# A Task Decomposition Scheme for Context Aggregation in Personal Smart Space

Hoseok Ryu[1], Insuk Park[1], Soon J. Hyun[1], and Dongman Lee[1]

[1] School of Engineering, Information and Communications University,
119, Munjiro, Yuseong-gu, Daejeon, 305-732, Korea
{hsryu, ispark, shyun, dlee}@icu.ac.kr

**Abstract.** In context-aware computing, the context aggregation is an important function of the context management. In an infrastructure-based smart space, a centralized context management system need not concern about its resource consumption for context aggregation. However, in a personal smart space which consists of only resource-constrained mobile devices, not only global resource consumption of the personal smart space but also that of the device which plays a role of a context manager (coordinator) must be minimized. In this paper, we propose a task decomposition scheme in which heavy context aggregation tasks to be imposed on a centralized coordinating device are decomposed and distributed to all the participating mobile devices (clients) in a mobile smart space. By decomposing and distributing the heavy aggregation operations the processing overhead upon the coordinating device can be minimized while providing equivalent context aggregation capability for applications, but maintaining the total amount of processing of all devices not to be significantly increased.

**Keywords:** Ubiquitous computing, Context awareness, Personal smart space, Context aggregation, Task distribution

## 1 Introduction

In recent years, most existing context-aware services are offered in an infrastructure-based smart space like a smart home or an office. A centralized context management system on a powerful, resource-rich machine gathers, processes, aggregates, and disseminates the context information. Context-aware services request and get notified of context information from the centralized context management system.

As a user carries several mobile devices, it composes a personal area network (PAN). The PAN environment with context management configures a new type of smart space, called a personal smart space [1], [2]. Each personal smart space includes a coordinator device and zero or more client devices. Each device may include several sensors and corresponding context providers which capture context information from them. Additionally, a coordinator device which has relatively more resource than client devices provides additional capabilities such as rule-based context aggregation, and managing context information and a list of context providers.

A context aggregation method such as logic inference requires high resource consumption [4], [5]. If a coordinator device is wholly responsible for context aggregation in a personal smart space, its processing overhead is significantly increased and its battery is exhausted. As a result, the personal smart space can last no longer. To solve this problem, the distribution of context aggregation has been proposed. By distributing the aggregation function, EDCI [6] focuses on reducing processing time taken in context reasoning, and Solar [7] focuses on increasing the reusability of existing context providers. However, none of them considers mobile ad-hoc settings. A recent work presents a middleware for context provisioning in a mobile environment which consists of resource-constrained devices. However, it does not consider resource consumption caused by context aggregation [3].

In this paper, we propose an efficient context aggregation scheme which avoids overburdening the coordinator device with context aggregation by distributing sub tasks of high level context in the client devices. The proposed scheme decomposes an aggregation task into several sub tasks based on the placement of the context providers on mobile devices in order for the sub tasks not to incur the wasteful network transmission. The evaluation results shows that the processing overhead of the coordinator device decreases about 50 percent while maintaining that of each client device is increased by only 5 percent, comparing with the total amount of processing in previous work.

The rest of the paper is organized as follows. Section 2 explains the motivation of the proposed scheme. Section 3 introduces the requirements for context aggregation in personal smart space. We discuss design consideration and describe the context management architecture for a personal smart space in Section 4. The implementation details of the proposed scheme are described in Section 5. Section 6 shows the performance analysis of our approach. The related work is presented in Section 7. Finally, conclusion follows in Section 8.

## 2 Motivation

We develop an example scenario for an over running status. In that scenario, Mr. Kim carries a cell-phone, a smart watch, a MP3 player, and a PDA. Each device has its sensors and their corresponding context providers as shown in Fig 1. The example scenario is as follows.

