Task-Driven Device Ensemble System Supporting Seamless Execution of User Tasks Despite Multiplexed Interruptions

Tatsuo Tomita[†], Kazumasa Ushiki[‡], Yoshiaki Kawakatsu[‡], Nobutsugu Fujino[‡], and Hiroshi Mineno[*]

[†]Fujitsu Laboratories Ltd., Japan
[‡]Human Centric Computing Laboratories, Fujitsu Laboratories Ltd., Japan
[*]Graduate School of Science and Technology, Shizuoka University, Japan
{tomita.tatsuo, ushiki.kazumasa, kawakatsu.yoshi, fujino}@jp.fujitsu.com, mineno@inf.shizuoka.ac.jp

Abstract – In the real world, multiple tasks that people conduct in their daily lives are often interrupted. In particular, when multiplexed interruptions occur while people are conducting tasks, they often forget to complete tasks that they were in the midst of accomplishing, prior to such numerous interruptions. It would be possible for people to accomplish such multiple tasks more efficiently if information and communication technologies (ICT) were leveraged to assist and support them in completing their tasks.

We have been proposing a "task-driven device ensemble system", which employs a user's handheld mobile device linked with various electronics devices available in the user's surroundings, to support execution of user tasks. We have expanded on this system to enable seamless execution of user tasks even when faced with multiple interruptions of such tasks. This paper provides an overview of requirements of our proposed system, and describes a prototype system we implemented, in addition to describing a sample case study of how the system can support retailers with their task execution. We also evaluate usability and practicality of our proposed system. Our qualitative and quantitative evaluation results verify that our proposed system satisfies the following targeted requirements and objectives, thus demonstrating that the system is sufficient for practical use.

Keywords: device ensemble, UPnP, task-driven, multiplexed interruptions, human centric

1 INTRODUCTION

We are currently conducting R&D in "Human-Centric Computing", in which information and communication technologies (ICT) subtly and unobtrusively support users, without the need for explicit user-operation. In the real world, depending on the users' real-time situations, users are often interrupted while they are in the midst of accomplishing various tasks in their daily lives – at times, if such interruptions are multiplexed, users may forget to complete some of those tasks.

For instance, it is reported that today's knowledge workers experience task interruptions on the order of every 4 to 11 minutes [1]. In addition, as illustrated in a hospital scenario in Ref. [2], nurses are assigned numerous patients and must complete routine tasks. However, if a nurse is interrupted during a task, while bearing in mind the priority of various tasks, the nurse is required to complete a multitude of tasks within limited time – if ICT could be leveraged to assist them in task management, nurses would have more time to focus on the tasks themselves as their core duties, thereby helping to prevent medical malpractice.

We have been proposing a "task-driven device ensemble system", which employs the users' handheld mobile devices such as smartphones, linked with electronic devices readily available in user surroundings, to enable greater efficiencies for task execution by individuals. In view of the aforementioned, the primary issue is how well the task-driven device ensemble system can handle multiplexed task interruption.

This paper describes a newly designed task-driven device ensemble system that operates seamlessly even with multiplexed task interruptions. The remainder of this paper is comprised as follows:

In Chapter 2 we describe related work. In Chapter 3, we present the concept of task-driven device ensemble systems, and our proposed newly designed task-driven device ensemble system. In Chapter 4, to evaluate usability of our proposed task-driven device ensemble system, we describe a prototype system that supports store clerks at retailers for consumer electronics and home appliances. In Chapter 5, we verify the results of our usability and performance evaluations. Chapter 6 describes our conclusion and future works.

2 RELATED WORK

There is already a considerable body of research addressing human behavioral support that is dependent on a person's context, or context-aware navigation systems [3]–[13], as well as research related to device collaboration. At the same time, very little work has been conducted on the two fields in combination.

Examples of work on context-aware navigation systems or human behavioral support are as follows: "Task-based mobile service navigation system" [3] employs a task model that analyzes real-world problems, thereby making it possible to search for service provider sites when a user specifies a task. iHospital [4] instantaneously supports business tasks in the real world. By providing hospital staff with Bluetooth-enabled communications units so that they can determine the location of other staff, and by sharing each other's status using mobile phones equipped with messaging capabilities, they are able to quickly respond to emergency surgeries. Wieland et al. [5] proposes and

ISSN1883-4566 © 2013 - Informatics Society and the authors. All rights reserved.

describes the implementation of a system for creating workflows based on sensor-detected contexts. WTAS [6] models tasks using a Petri net and decides which task should be performed based on the user's context as ascertained using wearable sensors. Once a task is decided, information such as maps is provided to support the execution of the task. Most of these papers focus on how to acquire the user's context and leverage it in supporting the user's tasks. Although these papers share the same objective as our work in this paper, they do not involve the linking of devices.

There have been a number of works addressing device collaboration [14]-[20], particularly using Universal Plug and Play (UPnP) [21], such as the following examples: Mets et al. [14] describes a context-aware multimedia management system for the home environment using UPnP. All content is integrated in a way that prevents the user from being aware of its location, and the MediaRenderer [22] closest to the user is automatically selected. Gashti et al. [15] makes use of UPnP proxies. In addition to enabling UPnP services (this includes fine-grained services that would ordinarily be called "functions") to be employed across subnets, each user's device and service information is registered, making it possible for users to automatically use services depending on the context. Ubiquitous e-Helper [16] is a composable UPnP-based service platform for linking among smartphones. Each of these studies uses UPnP to link between devices, and contexts. At the same time, they do not go as far as to consider interruptions in tasks.

