A Dynamic Control Scheme of Context Information based on Multi-agent

Hideyuki Takahashi[†], Yoshihisa Sato[†,‡], Takuo Suganuma[†,‡], and Norio Shiratori[†,‡]

[†]Research Institute of Electrical Communication, Tohoku University, Japan
[‡]Graduate School of Information Sciences, Tohoku University, Japan
{hideyuki, yosi-q, suganuma, norio}@shiratori.riec.tohoku.ac.jp

Abstract - In ubiquitous computing environments, an effective handling of Quality of Service (QoS) is needed to provide context-aware service in tune with the variation of context information. But QoS for overall system may fall if either the quantity or quality of context information exchanged among entities is insufficient or excess. Based on user context, resource context and QoS, we propose a dynamic control scheme for context information delivery. Based on highly autonomous and cooperative multi-agent system, we compose the entities to configure context-aware services. Our proposed scheme can efficiently adapt to the various changes in the real-world environment and maintain the QoS to the user satisfaction. The effectiveness of our proposed scheme is evaluated from the perspective of adaptability depending on the varying relationship between real-world and ubiquitous computing environment.

Keywords: Ubiquitous computing, Context information management, Quality of Service, Quality of Context, Multi-agent system

1 INTRODUCTION

In recent years, ubiquitous computing (ubicomp) [1] environment is emerging, where various kinds of sensors, hand-held terminals, and wireless networks cooperatively work to support daily lives of the people. As a distinctive service (e.g., wellness management service, telemedicine support service, and mobile information service) provided by this environment, context-aware services come to the front [2]–[4]. The context-aware services are the services in ubicomp environment based on the "context" which is the situation of each entity. Here, the entity is the element in ubicomp environment such as users, hardware devices, software, and networks.

The context of an entity is delivered in the form of information in the network, and used by other entities. We call this information as "Context Information" (CI). The CI is exchanged at very frequent intervals, then the entities should transfer and handle massive amounts of CI, as well as handling data for main services, using the shared resources. Especially in case of application that consumes a large amount of computational and network resources, such as multimedia communication service, the resources for service provisioning itself decrease due to excessive circulation and processing of CI. Therefore a critical degradation in the QoS in overall system may result.

To solve this problem, Tokairin et al. [10] proposed a Context Management Scheme to keep the QoS. Their scheme is based on the concept of quality of context (QoC), and increases QoS in ubicomp environment by managing CI delivery effectively. However, even if the types of entities are same, the behavior of each entity is different depending on the physical environments where they exist in the real world situation. Hence, the autonomous and adaptive control scheme of CI delivery is needed, according to the varied physical environment in the real world.

Our goal is to develop an effective CI managing scheme for context-aware services to provide appropriate QoS according to varied physical environment and user's requirements. This paper presents a dynamic control scheme of CI delivery based on multi-agent. To realize this we compose entities which configure context-aware services as highly autonomous and cooperative agents in this scheme. By employing the proposed scheme, the system can provide the context-aware services in varied physical environment flexibly by using the situational adaptability of each agent and their organizational behaviors.

Especially, we performed an initial experiment with the prototype system of a ubiquitous live streaming video service with our proposed scheme. From the results, we confirmed that the system can control the CI delivery according to provided QoS and the change of location and speed of a user entity in the physical environment. We also confirmed the effectiveness of our scheme in perspective of user-level QoS of the video streaming such as frame-rate and timeliness of service provisioning, according to the changes of entities in the real-world environment.

2 RELATED WORK AND PROBLEMS

2.1 Related Work

In this section, we present related works and summarize their problems. Some studies have explored concentrating on the acquisition and selection of context information (CI) for context-aware service provision [5], [6]. CHANSE [5] aims for easy configuration of context-aware service by using centralized management server of CI. In case of failure to get requested CI for configuration of service due to break down of a sensor device, the system can keep CI provisioning by selecting alternative CI from other available sensors. This mechanism gives a good solution to increase availability of context-aware service; however, it is required to describe static process of electing CI when a device is newly introduced. The dynamic reconfiguration of sensor devices is also expected when the device providing CI breaks down. ContextDistillery [6] proposes a framework that aims for abstraction of the up-to-dateness's variety of each CI. However, it is required to describe the process of selecting CI and up-to-dateness of CI statically in its design phase. The dynamic provision of CI to adapt to the operational situation of the system is expected in the real world.

Additionally, some studies have investigated focusing on Quality of Context (QoC) [7]–[9]. Buchholz et al. advocated the notion of QoC [7]. They introduced and defined from "precision", "probability of correctness", "trust-worthiness", "resolution", and "up-to-dateness" for QoC parameters. They also refer to relation of QoC, QoS, and Quality of Device (QoD). Sheikh et al. [8] define QoC aiming to deal with complicated specification of CI effectively in the middleware for context-aware service. They defined QoC parameters for more practical use than [7], they are, "Temporal Resolution", "Spatial Resolution", and "Probability of Correctness". They mainly discuss the signification of usage of QoC parameter, however, it is not clear how the QoC may be used in system operation for real world applications. An adaptive middleware framework [9] is proposed to provide and select CI depending on the application. They define QoC by "Precision" and "Refresh Rate". This framework computes Utility Function based on the QoC; CI are selected according to the computed amount. This is the pioneering work to use QoC in the real system. However it is difficult to control "Precision" and "Refresh Rate" dynamically because these QoC parameters are assumed to be preliminarily defined and advised.

