identity number, gender, age, social relationship, activity history, and so on. The *WHAT context* field represents profiles (device or property information) of sensor/service (identity, device information, etc.). The *WHERE context* field is filled with information such as the location and orientation of the user. In terms of the *WHEN context* field, it includes absolute time, as well as day information or relative time information, such as two hours after having a meal. The *HOW context* field represents the status of user — physiological condition, gestures, activity, and so on. Finally, the results of the high-level analysis after the processing of the reasoning module are stored in the *WHY context* field. Using the six questions and the structuring information gathered by the sensors allows us to accurately represent the user's situation as it can be observed by the sensors. We use this representation to provide a user-centric view of context that is appropriate for the anticipated systems we want to support with the architecture [2].

Our context representation has a variety of context usages:

- Preliminary context (*p-context*): Context that expresses the feature information and sensor description
- Integrated context (*i-context*): Context that forms the complete 5W1H
- User conditional context (*uc-context*): Context that expresses user-defined conditions for service execution as reconfigurable rules
- Service conditional context (*sc-context*): Context that expresses developer-defined conditions for service execution
- Final context (*f-context*): Context that contains control commands and the state of a service

Context Reasoning — Context fusion and reasoning in UCAM is a logical process that extracts a high-level (implicit) context from a low-level (explicit) context and maintains context consistency. Low-level and high-level contexts are classified by the level of intelligence (interpretation or reasoning). For example, a high-level context, such as when a user is watching a movie, can be extracted from a low-level context, such as identity, location, sound level, and device status information. Context fusion and reasoning are mainly performed in the context integrator. Context fusion contains information, such as a user's identity, location, activities, behavior, patterns, and intention. Originally, the context integrator was designed to provide information fusion on different levels and support the fusion of sensory data from various sources, which results in a system that accomplishes symbolic context fusion. Fusion on a raw data level is considered part of the sensor internals. By classifying and fusing according to user characteristics, user-centric context integration is supported. The reasoning is used to predict a user's behavior, attention, or intention by reusing context (context history) and adopting reconfigurable rules to the diverse environment. The rules are based on the Rete algorithm and are specified in the syntax of the LISP language. Context reasoning is implemented on top of the Java Expert System Shell (JESS) or the C Language Integrated Production System (CLIPS) according to the development environment. Our implemented reasoning simplifies reconfiguration in terms of rules structured in XML and executed by the user's explicit (i.e., direct control) or implicit (i.e., intention) input. For example, if the UCAM collects and fuses contexts such as identity, location, activity, and schedule data, it decides semantic information by executing the activity rule: "*Yoosoo is attending a meeting while sitting in room A.*" The generic behavior and functionality of a specific context-aware system are specified by the rules. Changes and reconfigurations of the system by a developer or maintainer are performed by adding, deleting, and modifying the rules in real time.

Community Management — In UCAM the concept of community computing is applied [8]. In this architecture we define community as a collection of UCAM entities sharing their context information to achieve common goals in dealing with services for users. Users, UCAM sensors, and UCAM services can all be members

The generic behavior and functionality of a specific context-aware system are specified by the rules. Changes and reconfigurations of the system by a developer or maintainer are performed by adding, deleting, and modifying the rules in real-time.