The following are works that combine context-aware navigation and device linking: Task Computing [23] proposes a task-centric technology for supporting people in performing tasks using services in the user's vicinity. In this study, human tasks are regarded as collections of services in nearby devices. In its implementation, however, the system first collects nearby services and then displays available tasks to the user based on these services. By contrast, in the system we propose, tasks are defined based on the user's context and then services on nearby devices are collected to execute those tasks. To add to this, their research is focused on an ontology for service linking. Bidot et al. [24] proposes, and then discusses the implementation and evaluation of, a workflow-management system in which workflows are created based on contexts and devices are controlled according to workflows. This system involves device ensembles. For example, to achieve the goal of helping a user relax at home, the system will control the lighting, as well as the air conditioning and a music player. UPnP is employed for this, and the Device Manager (DM) exists on household and other networks. This approach does not, however, take into account linking between the user's handset and nearby devices when the user moves around. While these works share the same goal as our research, they differ in their implementation from the task-driven device ensemble (device collaboration) that we have originally proposed.

Each of these works is focused on collecting contextual information and suggesting tasks for the user. In contrast, we are focused on what happens after a task is executed, i.e. multiplex interruptions that occur during a task and the recovery from these interruptions. These works also employ an ontology, whereas we do not. Generally speaking, the cost of constructing and maintaining an ontology is quite high, and therefore it can be said that our approach is more practical than these works. Table 1 illustrates the characteristics of our approach and these works.

Table 1: Comparison between our approach and the related works

Characteristic	Our Approach	Masuoka et al. [23]	Bidot et al. [24]
Context awareness	○	○	○
Device collaboration (with user's handheld mobile device)	○ (○)	○ (○)	○ (×)
User task definition	○	×	○ (+ workflow)
Support for multiplexed interruptions	○	×	×
Use of ontology	×	○	○

3 TASK-DRIVEN DEVICE ENSEMBLE

3.1 Concept behind the Task-Driven Device Ensemble

To support users in executing tasks, we have proposed a task-driven device ensemble in which the task at hand is first determined by the user's context, and then devices necessary for that task are discovered and are linked together. Figure 1 illustrates the concept behind the task-driven device ensemble.

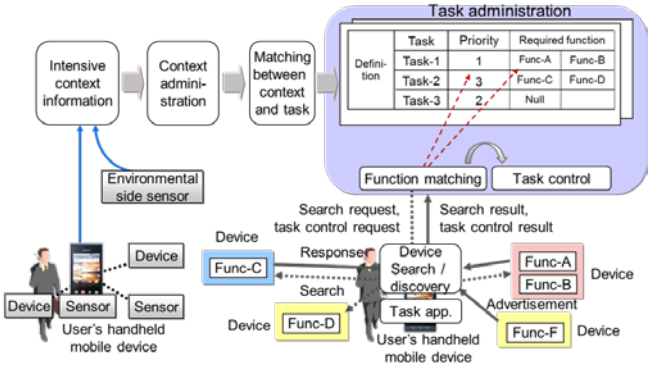

Figure 1: Concept behind the task-driven device ensemble

We presume that users will always be carrying with them handheld mobile devices such as smartphones equipped with multiple sensors. In the cloud, this sensor data can be combined with other environmental sensor data and abstracted in the user's context.

The task that best matches the user's context is selected. A task administration mechanism manages the priority of each task and the functions required to execute tasks. Tasks for users to perform are defined in advance by the users themselves, or particularly in the case of fixed job tasks, by

a field manager. The task manager searches for devices in the user's vicinity that feature functions needed to perform a task - if all the functions are discovered, the task is selected, devices that have the required functions are linked, and the task is executed.

If multiple tasks can be executed given the functions in the devices, the highest priority task is selected and executed. If the task can be performed on the user's handheld mobile device itself even without being linked to any nearby devices, there is no need for requested functions for the task. In such a case, the requested functions in the task definition table shown in Fig. 1 are defined as null.

3.2 Proposed Task-Driven Device Ensemble

We propose a task-driven device ensemble that allows multiplexed interruptions, in which it possible for the user to resume a previous task after encountering multiple task interruptions. In the workplace, in particular, there is typically a flow for tasks. At the same time, user task flows do not simply proceed in a linear order. Taking into consideration the priority of the tasks at hand, users may occasionally be interrupted during a task in order to perform a new task, only to return to the previous task upon completing the interrupting task.

To implement such a system, we expanded a basic task-driven device ensemble system, by adding a task state administration function to the task administration server. Figure 2 illustrates the task administration function added to the task administration server. If, during the execution of Task-2, a higher priority Task-1 arrives, the already-running Task-2 is put into interrupt mode and pushed onto the interrupt stack. Once Task-1 is complete, Task-2 is popped off the stack. This, in turn, enables multiplexed interruptions to be handled correctly.

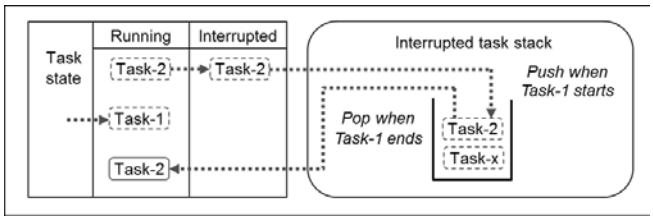

Figure 2: Task state administration for handling multiplexed interruptions

Figure 3 outlines the sequence whereby an already-running application for Task-2 is interrupted by an application for Task-1.