On the other hand, a flexible QoC control scheme during the system operation is proposed [10]; this scheme aims to provide the advanced QoS control ability. They propose context information management based on multi-agent system; while they focus on "up-to-dateness" as a QoC parameter. This scheme enables the control of CI delivery based on resource status and user's physical location. It is possible to manage QoS of overall system. However, because applicable real environment and services are restricted by the static value of the threshold of resource status and the fixed area of the user's location, which are specified in the design phase of the system, thier scheme has some limitations in scalability and flexibility of QoS control ability.

2.2 Problems

We tackle the control of the context information (CI) delivery. Here, we point out two current technical problems. We assume that ubicomp environment comprises physical environment (real space) and ubiquitous computing space (ubicomp space). We describe the problems of adaptability from viewpoints of relationship between real space and ubicomp space.

- **Effective CI delivery based on the run-time behavior of entities (P1)**: Suppose the cases where context-aware services are provided by using computational and network resources shared by main services and CI delivery in ubicomp space. The resources required to provide its main service is degraded due to the resource limitations of ubicomp space when the amount of CI exchanged among entities becomes huge. Consequently, the main service doesn't work properly or the QoS may be greatly decreased. The effective CI delivery in ubicomp space, deeply considering the run-time behavior of entities in real space, is essential.

- **Flexible CI delivery adapted to change of real space in long-term range (P2)**: We described the problem about reactive adaptability of CI delivery control in run-time in (P1); however, we must deal with the changes of real space in more long-term range. These changes may occur with reorganization of furniture in a room, replacement of the role of a room, modification of work flow, etc. We also need to take the deployment problem of a ubicomp space to a real space into account. It is necessary to customize and adjust the ubicomp space delicately in order to make it workable on the target real space because of diversity of real space. Therefore, it is required to adapt the QoC control scheme autonomously during the service provisioning when a ubicomp space is deployed on an arbitrary real space, and even when the real space changes in long-term range. Then, it is possible for the system to improve its QoC control ability by itself gradually over time.

3 DYNAMIC CONTROL SCHEME OF CONTEXT INFORMATION BASED ON MULTI-AGENT

3.1 Relation of QoS and QoC

A general model of context-aware service provision system is shown in Figure 1. The real space comprises four kinds of entities: hardware entities such as PC, PDA, and RFID Tag, software entities such as video transmission/receiving system working on PC, network entities such as wired/wireless networks that connect the entities, and user entities as human user of the system. Here "Context Information" (CI) is defined as the information which carries the situations of these entities.

The entities provide a service for user entities in cooperation with other entities. If the system controls a service based on a specific CI, we call this service "context-aware service." First, the system collects CI from each entity in real space, as shown in Figure 1. "CI Delivery and Handling System" processes the CI and passes only the necessary CI to "Service Provision System" that provides the main service. Service Provision System organizes the entity group to provide the service based on the CI, and the system starts the service for the user entity.

Both the systems in ubicomp space sometimes use the same resources; where resources mean computational resources and network resources. In this situation, if enough amounts of resources are not available, the conflict on the resources occurs

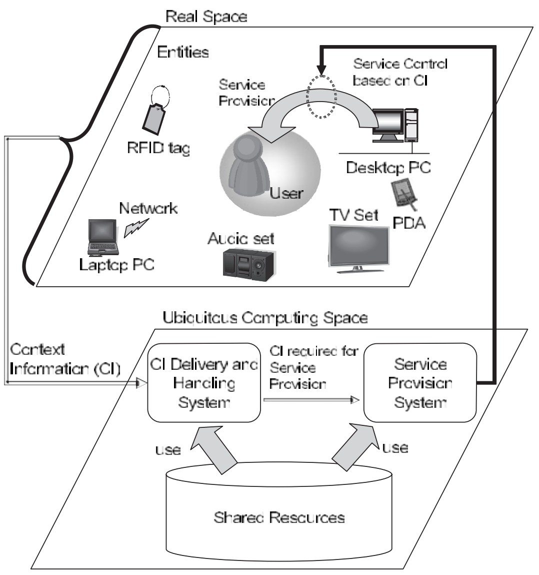

Figure 1: A model of context-aware service provision system in ubiquitous computing environment

and then we cannot get enough performance as described in Section 2.2(P1). Therefore, we need to tune the system to improve the provided QoS as much as possible.

We define QoS and QoC in ubicomp environment as measurement metrics to tune this system effectively. QoS represents the quality of service that ubicomp space provides to the users. We define QoS as following three parameters;

$$QoS =< Service{-}Quality, Timeliness, In{-}Placeness >$$

$Service - Quality$ means a quality of providing service itself. $Timeliness$ is an indicator that shows how the service starts in right timing from temporal viewpoint. $In - Placeness$ is a measure that shows how the service is provided in appropriate place from special viewpoint. Second and third items are essential in ubicomp environment, and it is important to provide service that satisfies user requirement to these things.

Additionally, QoC is defined based on [7];

$$QoC =< precision, correctness, trust{-}worthiness,\\ resolution, up{-}to{-}dateness >$$

Here, QoC means the quality of CI exchanged among entities and has tight relationship with QoS. A large amount of resources are consumed when CI is delivered and processed with high QoC. Then the resource that is used by the main service is reduced, and the value of each parameter of QoS decreases in the end; concretely speaking, $Service{-}Quality$ decreases. On the other hands, when QoC is decreased, the context awareness is also degraded. It causes troubles such as service delay in establishment (low $Timeliness$), unexpected starting of service in the place where the users do not exist (low $In{-}placeness$), and so on in these cases. These are fatal errors in context-aware services.

We focus on the relationship between QoS and QoC in this paper. We ensure QoS-aware context service provision by controlling QoC based on the condition of entities in real space and provided QoS, and realize better QoS as much as possible.