Mr. Kim is exercising on the running machine in a fitness center. While running continuously, Mr Kim's pulse and blood pressure may exceed his normal status and he may be wet with his sweet during his exercising. If his physical condition excesses beyond normal values, he is on the over running status which has to be taken care of. Therefore, his PDA alerts to Mr. Kim about adjusting the level of exercise, and shows the current physical condition information about him on PDA. To support this example scenario, there is an aggregation rule and ECA policy rule. Aggregation rule and ECA policy rule included in the context-aware exercise assistant application are represented as shown in Table 1 and 2, respectively. A mobile device can include several condition rules for an aggregation task. In this case, the centralized context aggregation may delay the context aware service not provided on a right time and

cause the concentration of computational overhead on a coordinator device. It is inefficient to conduct the execution of aggregation processing on every context change only in a coordinator device.

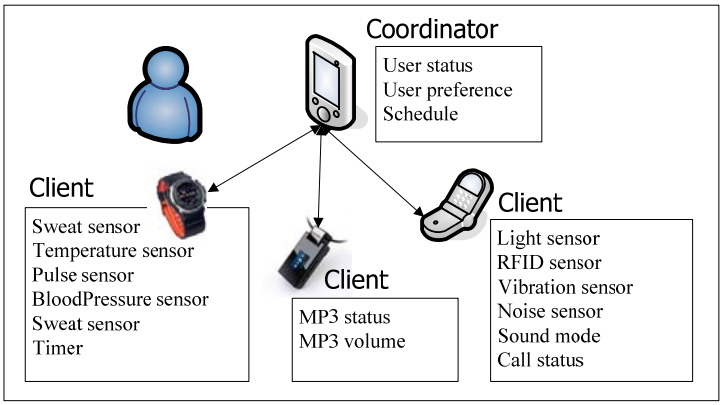


**Fig. 1.** An example of context information in a personal smart space

**Table 1.** The aggregation rule for example scenario

| **Condition rules:** | |
|---|---|
| a) Vibration(Kim, Running) ^ | Provided by cell-phone |
| b) Pulse(Kim, Over 140) ^ | Provided by smart watch |
| c) BloodPressure(Kim, Over 160) ^ | Provided by smart watch |
| d) Sweat(Kim, Wet) ^ | Provided by smart watch |
| e) -> Status(Kim, OverRunning) | On PDA |

**Table 2.** The ECA rule of context-aware exercise assistant application

```
On( Status(Kim, OverRunning))
    If(true)
    Do( start(service set2) )
      {    Alert to Kim for adjusting amount of exercise,
           Show the health information on PDA      };
```

## 3 Requirements for context aggregation in personal smart space

To provide the function of context aggregation in resource limited personal smart space, we introduce two main requirements of context management.

First, in personal smart space, constrained resource on each mobile device means that load balancing is an important issue for the context aggregation. Centralized context aggregation can cause the failure of coordinator device by exhausting the battery resource and the delay of context aware service by over-loaded aggregation processing on a coordinator. Therefore, to perform context aggregation effectively on

resource limited environment, the aggregation processing requires small processing overhead, and the aggregation task has to be distributed. To deal with these issues, we use simple inference mechanism instead of heavyweight ontology based inference engine, and we propose decomposition scheme for the distribution of aggregation task.

Second, the reliability of the context aware system is also an important issue. There are two kinds of aspects for reliability. One aspect is the reliability of a personal smart space. To deal with this aspect, Mobile Gaia [2] proposes election algorithm to select suitable coordinator when previous coordinator dies or disappears. Another aspect is the reliability of context event subscription. A personal smart space is configured with several devices including a coordinator device in an ad-hoc manner. Ad-hoc connectivity among devices can cause the change of network environment. In that case, it is not impossible to define all possible context event subscription for sub task rule according to network change. Therefore, characteristics of dynamic network change require flexible and adaptive context event subscription based on dynamic operating conditions varying over time and space.