(1) When the start request message for Task-1 arrives, the task control compares the priority of Task-1 with the priority of Task-2, which at that moment is running on the user's handheld mobile device (e.g. smartphone). Because Task-1 has a higher priority, the sequence proceeds to step 2.
(2) The task control causes the function matching unit to determine whether the functions required by Task-1—that is, Func-A and Func-B—can be supported by the devices nearby the user's handheld mobile device.
(3) The function matching unit sends a search request message to the user's handheld mobile device's device search/discovery unit.
(4) To determine whether nearby devices can support the functions required by Task-1, the device search/discovery unit multicasts a UPnP search message to nearby devices.
(5) Devices that are able to offer any of the needed functions respond to the search request message.
(6) The device search/discovery unit sends the search results received from the devices to the function matching unit.
(7) The function matching unit checks if Task-1 can be executed by those devices. In this case, it decides that Task-1 can be executed, and it notifies the task control of this result.
(8) Before starting the Task-1 application, the task control sends a suspend request message to the task execution control on the user's handheld mobile device in order to suspend the Task-2 application.
(9) Upon receiving the suspend message, the task application execution control suspends the Task-2 application.
(10) The task application execution control sends a reply message informing of the Task-2 application's suspension.
(11) After the Task-2 application has been suspended, the task control requests the start of the Task-1 application.
(12) The task application execution control starts the Task-1 application.
(13) Task-1 is executed using device collaboration with the devices near the user.
(14) Upon the completion of Task-1, the Task-1 application notifies the task application execution control of the completion.
(15) The task application execution control sends the task control notice that Task-1 is complete.
(16) The task control requests that the Task-2 application, which had been suspended by the Task-1 application, be resumed.
(17) The task application execution control resumes the suspended Task-2 application.

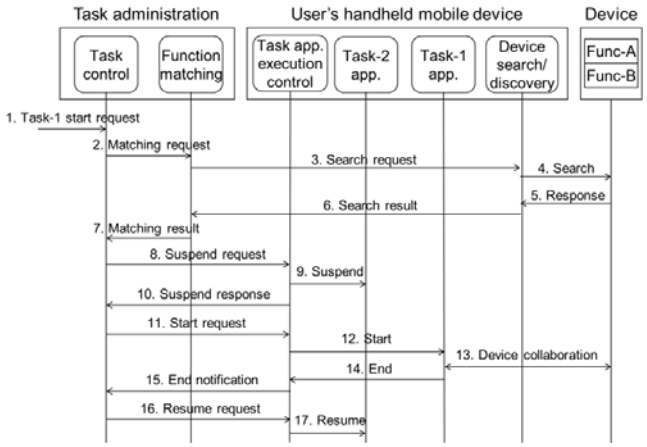

Figure 3: Sequential chart of task interruption control

The interworking between the task administration server and the user's handheld mobile device enables automated task switching, thus preventing users from forgetting to accomplish tasks, without additional burden to the user. As a user support system, it is ideal if user intervention can be kept to a minimum to minimize erroneous user operation, thus enabling appropriate task switching via the user support system.

3.3 Benefit of the Proposed Task-Driven Device Ensemble

The most prominent distinctive characteristic of our proposed concept is the comprehensive integration of context awareness, device collaboration and multiplexed task interruption. This comprehensive integration is realized in the following sequence: In addition to the user context (e.g. user location), device context (i.e. presence of the device in the user's vicinity and its functions) are identified in the cloud. Executable tasks and their priorities are identified according to these contexts along with task definitions, and multiplexed interruption control based on the highest-priority task is executed for the user's handheld mobile device. The task is executed via collaboration between the user's handheld mobile device and nearby devices.

Device collaboration execution is enabled only when a device equipped with functions required to execute the task is available and present nearby the user – it can be difficult for the user to discern the appropriate timing for execution of device collaboration. In our proposed concept, by integrating device collaboration with context awareness, and leveraging the cloud to recognize device context and to request the user's handheld mobile device to execute the appropriate task when device collaboration required for the task is possible, we enable the task to be executed with suitable timing, while eliminating the need for the user to be aware of device collaboration feasibility.

During a particular task execution, if a separate higher-priority task needs to be executed, the integration of context awareness and multiplexed task interruptions enables the system to prompt the user to execute the higher-priority and lower-priority tasks at their appropriate timing. In such a case, the data for the interrupted lower-priority task is preserved and stored, and execution of the lower-priority task resumes after completion of the higher-priority task, thus preventing the user from forgetting to complete the lower-priority task.

As aforementioned, device collaboration is possible only when a collaborative device is available and present nearby the user. Therefore, if the collaborative device becomes unavailable (e.g. if the device is switched off) during the task execution, the task execution cannot be completed: In this case, the task should be suspended, and meanwhile any other existing tasks that are executable should be executed and completed. This task control (i.e. task suspension and execution) that accommodates changes in device context during device collaboration is realized by the multiplexed interruption mechanism, and hence can be said to be a benefit resulting from the integration of device collaboration with multiplexed task interruption.

4 PROTOTYPE SYSTEM

We developed a prototype system based on the concept described in the preceding chapter, in order to evaluate its usability and practicality. This chapter is comprised of a description of the system design, implementation and a service scenario.

4.1 System Requirements

Requirements and quantitative objectives for the prototype system were set as follows:

(1) Management of multiplexed task interruptions and resumption
As described in the previous chapter, for the purpose of supporting human tasks, the system should be able to manage multiplexed interrupted tasks and then enable suspended tasks to be resumed.
As a numeric objective, in view of the system's practical use, it should be able to handle 20 multiplexed interruptions which is anticipated will be sufficient for use at nearly any feasible worksite or field.