3.2 Overview of the Dynamic Control Scheme of Context Information Delivery

This paper describes a "Dynamic Control Scheme of Context Information Delivery". This scheme needs to realizes ubicomp space that can provide effective context-aware services by adapting to various changes in real space. By employing the proposed scheme, the system can improve QoS by managing QoC based on the relationship between QoC and QoS described in Section 3.1. This scheme has the following two functions.

- **Dynamic QoC Control Function based on the behavior of entities in real space (F1)**: This function follows the short-term behavior of entities on the second time scale and automatically adjusts QoC to provide higher QoS and keep it as stable as possible. For example, in the case of ever-changing of the location CI of user entity and hardware entities caused by the user's movement in a room, this function avoids to decrease QoS by increasing/decreasing the QoC of the location CI according to the user's location and movement. This function solves (P1) described in Section 2.2.

- **Adaptation Function for long-term changes in real space (F2)**: This function tunes working parameters of (F1) according to the changing situation of entity for long-term (like in a matter of days) in real space. For example, this function can make the adjustment of algorithm of (F1) adaptively in case the trend of entity's working conditions are changed by the rearrangement of furniture and IT devices inside the room. This can solve (P2) of Section 2.2.

In addition, there needs to be another adjustment for further long-term changes in real space, like the change of number of entities or their quality, occurred by replacement of entity members. In this paper, we omit this type of adjustment.

Compared with existing studies, Tokairin et al. [10] have investigated the QoC adjustment based on user location considering trade-off between QoS and QoC. They suggest a method to vary CI along with the area where the user exists. The areas are determined in advance, and they assign value of QoC per area statically. This gives a partial function of (F1). However, this method lacks of flexibility and extensibility against the changes of real space. Our proposed scheme realizes (F1) fully to apply various real spaces easily. Moreover our scheme can overcome existing method in terms of the adaptability by introducing (F2) which is difficult to realize in existing works.

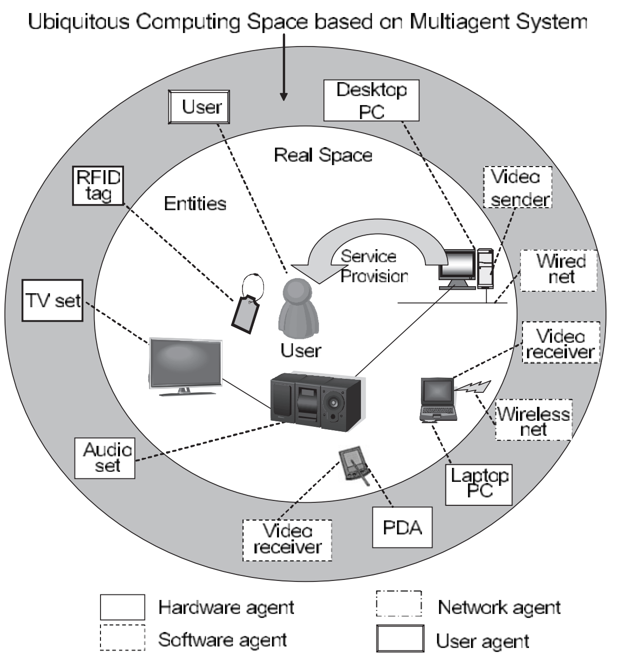

Figure 2: Ubiquitous computing space based on multi-agent

3.3 Architecture based on Multi-agent

The problems (P1) and (P2) in Section 2.2 originate from the limitations of autonomy and cooperativeness of each entity, and the lack of infrastructure for a flexible system construction to support context-aware services. We therefore compose these entities as highly autonomous and cooperative agents, and construct the ubicomp space as a multi-agent system. Figure 2 shows the Ubiquitous computing space based on multi-agent. Each entity is monitored and controlled by the individual agent. Each agent manages CI of each entity and exchanges the CI among the agents by using inter-agent communication protocols. The agent also has knowledge on the specification of the target entity. According to this specification knowledge and monitoring results, the agent can recognize the behavioral situation of the entity. The agents can produce an organization based on the contract among the agents, to configure service organization of entities dynamically [11], [12].

We expect the following advantages by using the concept of multi-agent.

- Effective service composition in ad hoc manner based on the dynamic selection and synthesis of CI by agent's ability of self-awareness and cooperativeness

- Autonomous acquisition of CI reflecting operational situation of entity by the reflection and autonomy ability of agent

- Advanced provision, delivery and distributed management of CI by the agent's cooperation ability

- Improvement of extensibility and flexibility in the system construction by the modularity and the organizational behavior of agent

3.4 Design of Dynamic Control Scheme of CI

(F1) and (F2) are functions to keep QoS high and stable by monitoring the observable CI that affects to QoS of context-aware service, and by adjusting QoC of the operable CI. In particular, we define (F1) as a mapping function that calculates the value of QoC by using appropriate observable parameter of CI as "variable" and pre-defined "coefficients" of each application. We also define (F2) as a function that updates the coefficients based on evaluation after a single service provisioning is finished. This mapping function is defined for every application and is maintained in the agent that has the decision making role.

In this paper, we consider the design of this scheme by using an application example in which the movement of a user is regarded as the change of situation of the real space. This service can be provided at the right place according to the user's location. Here, we regard the location information of the user entity as the observable CI. We also regard the up-to-dateness of location information of user entity as the QoC of operable CI.

In order to trace the movement of a user correctly, the faster the user moves, the higher up-to-dateness of the location information we need. This is because the system must keep the difference small between the user's actual position and the location information. Furthermore, we need to change the up-to-dateness of (QoC) based on not only user's moving speed but also the distance between the user and a service terminal. When the service terminal is far away, we can set low QoC and reduce the consumption of the resource, because the possibility of service provisioning is rather low. We also need high QoC and check the user's location frequently when the user closes to the service terminal, because the possibility of service provisioning increases. By definition of the mapping function that uses the CI as variables, we do not have to set the fixed value manually for mapping between observed CI and QoC. Therefore, we can change QoC continuously based on the changes of real space.