## 4 Context Management Architecture for Personal Smart Space

To achieve the decomposition of context aggregation tasks, we need to consider three issues. First, we have to consider how to decompose an aggregation task into several sub tasks. As mentioned in Section 3, a large number of network transmissions can cause more processing overhead. Therefore, we decompose an aggregation task into several sub tasks based on the locality of context provider. If a client device processes a sub task locally, it is possible to reduce the number of network transmission. In personal smart space, locality is the most important consideration of any other factors. Second, it can be possible that a device has two context providers of same type. In that case, decomposition mechanism must select a suitable provider. Except for locality of context provider, there are other considering factors to select a suitable provider like frequency, accuracy, and granularity of context. Third, we also deal with the reusability of current sub tasks in personal smart space. It is wasteful that a coordinator delivers existing sub task to the same device every decomposition time. This fact makes our mechanism require the reusability of current sub task.

### 4.1 System Architecture

Fundamental functionalities of the context management are gathering, reasoning, and delivery of context information. We define five components as follows. **Context Widget** abstracts the raw sensor data and provides abstract context information. **Context Aggregator** provides high-level context information from low-level contexts according to aggregation rules. In our architecture, to provide small processing overhead, we use composite event detection mechanism [11] as an inference mechanism instead of logic based inference engine. **Context Interpreter** keeps track of the context in which the user is interested and notifies to application when one of contexts is set to true. **Context Aware Application** implements context sensitive application policy, which is ECA policy performing action according to context event

change. **Context Manager** has the role of context repository and includes minimal context ontology.

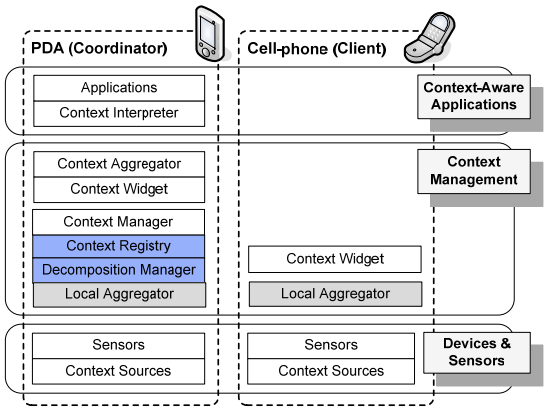


**Fig. 2.** Context management architecture for a personal smart space

In addition to these components, our efficient aggregation mechanism requires some extra components to provide the task decomposition scheme. Details of additional components are as follows. **Context Registry** manages the list of context providers. All context providers register themselves to context registry. Moreover, it supports to lookup the provider with the combination of context name and type. Moreover, it has the context properties for all existing context providers. **Decomposition Manager** receives decomposition requests and applies decomposition algorithm to generate context event subscription tree and sub task rule tree. After configuring two kinds of trees, decomposition manager adds sub task rules to the local aggregator and adds composite event rule to context aggregator. **Local Aggregator** detects that a certain sub task rule is satisfied and generates composite event. Then it notifies composite event change generated from sub task rules to context aggregator. And all devices in personal smart space have a local aggregator. Fig 2 shows an overall architecture of the context management in personal smart space depending on the role of devices.

### 4.2 Decomposition Algorithm

Decomposition algorithm uses aggregator name as an input parameter. In a personal smart space, several context aggregators can function as the status aggregator. A context aggregator includes one or more context aggregation rules. When a context provider appears or a context aggregator requests the decomposition of context aggregator, the decomposition manager gets aggregation rules from a context aggregator, and gets the device list currently available in a personal smart space from the context registry. This algorithm generates two kinds of trees: the one represents context event subscription and the other composite event rules and sub task rules. For a context aggregation rule, decomposition algorithm generates a sub task rule tree

with the result rule of the duplicated random value. Each result rule of a sub task is represented as RDF triple, SubTaskRule(ip, random value). And after finishing above sequences, sub task rules are inserted into sub task rule table, and a subscription tree is generated for a context aggregator.