(2) Real-time task processing
For reasons of usability and to enable a user-friendly and stress-free experience for users, the time between user operations and the delivery of device collaboration results should be within 2.0 seconds, comparable to the minimal average latency experienced when a TV is turned on by remote control.

(3) Scalability for simultaneous task execution
The task administration server handles numerous users' handheld mobile devices, and processes the tasks of these handheld mobile devices simultaneously. Therefore, scalability is important. As a numeric objective, in light of practical considerations, each server should have a capacity of handling 10,000 users' handheld mobile devices.

For the system requirements and objectives, we have designed an implementation structure as described in the following section.

4.2 Implementation

Figure 4 illustrates the structure of the prototype system and Table 2 outlines the system's hardware specifications. The system has been implemented using C++ for the nearby devices and Java for everything else.

On the server side, the system consists of a context administration server and task administration server, and the device side consists of user's handheld mobile devices and nearby devices. Each of these is discussed below.

Context administration server

The role of the context administration server is essentially to derive the user's context from information collected by the user's handheld mobile device and sensors in the user's environment. With this said, our research is primarily

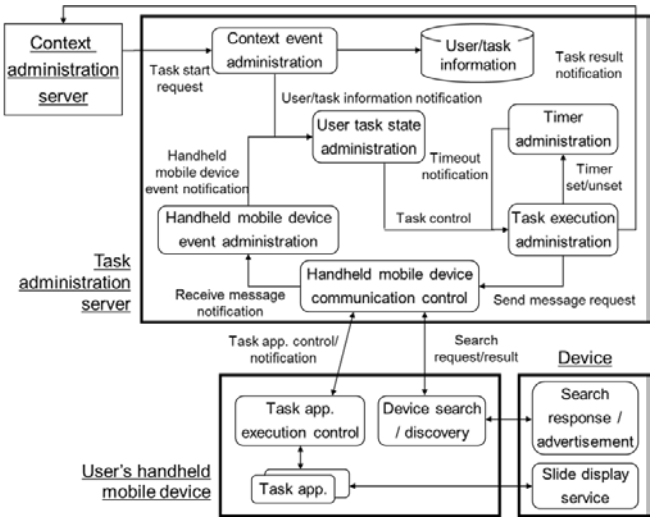

Figure 4: Prototype system configuration

Table 2: Hardware specifications of prototype system

Equipment Name	Specification
Context administration server	CPU: Xeon 2.4GHz×2, Memory: 1GB, OS: Fedora
Task administration server	CPU: Xeon 3.06GHz, Memory: 1GB, OS: Windows Server 2003
User's handheld mobile device	Android smartphone, OS: Android 2.1
Device (notebook PC)	CPU: Celeron 1.2GHz, Memory: 2GB, OS: Windows XP Professional SP3

focused on the task administration server. Therefore, for the purpose of this study we have implemented a pseudo-context administration server that only sends task start requests and receives task execution result notifications.

Task administration sever

The task administration server executes tasks at the request of the context administration server and sends acknowledgement of task execution results to the context administration server. The functions of the server are as follows:

(1) Context event administration
Upon receiving an event from the context administration server, this function will employ user information and task information to determine which users and tasks relate to the event. It will then send a notification to the user task state administration function containing information about the user and the executable task. To satisfy system requirement (3), a thread for processing the events is generated in advance for each user. This enables reduction of thread creation overhead that causes performance degradation when the tasks of numerous users' handheld mobile devices are processed simultaneously. This also has a great effect on real-time task processing of system requirement (2).

(2) User task state administration
In accordance with each of the server's event administration functions, this function manages each user's task execution state, i.e. whether a user is executing a task or waiting to execute a task. To satisfy system requirement (1), it deploys the task state administration mechanism that was described in Figs. 2 and 3 of Section 3.2. This will ensure that users can resume suspended tasks without fail, even when there are multiple interrupting tasks.

(3) Task execution administration
This function receives instructions from the user state administration function, issues search requests for functions needed by users' handheld mobile devices, and issues requests for the execution, interruption and resumption of task applications.

(4) Handheld mobile device event administration
This function awaits events from user's handheld mobile devices. Upon receiving an event, it will notify the user state administration function of the event.

(5) Handheld mobile device communication control
This function controls communications with users' handheld mobile devices and sends/receives messages.

(6) Timer administration
This function manages timers when various kinds of requests are resent by the task execution administration function.

Users' handheld mobile devices

Android smartphones were used for the users' handheld mobile devices, and UPnP was employed for controlling the devices and searching for functions available on the devices and user handheld mobile devices. We used a UPnP library developed by Fujitsu for use with Android. The functions of the prototype user handheld mobile devices are as follows:

(1) Task applications
These are applications intended to support the execution of tasks. We developed applications based on a test scenario for supporting sales clerks at an electronics retail store. Details of the scenario are described in section 4.3.

(2) Task application execution control
This function awaits events from the user state administration function. Upon receiving an event, it will execute, suspend, or resume a task application, and then acknowledge the result.

(3) Device/service search
This function will use UPnP's M-SEARCH to search for devices or UPnP services (functions for executing tasks) that have been specified by the task administration server.

Nearby devices

For this prototype, we employed a notebook PC as a nearby device. The implemented features are as follows:

(1) Search response/advertisement
Search response will respond to the user handheld mobile device if the searched service (functions) exists on the device. Advertisements will periodically multicast the services offered by the device.