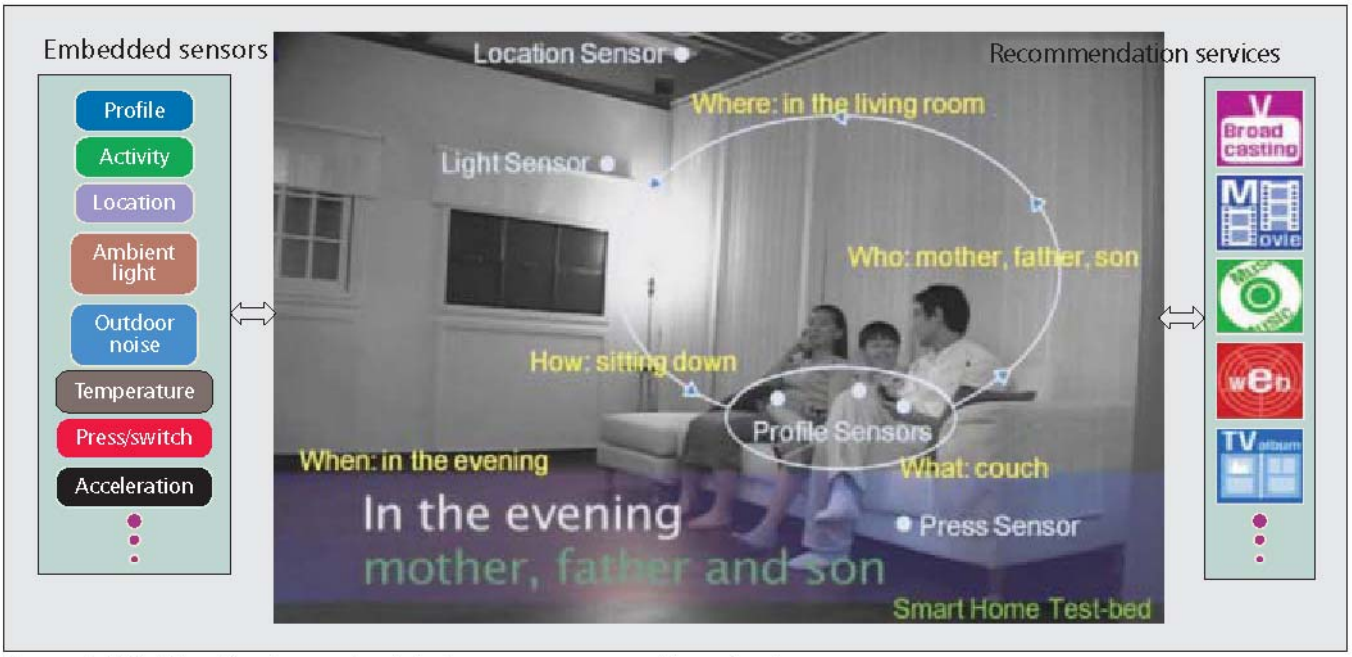


Figure 2. *UCAM application users' activity in a context-aware media application.*

of the UCAM community. For example, when a person uses a photo sharing system, a user, photo sharing service, and several sensors recognizing the user's activity and environmental changes are members of the photo sharing system. In addition to these members, other users who want to share photos and are connected with a main user can also be members of the community. Users are the subjects of the UCAM systems. UCAM sensors recognize a change in users and the environment, and UCAM services (actuators) affect them. When these entities have a common goal, they are members of the UCAM community. The communicator in UCAM manages communities, and it also takes charge of configuring the network and exchanging messages, including contexts among community members. The communicator finds community members located in the same environment and having a common goal, at which point the communicator senses various community members' changes in runtime by virtue of context exchange and transmits context information to community members. Through the relationship between community members, the level of contextual information delivered in heterogeneous environments is determined. The communicator selectively receives contexts from the UCAM entities related to the current service by using the determined context level.

Service Provisioning Management — Service provisioning management in UCAM describes a specific service and then executes the service as a response. In UCAM a profile described in XML is loaded from a file; thus, all information related to the service is determined in order to define the properties and behaviors available. The service profile includes the type of service, its location, its default conditional context, and a set of functions as actions. The default context is also delivered to the context manager to allow it to trigger the default response of the service. In runtime the profiled service is managed with the context delivered from the context manager. The context concerns an action to be executed and the user information. Therefore, the status of the service is also changed after executing the action, and the updated status is delivered to the context management area to notify it of the execution result. Through the service profile, any service developer can extract the common functions of UCAM, and then (s)he can inherit these basic functions and easily set the parameters of each. In short, any developer can utilize the context information in the network without having real hardware types, features, and interfaces.

UBIHOME DEMONSTRATION OF UCAM

To evaluate a realistic context-aware application, it is a good idea to consider sensing technology and techniques to monitor a user's activity from various sensors. Such technologies can serve as an example of the UCAM architecture. For a smart environment, we utilized our smart home testbed, ubiHome [9]. As shown in Fig. 2, various kinds of real UCAM sensors and UCAM services have been embedded in ubiHome. Thus, we implemented a context-aware media application for multiple users that is based on the UCAM architecture for the purpose of interacting with various UCAM sensors and UCAM services in ubiHome. This media application provides services that recommend media contents to a user according to the user's profile; this includes recommending content preferences by using real sensors embedded in a smart home. Figure 2 shows examples of the UCAM application, and presents implemented sensors and services with context representation in ubiHome.