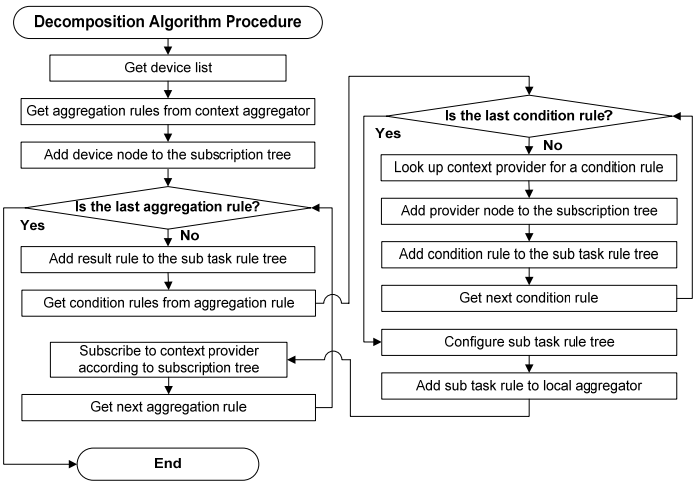


**Fig. 3.** Flowchart of task decomposition algorithm

Fig 3 shows the algorithm for task decomposition. In this algorithm, there are some mechanisms to select context provider of same type and to increase the reusability of existing sub task rules. We explain later in detail about these issues.

### 4.2.1 Selection of suitable context provider

If there are several context provider of same type on the same device, we need to select more suitable provider among them. The function of *selectProvider (possibleProviderList)* provides the mechanism for selection of suitable provider. When a context provider list is registered to Context Registry, some considering factors like frequency of context change, accuracy of context, and granularity of context are also registered as the form of ContextProperties class. Then we measure the utility value from the result of utility function. Fig 4 shows the utility function for a suitable context provider.

**Utility function**

Utility value = $(fc*w_f)+(ac*w_a)+(gc*w_g)$

fc: frequency of context change
ac: accuracy of context
gc: granularity of context

**Fig. 4.** Utility function for selection of context provider

Each factor has its weight value according to developer's policy. This algorithm calculates utility values for every possible provider, and selects the appropriate provider which has the highest utility value.

### 4.2.2 Reusability of existing sub task rules

Reuse of existing sub task rules makes it possible to avoid the delivery of a new sub task rule. Before adjusting the function of *addSubTaskRule()*, Decomposition Manager checks that a certain sub task rule exists or not in personal smart space. Decomposition manager on coordinator device has the list of sub task rules provided in personal smart space. Just before delivering a sub task rule to other device, this algorithm check the list of sub task rules. Then if there is a certain sub task rule in personal smart space, it does not deliver that sub task rule. It only subscribe to context provider which already has reusable sub task rule.

## 5 Implementation

We implement the proposed architecture as part of our ubiquitous computing middleware, called Active Surroundings [10]. Context management components in Active Surroundings run on IBM J9 (J2ME VM). In this section, we show the interaction among components in our proposed architecture.

Fig 5 shows the interaction among components to decompose an aggregation task into several sub tasks. Interaction among components is divided into two phases: registration phase and decomposition phase. In registration phase, all context widgets, aggregators, and local aggregators register itself to context registry. In decomposition phase, decomposition manager conducts a decomposition processing when a context aggregator or context registry requests decomposition. The overall interaction procedure among components works as follows.