(2) Slide display service
In terms of the UPnP services on our prototype, we only developed a slide display service.

4.3 Service Scenario

As illustrated in Table 3, we developed a task application to support sales clerks working at an electronics retail store. Because the purpose of our study was to perform a basic test with multiple task interruptions, we selected a relatively simple scenario. Even for more complicated scenarios, the process during multiplex interruptions will remain basically the same.

We envision a work support flow for these applications as follows:

(1) Store clerk A is replenishing merchandise (default state).
(2) If a customer visits the store, the customer care task application will interrupt clerk A's merchandise replenishment job. We assume that each customer can be identified by means of a store membership card, etc. Clerk A operates the customer care task application to access the customer's information and then serves the customer.
(3) If there is a display device available, such as a notebook PC, that can be used to explain a product, the task application for product explanation is executed and the previous customer care task application is interrupted. Clerk A operates the current product explanation task application and then gives an explanation using product sales slides shown on the nearby display device.

Table 3: Task applications of prototype system

Application Name	Device Collaboration	Overview
Merchandise replenishment	×	The staff can confirm the number of sales items to be replenished.
Customer care	×	The staff is notified that there is a customer to be taken care of.
Product explanation	○	The staff can explain the sales items by displaying description slides on a nearby PC display.

5 EVALUATION

We evaluated the prototype system for usability and practicality. This chapter discusses the system's qualitative and quantitative evaluations.

5.1 Qualitative Evaluation

We confirmed that the implemented prototype operates with accuracy. Figure 5 is a screenshot of an actual user's handheld mobile device. It displays the various phases of the system's operation. First, store clerk A is executing the merchandise replenishment task. Second, a customer arrives at the store, and the customer care task interrupts the previous task. Third, store clerk A serves the customer, and the explanation task interrupts the previous task. Last, the interrupted tasks resume in sequential order.

This demonstrates that the task applications were executed properly and that transitions with multiple task interruptions worked well.

As we focus on task administration in this research, we verified that the basic mechanism of the task administration was realized. Specifically, we confirmed multiplexed interruption coupled with nearby device discovery required for a task execution, resumption of the interrupted lower-priority task after completion of the higher-priority task, and task execution through collaboration between the user's handheld mobile device and the nearby device.

At this stage, we have not yet implemented functions necessary for the process prior to the task administration illustrated in Fig. 1 – in other words, sensor data collection via the sensors in the user's handheld mobile device, generation of user context based on sensor data, and matching between the user's context and task execution conditions. We intend hereafter to implement these functions and evaluate the practicality of our proposed concept in its entirety.

(a)Merchandise replenishment (b) Customer care (c) Product explanation

Figure 5: Screenshots of user's handheld mobile device in prototype system

5.2 Quantitative Evaluation

In terms of quantitative objectives, we evaluated the prototype system for the number of multiplexed interruptions, processing time, and scalability.

Number of multiplexed interruptions

We estimated the number of possible multiplexed interruptions for the system. To accomplish this, we measured the elapsed time for multiplexed task interruptions (namely, the processing time required for interrupting and resuming) for each number of concurrent multiplexed interruptions. We then measured the time interval from when the pseudo-context administration server sends a new task execution notification, through the interruption of the previous task by the new task, and up until the screen of the newly executed task is displayed. 5 measurements were taken and averaged for each number of concurrent multiplexed interruptions from 2 to 20. Figure 6 shows these measurement results.

This graph shows an average time between 1.9 to 2.3 seconds for the entire range of measurement, despite variations in time intervals resulting from fluctuations in the handheld mobile device load due to wireless network traffic and other factors. These results indicate that, for up to at least 20 interruptions, the number of multiplexed interruptions does not have an impact on processing time. Therefore, the maximum number of multiplexed interruptions is at least 20. This, in turn, satisfies the quantitative objective for system requirement (1).

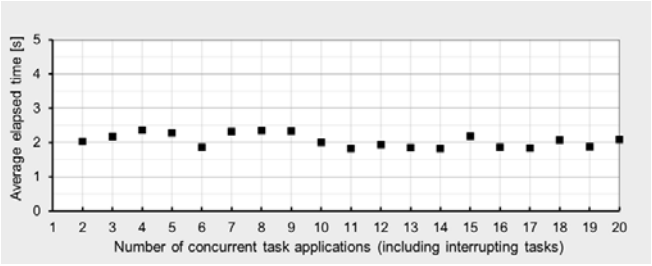

Figure 6: Elapsed time for task interruption

Processing time

We measured processing time in terms of usability. For this purpose, 2 kinds of time were measured: (1) the amount of time for handheld mobile device-side device collaboration to have an impact on practical usability; and (2) server-side task event processing time.

With respect to (1), we evaluated the operability of a task application running on the user's handheld mobile device. To do so, using the product explanation task, we measured the time elapsed between when the user presses the slide control button and when the result of the designated slide control is actually displayed on the PC through the device collaboration.

Table 4 shows the average elapsed time after performing this operation 5 times.

Although the slide show start time includes the time required for Adobe Acrobat Reader to launch and is therefore longer than other operations, all of the operations run in a sufficiently short enough amount of time. This confirms that the device collaboration mechanism works for users without any stress, and that this satisfies the quantitative objective for system requirement (2).