3.5 An Example of Design of Proposed Scheme

We illustrate an example of design of the proposed scheme by using the ubiquitous video streaming service as shown in Figure 3. We consider the following scenario in this service. The user moves in a room while receiving movies with the handheld PC. When the user moves, the system switches the display device and migrates the video player service to a desktop PC that can display the video in higher quality. In this scenario, the desktop PC that can provide the service calculates distance between the user and desktop PC. Moreover it calculates the user's speed from movement distance of the user and an elapsed time from the previous point. The system sends

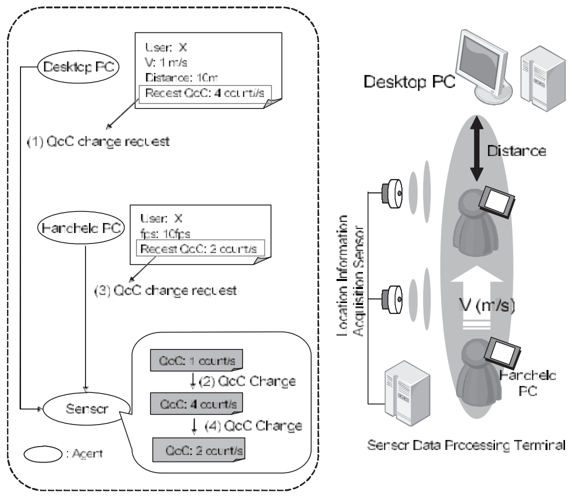

Figure 3: An example of dynamic QoC control function

a QoC change request according to the user's speed and the distance between the user and the desktop PC.

Furthermore, QoC should be controlled considering status of QoS to the user. Suppose that the desktop PC shown in Figure 3, for instance, sets the high QoC, whereas the handheld PC runs out of resources because of the excessive delivery of CI. Thereby it may degrade service quality of video. By monitoring QoS parameters such as frame rate of video, when the actual decrease of QoS occurred, the system requests to change up-to-dateness (QoC) lower. This QoS-based control of QoC can keep the behavior of the entire system stable.

Here is an example of QoC calculation function with a scenario shown in Figure 4. A user moves from point A to point B with handheld PC connected to the Wireless network. The distance between point A and point B are the cover area of the ultrasonic location sensor (ZPS). The service migration occurs when the user get close to the desktop PC at point B. Here, we use the frequency of location information update as the $up{-}to{-}dateness$ parameter of QoC, and frame rate (fps) as the $Service{-}Quality$ parameter of QoS as mentioned in Section 3.1. In this case, we calculate QoC as follow:

$$QoC = \frac{w}{(v+p)\{(X-x')+\alpha\}} \tag{1}$$

where X is the maximum distance between point A and point B as shown in Figure 4, x' is the distance between the user and the terminal that is the destination of the service migration, v is the speed of the user, p is a constant to consider when the user does not move, and w and α are constants to set initial value of QoC when the user stays at point A. The Adaptive Function (F2) is achieved by adjusting each coefficient value X, w, p, and α.

Moreover, the system changes QoC when the average of the frame rate decreases more than that requested by the user. We calculate and update the new QoC value by following simple expression:

$$QoC = \frac{QoC'}{2} \tag{2}$$

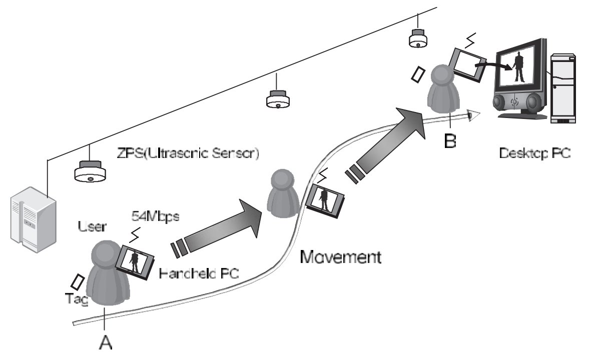

Figure 4: A scenario of QoC control experiment

where QoC' is the QoC before the request is issued.

4 IMPLEMENTATION

4.1 An Application to Live Streaming Service for Ubicomp Environments

We developed an experimental system of live streaming service for ubicomp environments. This is a system that transmits a live video captured by multiple cameras to users at remote locations over the network. This system can play the video on a display device such as TV monitor or display of PC which is located at the nearest position to the users. The system has functions to switch the displays and to migrate the video player service according to the user's position and request of QoS. For example, we expect various application areas like a supervising system for the appearance of the elderly person at home, user's baby in the day-care center, and the indoor pet.