This media application recognizes each user's location using the location tracking sensor and grasps multiple users' activities simultaneously.

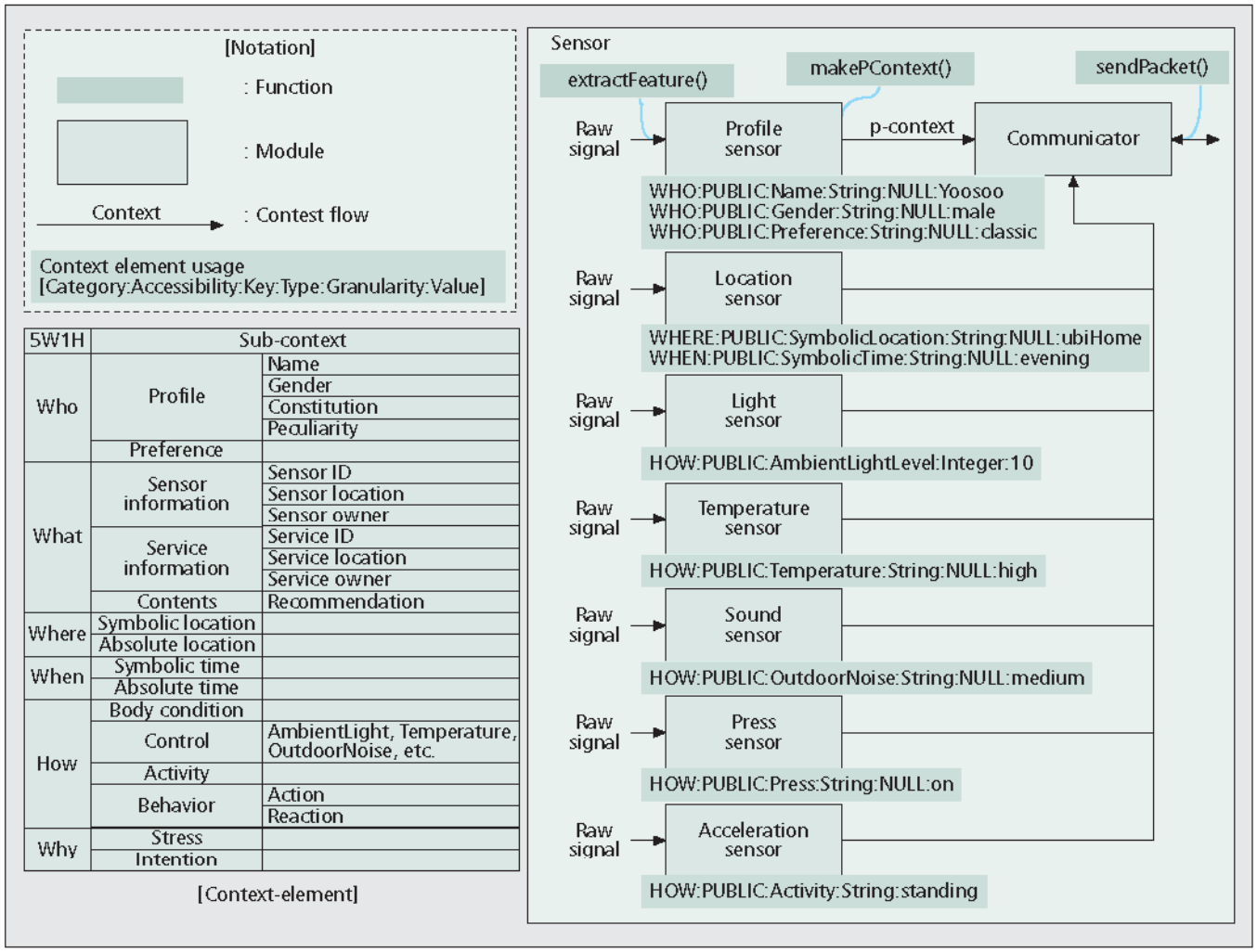


Figure 3. *UCAM application: implemented context and its flow in implemented sensors.*

As shown in Fig. 2, the activity of users is tracked by the UCAM. In this example the 5W1H context contains the activity information: who (mother, father, son), what (couch), where (in the living room/testbed), when (in the evening), why (not decided), and how (sitting down). The location sensor is based on the UCAM sensor, as shown in Fig. 1, and generates a *p-context*, including a user's location. The generated *p-context* is delivered to the UCAM service through communicators. The context integrator in the media application infers users' explicit intentions based on their activities by integrating multiple context inputs from a set of sensors, and creates an *i-context*. To reason out the intention, the *uc-context* from the interpreter is referred to as the rules. The media application then recommends appropriate services (TV, movie, music, web, album, etc.) to users by managing contexts (*f-context*) in the context manager. An application developer simply interfaces with the service provider to execute this application.