Table 4: Elapsed time for device collaboration

Operation	Elapsed Time (sec)
Start slide show	2.0
Change slide	0.3
End slide show	0.4

For (2), we evaluated the task-related event processing time of the task administration server, i.e. the server response time when responding to context changes, such as when task execution becomes possible, from the context server, as well as the task administration server response time when responding to operations on users' handheld mobile devices. To do so, we measured the processing time of the task administration server while gradually increasing its processing load. The specific evaluation criteria are outlined below:

(1) Task execution without interruption:
Under the condition that there is no task application running in the user's handheld mobile device, the time between the pseudo-context administration server requesting the execution of the merchandise replenishment task and the task administration server requesting that the user's handheld mobile device executes the task.

(2) Task execution with interruption:
Under the condition that the merchandise replenishment task application is running in the user's handheld mobile device, the time between the pseudo-context administration server requesting the execution of the customer care task and the task administration server requesting that the user's handheld mobile device executes the task.

(3) Task resuming:
The time between receiving notice of the completion of the customer care task from the user's handheld mobile device, up through notifying completion of the task to the pseudo-context administration server and sending a request to the user's handheld mobile device in order to resume the merchandise replenishment task.

Figure 7 shows the average time for each of the above processes, each of which was measured 30 times.

Each processing time displays a slight upward trend on account of increased server load, but even the longest time was only 130 milliseconds. Therefore, it can be said that task execution and handheld mobile device processing can be performed in almost real time.

This satisfies system requirement (2). Discrepancies between the processing times are considered to be due to differences between each of the executed processes. This graph also demonstrates that task interruptions require CPU power and that resuming tasks consumes the largest amount of CPU power.

If searching for devices or services (functions) is performed prior to the execution of a task, the search time should be added to (1) or (2) in Fig. 7. According to UPnP specifications, the wait time for M-SEARCH must be greater than or equal to 1 second and should be less than 5 seconds inclusive. As a result, this wait time is dominant, and in consideration of usability, it should be set to 1 second.

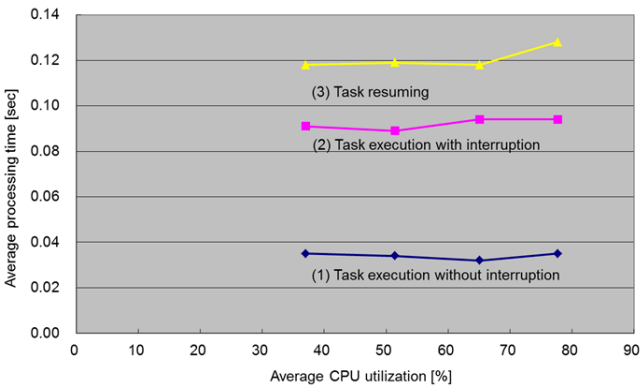

Figure 7: Task event processing time of prototype system

Scalability

We evaluated the prototype system for scalability. For this purpose, we estimated (1) the capacity of the task administration server, and (2) the processing performance of the task administration server.

With respect to (1), our prototype system consumes exclusively 1 thread per user. Therefore, the system capacity is equivalent to the number of threads that can be

simultaneously generated on the server. In the case of our prototype system, this number was 12,000.

For (2), we measured the number of tasks that the task administration server was able to process in one hour while gradually increasing the server processing load. We defined the maximum server capacity to be the largest number of tasks possible before CPU utilization reached 80%. We measured processing performance for 2 scenarios: 1) The simplest single-task scenario, and 2) the most complicated 3-task scenario. The single-task scenario only executed the merchandise replenishment task from Table 3. The 3-task scenario executed the 3 tasks and interruptions in the order listed in Fig. 5.

The results are shown in Figs. 8 and 9. The number of tasks processed per hour was approximately 760,000 for the single-task scenario and 670,000 for the 3-task scenario. In both scenarios, only 1 task is processed at a time, so if a user were to perform each task in an average of 3 minutes, for a single user it would be possible to process 20 tasks in an hour. Therefore, the potential system capacity would be approximately 38,000 users for single-task scenarios and 33,000 users for 3-task scenarios. This difference is considered to be due to the additional overhead required for task interruptions and resuming.

The previous capacity estimate of 12,000 user threads is thought to be due to limitations in the settings of the Java programming language, which is used on the server. In any case, these results satisfy the quantitative objective for system requirement (3) of our prototype system.