4.2 Design of Experimental System

Figure 5 presents the agent's organization used in this experimental system. The role of each agent is shown as follows:

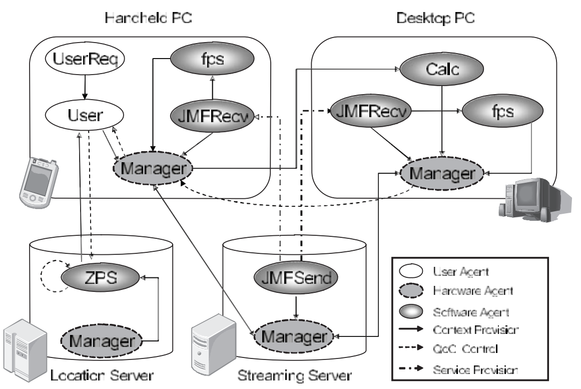

Figure 5: Agent organizations in experimental system

- *User*: This agent manages CI about a user entity. It manages the user's profile, request of QoS, location information, etc.; it sends the user-related information to other agents.
- *UserReq*: This agent controls U/I software entities that have functions to acquire user request. It informs the user request obtained through this U/I to the *User* agent.
- *ZPS*: This agent has function of acquiring and providing the location information of a specific entity with its tag using an ultrasonic positioning sensor system [14]. This agent can control the up-to-dateness of location information dynamically based on requests (QoC change request) from other agents; this agent informs the CI to other agents.
- *Manager*: This agent manages all the agents on each terminal PC; *Manager* can communicate with the other *Manager*. It judges timing and devices for transmitting and receiving the video service to be switched.
- *JMFSend, JMFRecv*: These agents control Java Media Framework (JMF) [15] which is an entity of multimedia communication software component. *JMFSend* agent transmits streaming video and *JMFRecv* agent receives it based on the request of *Manager*. These agents can adjust the quality of video such as encoding quality, data rate, and frame rate.
- *Fps*: This agent monitors the frame rate of streaming video which *JMFRecv* receives. It passes on a warning to *Manager* when it detects requirement violation about the frame rate.
- *Calc*: This agent calculates the distance between the user and the target PC, and the speed of the moving user using the location information; this agent derives value of QoC by using the mapping function described in Section 3.5.

Figure 6 shows the experimental environment. We employed DASH [11] and IDEA [13] for the software infrastructure. DASH is a rule-based multi-agent framework, while IDEA is a development-support environment of the DASH agents. Figure 7 shows a snapshot of a handheld PC and an ultrasonic positioning sensor system (ZPS). We used ZPS as a sensing device for the location information. In this system, a ZPS tag is carried by a user. Thus the location information of the tag is mapped onto the location information of the user entity.

We used C++ and java as program languages of the system build on Windows XP operating system. As for the transport layer, we used TCP in the control part and UDP for the video transmission. We constructed the environment with two kinds of networks: wired (Ethernet 100 Mbps) and wireless (IEEE802.11g, 54 Mbps) networks. In addition, we used a Web camera to capture the video and PC display to receive and play the video delivered from the Web camera.

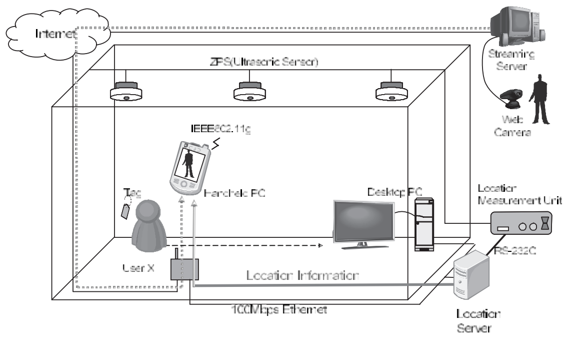

Figure 6: Experimental environment

As for the default setting, a handheld PC which is always carried by a user displays the video form a Web Camera in Figure 6. Moreover, the destination of output of the video migrates from the handheld PC to the display of the desktop PC when the user approaches a desktop PC. *User* and *UserReq* reside in the handheld PC. *User* manages the user's location information receiving from *ZPS*.

5 EXPERIMENTS AND EVALUATION

5.1 Experimental Method

We performed experiments to confirm effectiveness of the proposed scheme. In these experiments, we evaluated that the system can satisfy QoS requirement of user by QoC control which is adapted to the user's location and moving speed. We use a prototype system described in Section 4.2.

We measured frame rate in receiving the streaming video. This corresponds to the $Service{-}Quality$ parameter of QoS described in Section 3.1. We also measured the time that has been spent on the migration of the video streaming from the handheld PC to the desktop PC. This is regarded as the $Timeliness$ parameter of QoS. Moreover, we used frequency of location information update as the $up{-}to{-}dateness$ parameter of QoC.

This experiment uses the scenario of Figure 4. The user moves from point A to point B at a constant speed with the

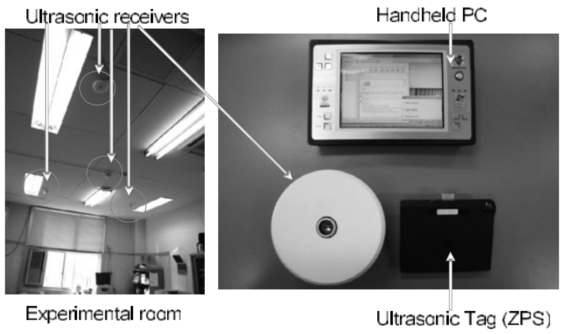

Figure 7: A snapshot of hardware configuration for experimental system

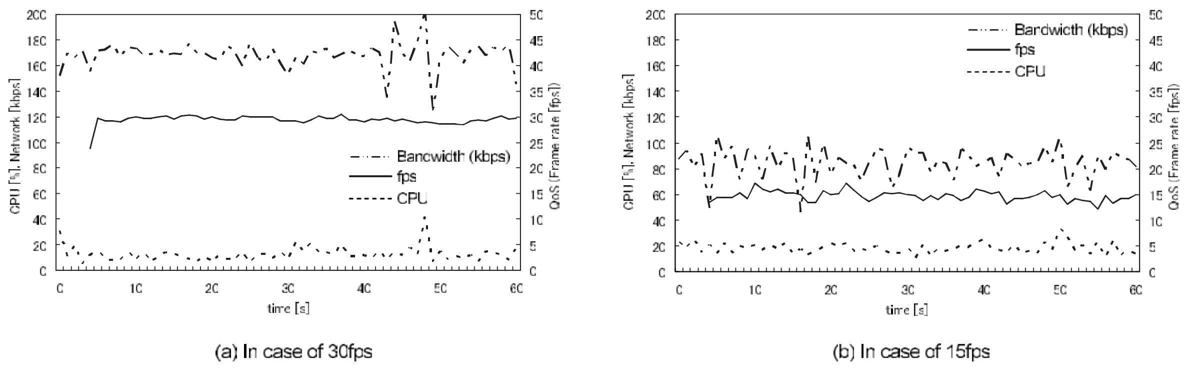

Figure 8: Experimental results of the handheld PC's resources consumption without receiving the location information

handheld PC. The service is migrated when the user approaches to a desktop PC at point B.