Figures 3 and 4 represent how the media application works in practice; the implemented functions, modules, contexts, and context elements are described. The 5W1H context can be categorized by multiple subcontexts according to each characteristic, as shown in Fig. 3. Figure 4 also shows the context reasoning process, which obtains everything from raw data (in heterogeneous sensors) to high-level context (in service). Thus, the user can evaluate the usefulness of the context reasoning. Through the relationship among family members, a developer can analyze in advance the users' experiences in a smart home.

Evaluation

We evaluated the proposed architecture with several UCAM sensors and UCAM services in ubiHome [9]. The evaluation was conducted as follows. For the experimental setup, we used a desktop computer (Intel Xeon 2.4 GHz, 2 Gbytes RAM), a real lamp (UCAM service), and several real/virtual sensors (UCAM sensor).

We first investigated the context collection to collect the *p-context* and decide the *f-context*. We used a real lamp service, four virtual UCAM sensors, and several real UCAM sensors (four sets of wearable sensors [activity, acceleration, and profile] and environmental sensors [ambient light, outdoor noise, temperature, press, and location]: a total of 32 real sensors). Every sensor generated different contexts every 500 ms. Actually, every sensor generated 1900 contexts:

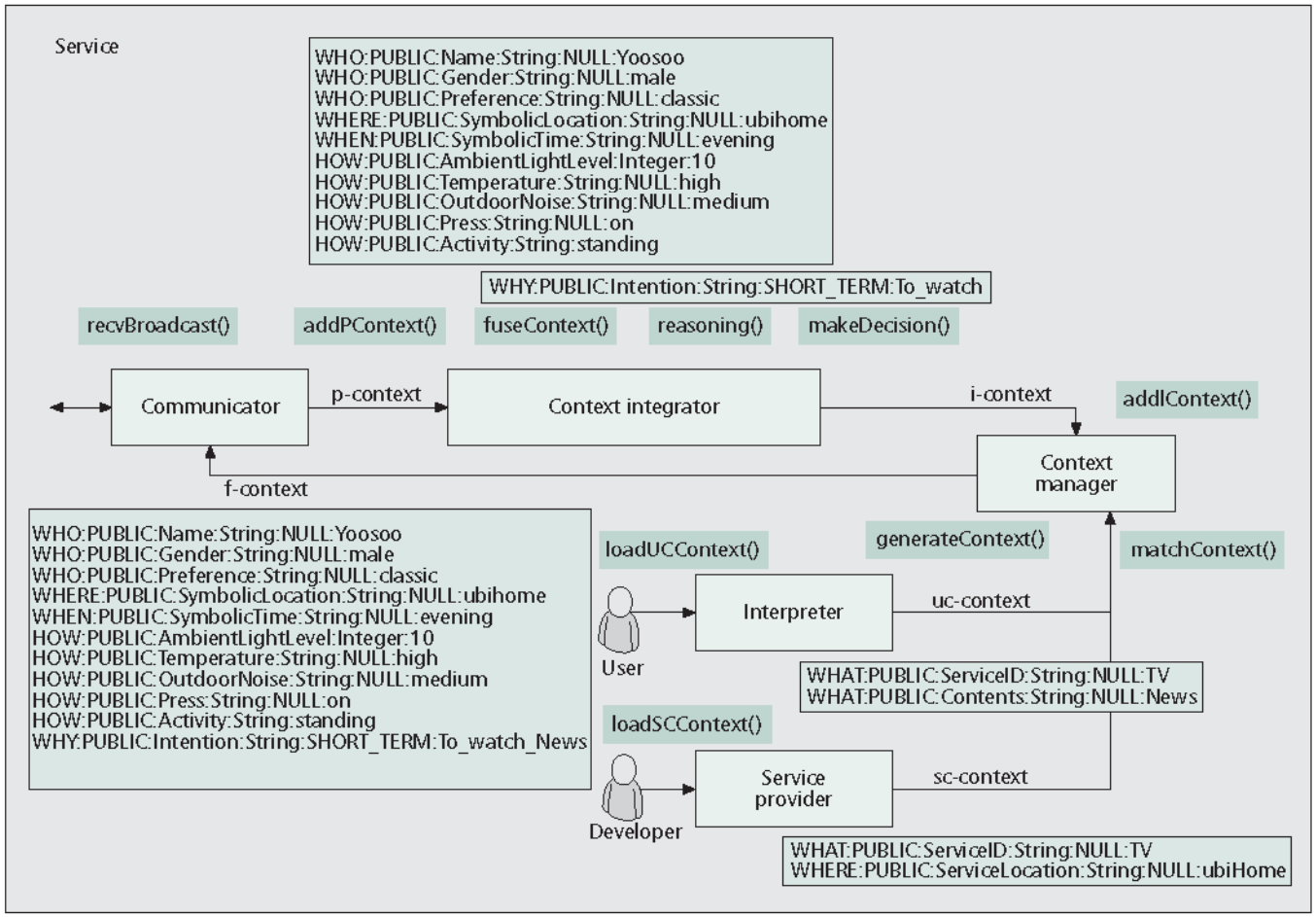


Figure 4. *UCAM application: implemented context and its flow in the implemented service.*

200 virtual sensor inputs, 200 activities, 300 accelerations, 100 ambient lights, 100 outdoor noises, 100 temperatures, 300 presses, 200 locations, and 400 profiles. We assumed that contexts from sensors had no errors. The implemented UCAM service collected contexts according to each individual user in real time, and it made a decision every 1000 ms. Moreover, the UCAM service made a decision by using the current collected contexts and recorded contexts (context history) at each decision interval. The experiment measured four cases: time to collect contexts [Time(collection)], time to decide [Time(decision)], number of contexts in collecting contexts [no.