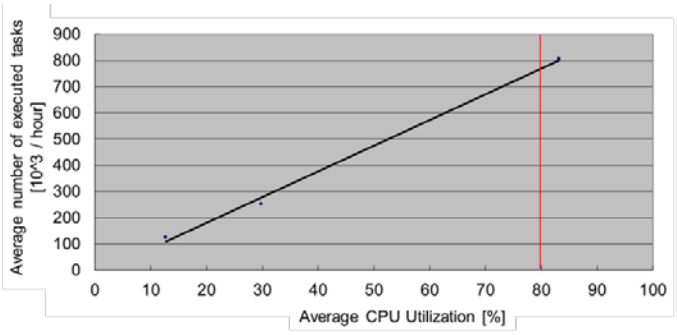

Figure 8: Task processing capacity for single-task scenario

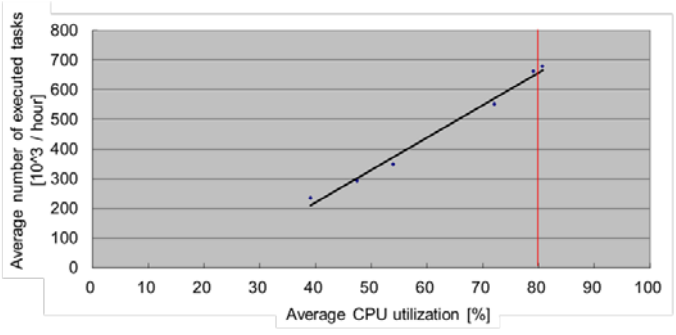

Figure 9: Task processing capacity for 3-task scenario

6 CONCLUSION

We proposed an expanded “task-driven device ensemble system” that supports user behavior, via seamless execution of user tasks despite multiplexed interruptions. We also implemented a prototype system envisioned to support store clerks at retailers, and evaluated the prototype system to verify that it indeed operates with precision as intentionally designed. Our quantitative evaluation results were: (1) Seamless execution of user tasks even with at least 20 multiplexed interruptions, (2) Real-time processing within 2.0 seconds via linked devices, and task processing time of less than 130 milliseconds in the task administration server, (3) Scalability of system capacity for at least 12,000 users per server. Our results verified that the implemented prototype system satisfied our requirements and objectives, and is sufficient for practical use.

In our future works, we aim to achieve the following: (1) Increase system capacity, (2) System expansion to include operability with non-UPnP devices, (3) Conduct user-derived/user-centric evaluation. For (1), although the system capacity of our current prototype system is dependent on and limited by the number of user threads that the system can simultaneously generate, to enable practical use, in future works we will eliminate this limitation by dynamically allocating one of the pooled threads to the requested event processing as needed. In regards to (2), we will target system operability that includes linking with non-UPnP devices that are widely available. Regarding (3), although for this work our evaluations were primarily to verify our prototype system performance, in future works we intend to evaluate user experience and efficacies through field trials.

REFERENCES

[1] L. Dabbish, G. Mark, and V. Gonzalez, ``Why Do I Keep Interrupting Myself?: Environment, Habit and Self-Interruption,'' ACM Conference on Human Factors in Computing Systems, pp.3127-3130 (2011).

[2] A. Terada, T. Takaya, R. Nishino, M. Iida, E. Sato, M. Matsutani, Y. Hirabayashi, Y. Sakyo, T. Ibe, N. Matsuzaki, Y. Murakami, and M. Momoi, ``Anticipating Professional Nursing Practice: Trial and Evaluation of a Bridge Program for Graduating Students Part 3 -Multitasking Scenario Exercises-,'' Journal of St. Luke’s Society for Nursing Research, Vol.12, No.2, pp. 58-64 (2009).

[3] Y. Fukazawa, T. Naganuma, K. Fujii, and S. Kurakake, ``Proposal and User Evaluation of Enhanced Task-based Mobile Service Navigation System,'' Information Processing Society of Japan (IPSJ) Journal, Vol.50, No.1, pp.159-170 (2009).

[4] T. R. Hansen, J. E. Bardram, and M. Soegaard, ``Moving out of the Lab: Deploying Pervasive Technologies in a Hospital,'' IEEE Pervasive Computing, Vol.5, No.3, pp.24-31 (2006).

[5] M. Wieland, P. Kaczmarczyk, and D. Nicklas, ``Context Integration for Smart Workflows,'' Sixth Annual IEEE International Conference on Pervasive Computing and Communications, pp.239-242 (2008).

[6] S. Xiahou, and X. Xing, ``The WTAS Framework: A Petri net based wearable task assistance system,'' Proceedings of the 2nd International Conference on