We used Expr. (1) and Expr. (2) which are described in Section 3.5 to determine QoC. In this experiment, the parameters of Expr. (1), they are, X, w, p, and α, were fixed and set to the following values based on the experience of preliminary experiments.

$$X = 5550,\ w = 10^3,\ p = 0.1,\ \alpha = 1000$$

We also assumed that the QoS requirement of the user is 30 fps. In addition, the system considers this situation as QoS violation and sends QoC change request to $Manager$ agent; its request value is decided by Expr. (2) when the 5-seconds average of the frame rate decreases less than 28 fps. For the video streaming with 30 fps and 15 fps, the handheld PC's cpu usage, network useage, and fps without receiving the location information are shown in Figure 8, respectively.

This prototype system can calculate the user's speed (v) dynamically and can use it to decide QoC value. However, to show the effect of the proposed scheme clearly, we set v to the fixed value as 0.1, 0.5, 1.0, and 2.0, and compared the results from each case.

5.2 Experimental Results

Figure 9 shows the experimental results. When the user moved from point A to point B, these graphs represent temporal changes of QoS (fps) of received video and QoC (update frequency per second). In each graph, fps falls rapidly on the way and afterwards rises again. This shows that the streaming video migrates from handheld PC to desktop PC. In Figure 9, (a)∼(d) are results of each case of fixing the value of v as 0.1, 0.5, 1.0, and 2.0.

In any case, when the user approached point B from point A, QoC was observed to increase. The case of $v = 0.1$ (Figure 9(a)), for example, after the user left from point A at once, QoC was once per 3 or 4 seconds, and then, QoC was updated to once per 2 seconds at about 30 seconds. After that, QoC was updated 1 time/s at about 50 seconds and the service migrated to desktop PC smoothly. In addition, the average frame rate was 29.1 fps and our proposed scheme did not affect to the QoS requirement of the user.

From a viewpoint of difference of moving speed of the user, it was observed that the more the speed increased, the higher QoC became. In case of $v = 0.5$ (Figure 9(b)), for example, after the user left from point A at once, QoC was set 1 time/s, and updated to 2 times/s at about 10 seconds. After that, QoC was updated 3 times/s at about 40 seconds. Compared with the case of $v = 0.1$ (Figure 9(a)), it is found that, the more the user approached the desktop PC, the more the update frequency of QoC increased.

Moreover, in cases of $v = 1.0$ and 2.0, we also observed QoC control behavior based on QoS monitoring. In case of $v = 1.0$ (Figure 9(c)), when 30 seconds passed after the user left from point A, QoC increased to about 3.5 times/s, but when at the point of 40 seconds, it decreased to about 2 times/s, and it increased to about 4.5 times/s again at the 45 seconds point. We can analyze this behavior as follows: fps had decreased at about 35 seconds; subsequently, *Manager* agent on the handheld PC sent the request to decrease QoC to decrease QoS (fps); and then *Manager* agent sent the request to increase QoC when fps was recovered. We can also see the same behavior occurred during 60 seconds to 90 seconds. In case of $v = 2.0$ (Figure 9(d)), the control behavior was observed within the range of 15 seconds to 22 seconds; QoC decreased to 2 times/s. From these results of QoC control based on fps monitoring, the service migration process was successfully executed at about 57 seconds.

We also show a result of the experiment in which the *fps* agent was inactivated. In this case, QoC control based on change of QoS is disabled. The result in case of $v = 2.0$ is shown in Figure 10.

Compared with the situation where fps agent alives (Figure 9(d)), QoC increased gradually and it did not decreased. Moreover, QoC started to decrease from 70 seconds. This is because QoC became 8 times/s at 40 seconds and this ex-