(collection)], and number of contexts in decision [no.(decision)]. In Fig. 5 the x axis represents the sequence of events in processing a context in UCAM, which includes input (*p-context*), processing, and output (*f-context*). The UCAM operated 500 cycles for about 1900 context inputs. We measured the changes of the collection and decision at each processing cycle.

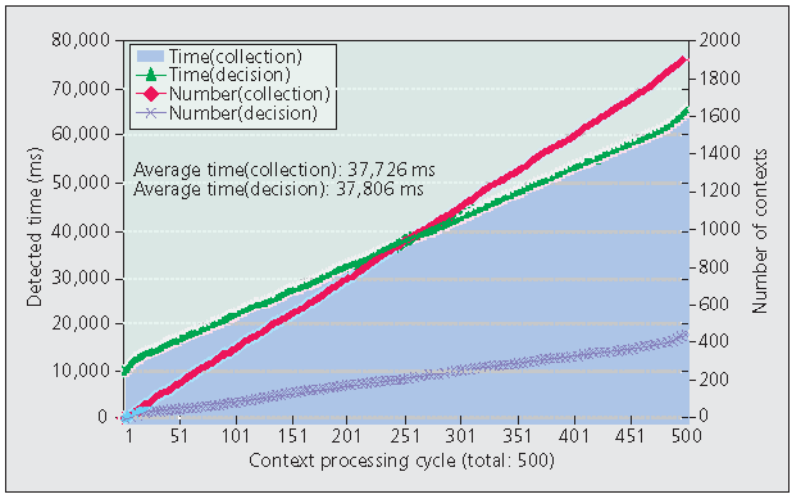


Figure 5. *The measurement results of collection and decision.*

In Fig. 5 we can see that Time(decision) is slightly higher than Time(collection), and Number(decision) is smaller than Number(collection). This is due to the fact that making a decision needed the context processing time for context management. However, the average difference (80 ms) between collection and decision was not much larger. As Number(collection) increased, Number(decision) also increased. The number of contexts was due to the fact that the decision generated a few contexts (436) by fusing and reasoning about the large number of collected contexts (1900). The UCAM reduced unnecessary contexts and generated high-level contexts, because the decision-to-collection ratio (Number(decision)/Number(collection)) decreased so that it steadily converged as the processing cycle

increased. Thus, we know that the proposed architecture collected large amounts of sensory data (1900) from heterogeneous UCAM sensors and made a decision by integrating those contexts without requiring much processing time. Accordingly, we could verify that the proposed architecture dynamically managed the contexts under changing large-scale sensory data in real time.