Information Science and Engineering, pp.2487-2490 (2010).
[7] D. Cheng, H. Song, H. Cho, S. Jeong, S. Kalasapur, and A. Messer, ``Mobile Situation-Aware Task Recommendation Application,'' Proceedings of the Second International Conference on Next Generation Mobile Applications, Services and Technologies, pp.228-233 (2008).
[8] A.Terracina, S. Beco, T. Kirkham, J. Gallop, I. Johnson, D. Randal, and B. Ritchie, ``Orchestration and Workflow in a mobile Grid environment,'' Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops, pp.251-258 (2006).
[9] M. Michou, A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis, ``A Semantics-Based User Model for the Support of Personalized, Context-Aware Navigational Services,'' Proceedings of the First International Workshop on Ontologies in Interactive Systems, pp.41-50 (2008).
[10] F. Tang, M. Guo, M. Dong, M. Li, and H. Guan, ``Towards Context-Aware Workflow Management for Ubiquitous Computing,'' Proceedings of the International Conference on Embedded Software and Systems, pp.221-228 (2008).
[11] Z. Chen, Z. Shao, Z. Xie, and X. Huang, ``An attribute-based scheme for service recommendation using association rules and ant colony algorithm,'' Proceedings of the Wireless Telecommunications Symposium, pp.1-6 (2010).
[12] D. Bouneffouf, A. Bouzeghoub, and A. Gancarski, ``Following the User's Interests in Mobile Context-Aware Recommender Systems: The Hybrid-e-greedy Algorithm,'' Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, pp.657-662 (2012).
[13] A. Seetharam, and R. Ramakrishnan, ``A context sensitive, yet private experience towards a contextually apt recommendation of service,'' Proceedings of the 2nd International Conference on Internet Multimedia Services Architecture and Applications, pp.1-6 (2008).
[14] K. Mets, J. Nelis, D. Verslype, P. Leroux, W. Haerick, F. De Turck, and C. Develder, ``Design of a Context Aware Multimedia Management System for Home Environments,'' Proceedings of the Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp.49-54 (2009).
[15] S. Gashti, G. Pujolle, and J. Rotrou, ``An UPnP-based context-aware framework for ubiquitous mesh home networks,'' Proceedings of the IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications, pp.400-404 (2009).
[16] J. Zao, Y. Liu, M. Yang, S. Li, W. Chen, C. Chen, K. Huan, J. Hu, and L. Kuo, ``Ubiquitous e-Helpers: An UPnP-based home automation platform,'' Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp.3682-3689 (2007).
[17] K. Togias, C. Goumopoulos, and K. Achilles, ``Ontology-Based Representation of UPnP Devices and Services for Dynamic Context-Aware Ubiquitous Computing Applications,'' Proceedings of the Third International Conference on Communication Theory, Reliability, and Quality of Service, pp.220-225 (2010).
[18] J. Ding, Y. Sheng, C. Tu, C. Huang, and J. Su, ``The Management of Device Group for Home Automation Network,'' Proceedings of the Fifth International Conference on Digital Telecommunications, pp.44-47 (2010).
[19] Y. Cui, and H. Lee, ``Method of Device Matching for QoS Based UPnP Framework in Cloud Computing Service,'' Proceedings of the First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering, pp.222-227 (2011).
[20] E. U. Warriach, E. Kaldeli, J. Bresser, A. Lazovik, and M. Aiello, ``Heterogeneous device discovery framework for the Smart Homes,'' Proceedings of the IEEE GCC Conference and Exhibition, pp.637-640 (2011).
[21] UPnP Forum, http://www.upnp.org/.
[22] UPnP Forum, MediaRenderer: 3 Device, http://upnp.org/specs/av/UPnP-av-MediaRenderer-v3-Device.pdf (2010).
[23] R. Masuoka, B. Parsia, and Y, Labrou, ``Task Computing -The Semantic Web meets Pervasive Computing-,'' Proceedings of the Second International Semantic Web Conference, pp.866-881 (2003).
[24] J. Bidot, C. Goumopoulos, and I. Calemis, ``Using AI Planning and Late Binding for Managing Service Workflows in Intelligent Environments,'' Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications, pp.156-163 (2011).
[25] K. Ushiki, T. Tsunoda, Y. Kawakatsu, N. Hasegawa, and N. Fujino, ``Development and Evaluation of Task Driven Device Orchestration System for User Task support,'' IPSJ SIG Technical Report, Vol. 2011-MBL-58, No. 21, pp.1-6 (2011).
[26] Y. Kawakatsu, K. Ushiki, T. Tsunoda, N. Hasegawa, and N. Fujino, ``Development and Evaluation of Task Driven Device Orchestration System for User Work support,'' Proceeding of the FIT2011, No.4, M-019, pp.309-310 (2011).
[27] JAVA, http://www.oracle.com/jp/technologies/java/overview/index.html

(Received October 19, 2012)
(Revised December 3, 2012)

Tatsuo Tomita is President of Fujitsu Laboratories Ltd. He received a Bachelor of Science (B.S.) in science from the University of Tokyo in 1972. He joined Fujitsu Limited in 1973, and held various positions in computer-related groups for 32 years. In 2005, he served as

Corporate Vice President and President, Mobile Phones Business Unit. In 2007, he was named Corporate Senior Vice President and President, System Products Business Group. In 2008, he became a Member of the Board, and Corporate Senior Executive Vice President in charge of Fujitsu's Product Business Group. In April 2010, he was appointed President of Fujitsu Laboratories, the key R&D organization for Fujitsu Limited. His industry-wide activities in Japan include the following: a member of the Sub-Committee on Planning, Committee on Industrial Technology of the Keidanren of Japan, a member of the Steering Board for the Tsukuba Innovation Arena (TIA) of Japan. He is a member of the Information Processing Society of Japan (IPSJ).

Kazumasa Ushiki received a B.E. and M.E. in information engineering from Shizuoka University in 1989 and 1991, respectively. He joined Fujitsu Laboratories Ltd. in 1991. His research interests include application service control architecture for mobile and ubiquitous network. He received the IEICE Young Investigators Award in 1998. He is a member of IEICE.

Yoshiaki Kawakatsu received a B.E. in electrical engineering from University of Miyazaki in 1990. He joined Fujitsu Kyushu Communication Systems Limited (currently, Fujitsu Kyushu Network Technologies Limited) in 1990. He was transferred to Fujitsu Laboratories Ltd. in 2010. His research interests include mobile computing and sensing technology. He is a member of IEICE.

Nobutsugu Fujino received a B.S. and M.E. in electronics engineering from Osaka Prefecture University in 1984 and 1986, respectively. He joined Fujitsu Laboratories Ltd. in 1986. Since then he has been engaged in radio communication systems and mobile computing, and is currently a research manager of human-centric computing and multi device interaction technology. His research interests include mobile and ubiquitous computing and network applications. He received the IPSJ Industrial Achievement Award in 2003. He received a Ph.D. in informatics from Shizuoka University in 2008. He is a member of IPSJ.

Hiroshi Mineno received his B.E. and M.E. from Shizuoka University, Japan in 1997 and 1999, respectively. In 2006, he received a Ph.D. in Information Science and Electrical Engineering from Kyushu University, Japan. Between 1999 and 2002 he was a researcher in the NTT Service Integration Laboratories. In 2002, he joined the Department of Computer Science of Shizuoka University as an Assistant Professor. He is currently an Associate Professor. His research interests include sensor networks as well as heterogeneous network convergence. He is a member of IEEE, ACM, IEICE, IPSJ and Informatics Society.