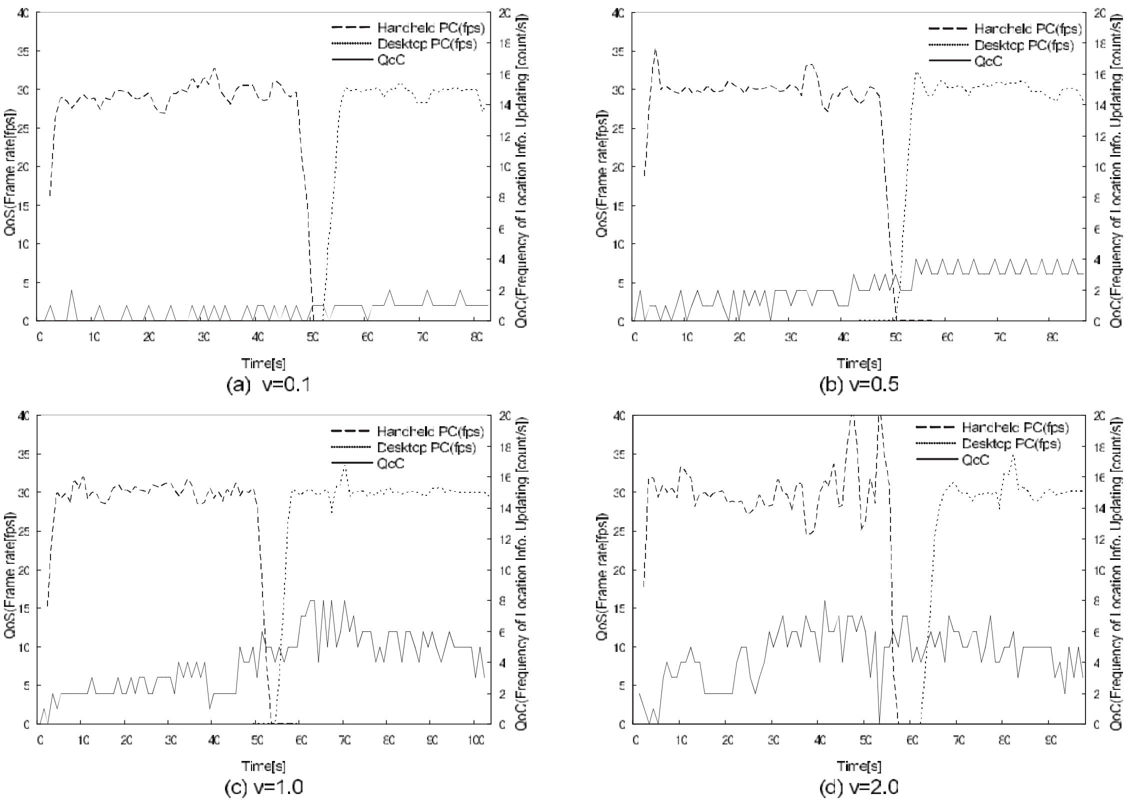

Figure 9: Experimental results of service migration experiment (v as 0.1, 0.5, 1.0, and 2.0)

cessive QoC caused the unstable frame rate. Therefore, we found that the CPU resource of handheld PC decreased and QoC processing did not catch up.

Additionally we are trying to experiment on using a data communication network of Personal Handy-phone System (PHS) as the network link of the handheld PC. Figure 11 shows the result in case of $v = 0.1$ using PHS/128kbps. In this case, we assumed that the QoS requirement of the user is 15 fps. In the same way as the case of the handheld PC used IEEE802.11g (Figure 9(a)), QoC was observed to increase when the user approach point B from point A, and our proposed scheme did not affect to the QoS requirement (15fps). This result shows that our proposed scheme can apply various network environments.

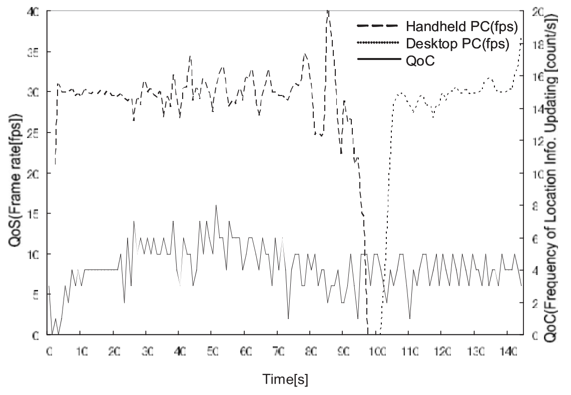

Figure 10: Experimental result in case of no fps agent available ($v = 2.0$)

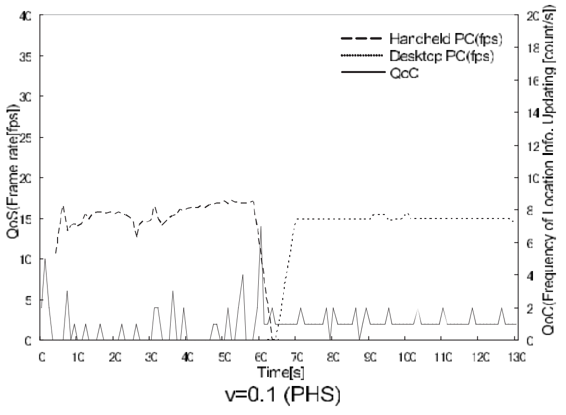

Figure 11: Experimental result in case of using PHS ($v = 0.1$)

5.3 Discussion

We show that it is possible to control QoC dynamically based on user's position and moving speed from the experimental results. We also show the problem that providing excessive QoC causes decrease in QoS depending on the user's speed. We confirmed that the system could tune QoC properly to improve QoS by the effect of our proposed QoC control scheme. In this experiment, we did not show the effect of Adaptation Function (F2); however, we give the basis of the function. Further investigation is needed to design and implement the (F2).

Here we discuss the concept of "service session". The service session means a time period when sets of service provisioning are executed in the same configuration of real space. From this classification viewpoint, the function (F1) is regarded as an adaptation in a single service provisioning; whereas (F2) is an adaptation in a single service session based on the evaluation of each single service provisioning. We have to cope with the adaptation through multiple service sessions to realize more effective QoC control. This will contribute to reduce adaptation overhead greatly.

In terms of the improvement of (F1), the existing system keeps same QoC after the user arrives at point B, because QoC control after the service migration is not considered. In the future work, we need to cope with such case when it is unnecessary to increase QoC after service migration; for instance, by suppressing the QoC value.

6 CONCLUSION

We presented a dynamic control scheme of context information delivery based on multi-agent, to develop an effective context information managing scheme for context-aware service with appropriate QoS according to user's request and changing real space. We also designed and implemented an initial experimental system. From results of experiments by the experimental system, we confirmed the effectiveness of the proposed scheme; it can manage an up-to-dateness (QoC) of positional information according to the QoS, user's position and user's speed. We can greatly improve the adaptation ability of the ubicomp space to the real space by employing our proposed scheme.