Next, we investigated the efficiency of the context delivery to evaluate the community management in the proposed architecture. We assumed that there were transmission errors (3 percent of transmissions). The relation between UCAM sensors and a UCAM service was randomly determined with uniform distribution, and the relation ranged from 0 to 1. For this experiment, we used 100 virtual UCAM sensors that had relations with a service. Each UCAM sensor generated 1000 contexts (100,000 contexts in total) and then sent them to the network. For the experiment, we measured the number of received contexts and the number of appropriate contexts. The accuracy of received contexts was defined as the probability that calculated the ratio of the number of appropriate contexts to the number of received contexts. We also counted the number of inappropriate context transmissions. Thus, we compared the number of inappropriate context deliveries with community management to the number of inappropriate context deliveries without community management.

In Fig. 6 we can see that the levels of accuracy of both received contexts were very similar and that the proposed case decreased the number of inappropriate transmissions. This means that the proposed architecture maintained the accuracy of received contexts, although many contexts from multiple UCAM sensors were sent to a UCAM service. When all of the UCAM sensors and a UCAM service are strongly connected (i.e., it is likely that they have a common goal), the number of inappropriate context transmissions approaches zero, whether or not the communicator works. However, if they are not connected with each other because they do not have a common goal, the result is improved by the communicator. When they are loosely connected, the number of unnecessary context deliveries decreases because the communicator sends contexts to community members only. Thus, we can conclude that the proposed architecture reduces unnecessary traffic by recognizing communities of multiple information sources in large-scale heterogeneous environments.

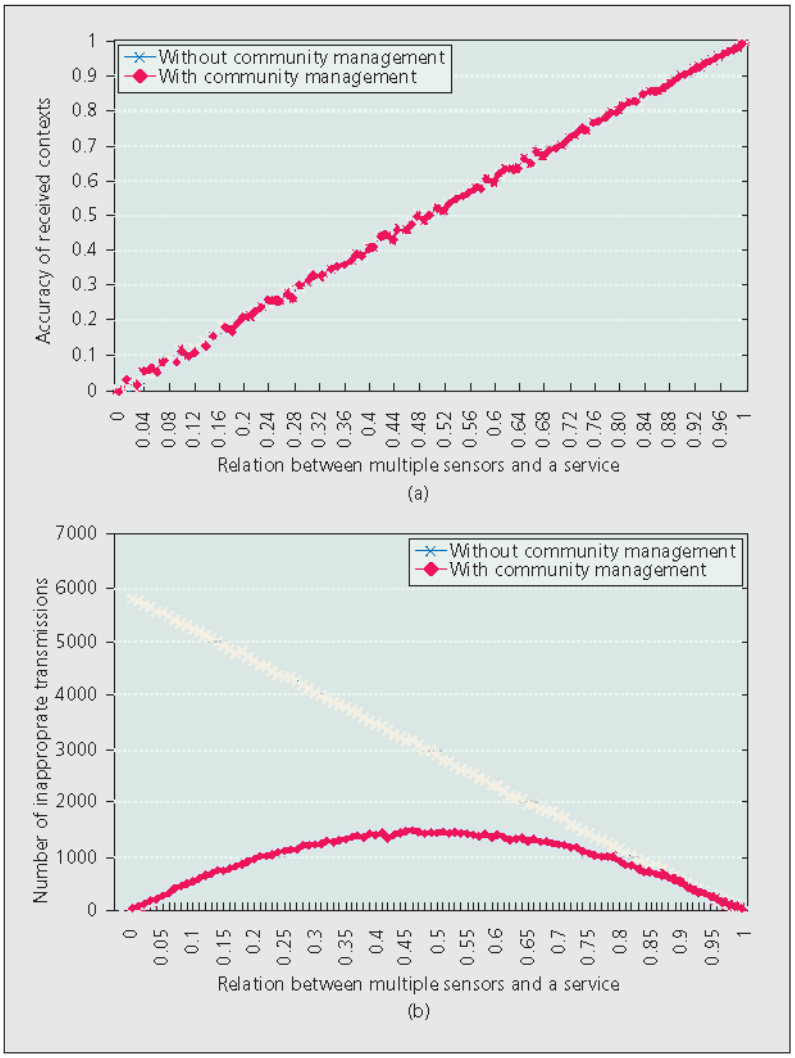


Figure 6. *The measurement results of community managements: a) the measurement of the accuracy of received contexts; b) the measurement of inappropriate transmissions..*

Conclusion

In this article we explain a context management architecture, which is a generalized context-aware architecture for heterogeneous smart environments. We designed and developed a hierarchical architecture that collected large-scale contexts from multiple heterogeneous sensors, added enhanced intelligence abilities to the proposed architecture, and developed the community management method for social awareness. Moreover, we suggested a service provisioning method that enables easy and rapid development of context-aware services. The proposed architecture can be extended to diverse application domains for large-scale smart environments. Also, the architecture is generalized as a library in order to easily be used by application developers.

In future work we will improve the prediction of users' behavior, even though the proposed architecture supports the reasoning aspect. Furthermore, we will evaluate the proposed architecture with evaluation methods used in the HCI field. Some evaluation factors such as architecture structure, scalability, accountability, reconfiguration, reusability, debugging, openness, and flexibility can be useful for the general approach and intelligence ability of the proposed architecture.

Acknowledgment

This research is supported by the Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA), under the Culture Technology (CT) Research & Development Program 2009.

REFERENCES

[1] D. Salber, A. Dey, and G. Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications," *Proc. ACM Conf. Human Factors Comp. Sys.*, 1999, pp. 434–41.

[2] D. Hong, H. Schmidtke, and W. Woo, "Linking Context Modelling and Contextual Reasoning," *Proc. 4th Int'l. Wksp. Modeling Reasoning in Context*, 2007, pp. 37–48.

[3] T. Gu *et al.*, "Toward an OSGi-Based Infrastructure for Context-Aware Applications," *IEEE Pervasive Comp.*, vol. 3, no. 4, 2004, pp. 66–74.

[4] J. Bardram, "The Java Context Awareness Framework (JCAF) — A Service Infrastructure and Programming Framework for Context-Aware Applications," *Proc. 3rd Int'l. Conf. Pervasive Comp.*, 2005, pp. 98–115.

[5] N. Behlouli, C. Taconet, and G. Bernard, "An Architecture for Supporting Development and Execution of Context-Aware Component Applications," *Proc. IEEE ICPS '06*, 2006, pp. 26–29.

[6] A. Schmidt *et al.*, "Advanced Interaction in Context," *Proc. 1st Int'l.Symp. Handheld Ubiquitous Comp.*, 1999, pp. 89–101.

[7] S. Jang and W. Woo, "ubi-UCAM: A Unified Context-Aware Application Model," *Proc. CONTEXT '03, LNAI*, 2003, pp. 178–89.

[8] M. Kumar *et al.*, "PICO: A Middleware Framework for Pervasive Computing," *IEEE Pervasive Comp.*, vol. 2, no. 3, 2003, pp. 72–79.

[9] Y. Oh *et al.*, "The ubiTV Application for a Family in ubiHome," *Proc. 2nd Ubiquitous Home Wksp.*, 2005, pp. 23–32.