We are planning to design and develop (F2) and evaluate it. We only use fps as the QoS parameter and the up-to-dateness of location information as the QoC parameter so far using an experimental system. Our future work includes an extension of the parameter of QoS and QoC, and evaluation considering many situations such as healthcare service using vital sensor and environmental sensor. As for QoS, we will use picture size or resolution as a QoS parameter, or accuracy or granularity as a QoC parameter. Moreover we would like to advance detail definition and modeling of relation of QoS and QoC.

ACKNOWLEDGEMENT

This work was partially supported by Sendai Intelligent Knowledge Cluster and the Ministry of Education, Culture, Sports, Science and Technology, Grants-in-Aid for Scientific Research, 19200005.

REFERENCES

[1] M. Weiser, The Computer for the Twenty-first Century, Scientific American, Vol. 265, No. 3, pp. 94–104 (1991).

[2] N. Davis, K. Cheverst, K. Mitchell and A. Efrat, Using and determining location in a context-sensitive tour guide, IEEE Comput., Vol. 34, No. 8, pp. 35–41 (2001).

[3] T. Yamazaki, Ubiquitous Home: Real-life Testbed for Home Context-Aware Service, Proc. of First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05), pp. 54–59 (2005).

[4] E. D. Mynatt, A.-S. Melenhorst, A. D. Fisk and W. A. Rogers, Aware technologies for aging in place: understanding user needs and attitudes, IEEE Pervasive Comput., Vol. 3, No. 2, pp. 36–41 (2004).

[5] T. Nakamura, M. Matsuo and T. Itao, Context handling architecture for adaptive networking services, Proc. of IST Mobile Summit 2000, pp. 295–302 (2000).

[6] K. Fujinami, T. Yamabe and T. Nakajima, Take me with you! : A Case Study of Context-aware Application integrating Cyber and Physical Spaces, Proc. of ACM Symposium on Applied Computing (SAC) 2004, pp. 1607–1614 (2004).

[7] T. Buchholz, A. Kupper and M. Schiffers, Quality of context: what it is and why it need", Proc. of the Workshop of the HP OpenView University Association 2003 (HPOVUA2003) (2003).

[8] K. Sheikh, M. Wegdam and M. van Sinderen, Middleware Support for Quality of Context in Pervasive Context-Aware Systems, Proc of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07) (2007).

[9] M. C. Huebscher and J. A. McCann, An adaptive middleware framework for context–aware applications, Springer Personal and Ubiquitous Computing Journal, Vol. 10, No. 1, pp. 12–20 (2005).

[10] Y. Tokairin, K. Yamanaka, H. Takahashi, T. Suganuma and N. Shiratori, An Effective QoS Control Scheme for Ubiquitous Services based on Context Information Management, Proc. of International Workshop on NGN and their Impact on E-Commerce and Enterprise Computing(NGN-EC2) in conjunction with CEC'07 and EEE'07, pp. 619–625 (2007).

[11] S. Fujita, H. Hara, K. Sugawara, T. Kinoshita and N. Shiratori, Agent-based Design Model of Adaptive Distributed System, The International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies, Vol. 9, No. 1, pp. 57–70 (1998).

[12] H. Takahashi, Y. Tokairin, T. Suganuma and N. Shi-

ratori, Design and Implementation of An Agent-based middleware for Context-aware Ubiquitous Services, Frontiers in Artificial Intelligence and Applications, New Trends in Software Methodologies, Tools and Techniques (Proc. of the 4th SoMeT2005), Vol. 129, pp. 330–350 (2005).

[13] T. Uchiya, T. Maemura, L. Xiaolu, and T. Kinoshita, Design and Implementation of Interactive Design Environment of Agent System, Proc. of the 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2007), LNCS4570, pp. 1088–1097 (2007).

[14] Furukawa CO,. LTD., ZPS (Zone Positioning System), http://www.furukawakk.jp/products/ZPS_1.html (in Japanese)

[15] Sun Microsystems, Java Media Framework(JMF), http://java.sun.com/products/java-media/jmf/.

(Received October 15, 2008)
(Revised July 16, 2009)

Hideyuki Takahashi is a research fellow of Research Institute of Electrical Communication of Tohoku University, Japan. He received his doctoral degree in Information Sciences from Tohoku University in 2008. His research interests include ubiquitous computing and agent-based computing. He is a member of IPSJ.

Yoshihisa Sato received M. S. degree in 2009 from Tohoku University, Japan. Currently, he works for the IBM Corp. His research interests include agent-based computing and context-aware service provisioning.

Takuo Suganuma is an associate professor of Research Institute of Electrical Communication of Tohoku University, Japan. He received a Dr.Eng. degree from Chiba Institute of Technology. He received UIC-07 Outstanding Paper Award in 2007, etc. His research interests include agent-based computing, flexible network, and symbiotic computing. He is a member of IEICE, IPSJ and IEEE.

Norio Shiratori received his doctoral degree from Tohoku University, Japan in 1977. Presently he is a Professor of the Research Institute of Electrical Communication, Tohoku University. He has been engaged in research related to symbiotic computing paradigms between human and information technology. He was the recipient of the IPSJ Memorial Prize Winning Paper Award in 1985, the Telecommunication Advancement Foundation Incorporation Award in 1991, the Best Paper Award of ICOIN-9 in 1994, the IPSJ Best Paper Award in 1997, the IPSJ Contribution Award in 2007, and many others. He was the vice president of IPSJ in 2002 and now is the president of IPSJ. He is also a fellow of IEEE and IEICE.