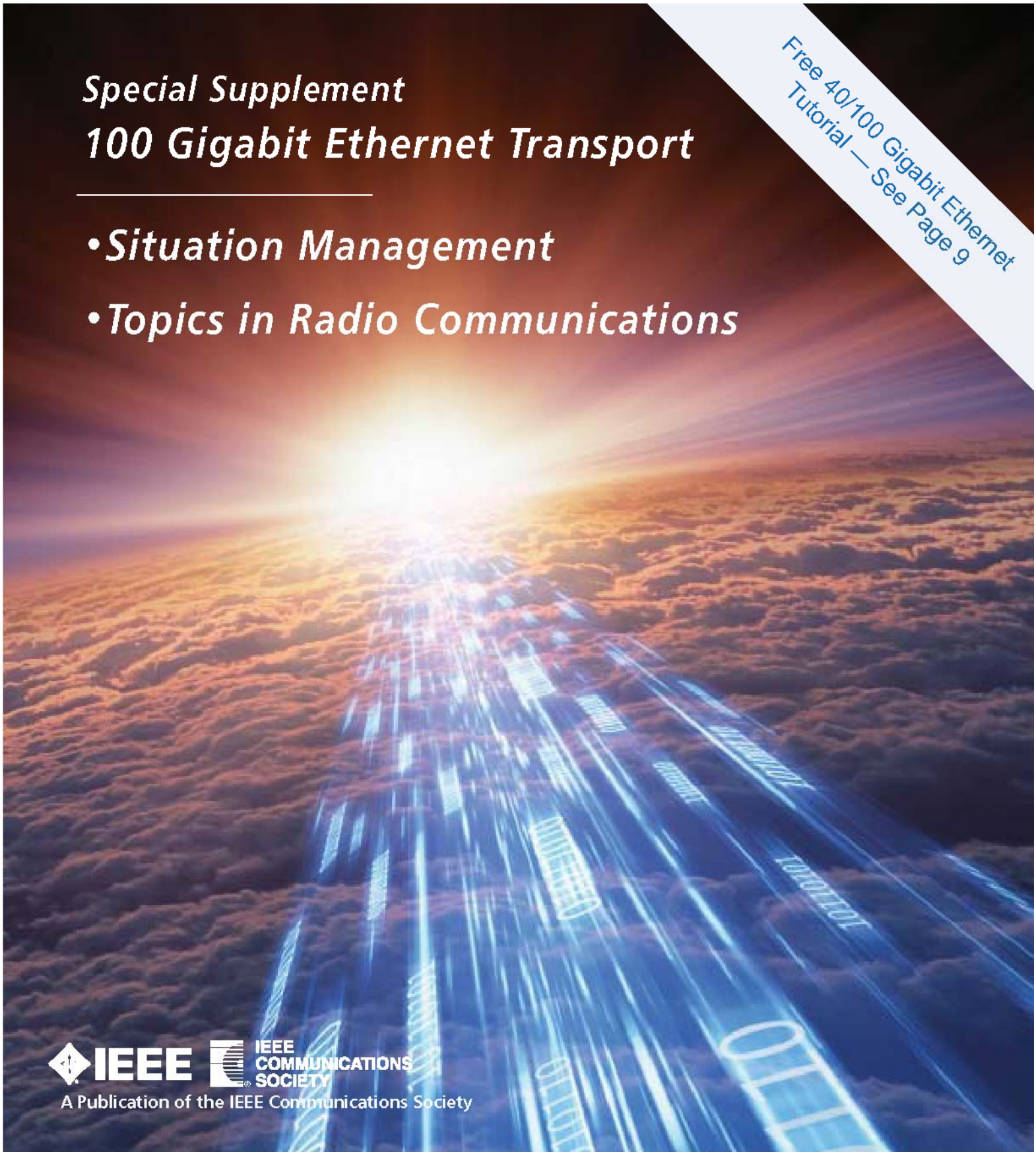
BIOGRAPHIES

YOOSOO OH (yoh@gist.ac.kr) received his B.S. degree from Department of Electronics and Engineering, Kyungpook National University in 2002, and his M.S. degree from the Department of Information and Communications (DIC), Gwangju Institute of Science and Technology (GIST), Korea, in 2003. He is currently working toward his Ph.D. degree in DIC, GIST. His research interests include context-aware computing technology, human–computer interaction, ubiquitous computing, wearable computing, and smart home.

JONGHYUN HAN (jhan@gist.ac.kr) received his B.S. from the Department of Computer Science, Ajou University, Suwon, Korea, in 2005 and his M.S. from DIC, GIST in 2007. He has been a Ph.D. student in DIC, GIST since 2007. His research interests include social network, web search and & mining ,and context awareness for ubiquitous computing.

WOONTACK WOO (wwoo@gist.ac.kr) received his B.S. in electronics engineering from Kyungpook National University in 1989 and his M.S. in electronics and electrical engineering from POSTECH in 1991. In 1998 he received his Ph.D. in eectrical engineering systems from the University of Southern California. In 1999, as an invited researcher, he joined Advanced Telecommunications Research (ATR), Kyoto, Japan. Since February 2001 he has been with GIST, where he is an associate professor in DIC and director of the Culture Technology Institute (CTI). His research interests include 3D computer vision and its applications, including attentive AR and mediated reality, HCI, affective sensing, and context awareness for ubiquitous computing.

IEEE Communications MAGAZINE
www.comsoc.org
March 2010, Vol. 48, No. 3



Special Supplement
100 Gigabit Ethernet Transport

- Situation Management
- Topics in Radio Communications

Free 40/100 Gigabit Ethernet Tutorial — See Page 9

IEEE COMMUNICATIONS SOCIETY
A Publication of the IEEE Communications Society