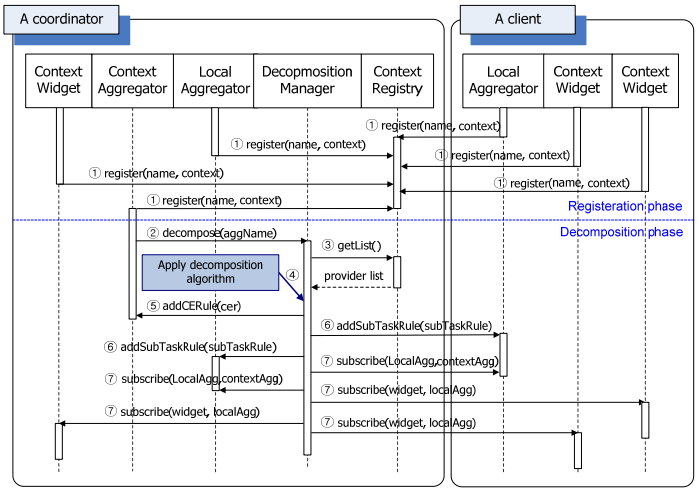


**Fig. 5.** Interaction among components for task decomposition

When other context aggregator appears, it requests decomposition to decomposition manager, and repeat the above procedure of the task decomposition.

## 6 Performance Analysis

In this section, we show how decomposition scheme reduces processing overhead for context aggregation. We expect that computational overhead for context aggregation is distributed in personal smart space. In the experiments, we measure the aggregation processing time on both coordinator and client devices and compare them with total processing time taken in the previous approach.

The over-running scenario in Section 2 is used for the experiments. We generate 100 context event changes randomly. Then we compare two cases: processing time in the previous and our approach. Total processing time for two cases is derived from the sum of processing time on coordinator, processing time on clients, and network transmission time for delivery context change events as shown in Equation (1).

$$PT(total) = PT(coor) + PT(cli) + NTT \quad (1)$$

PT(total) represents total processing time, PT(coor) represents processing time on coordinator device, PT(cli) represents processing time on clients, and NTT represents network transmission time, respectively. We use two PDAs, HP rx3715(Processor speed: 400MHz, Installed RAM: 152MB), and their operating system is Microsoft Windows Mobile Pocket PC 2003. We use IBM J9 as the VM to run our systems.

Fig 6 shows the result of the experiments considering four metrics: PT(total), NTT, PT(coor), and PT(cli).

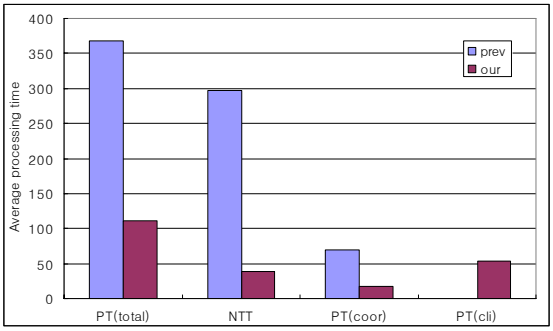


**Fig. 6.** The processing time and network transmission time on coordinator and clients

As Fig 6 shows, NTT takes the most part of PT(total). It means that network transmission affects the processing overhead for context aggregation significantly. Additionally, in case of our scheme, processing overhead only on client device is larger than that of previous work. However, processing overhead on client devices is slight to be ignored, comparing with the network transmission overhead.

The previous approach requires more PT(total) because it notifies context event to other device for every context change. On the contrary, our approach notifies the context change event only when a sub task rule is satisfied. In this way, the proposed scheme reduces the processing time for context aggregation. By reducing the aggregation processing overhead, it is possible to provide context aware services more efficiently and minimize the resource usage in personal smart space.

## 7 Related works

Previous research projects present infrastructure based context management architecture in typical smart space. Ontology based context aware middleware approaches like SOCAM, Context aware middleware in Gaia, and Cobra provide context information in resource plentiful environment [3], [4]. However, as a new concept of personal smart space appears, context aware systems need to consider limited resources on each device.

To provide context information in resource constrained mobile devices, Contory presents a context factory middleware for context provisioning on smart phone [5]. It supports three kinds of context provisioning methods including distributed context in ad hoc networks. The flexibility on switching one method to another at run time allows optimizing the utilization of computing and communication resources. However, unfortunately, it does not consider the processing overhead concentrating on a coordinator device for context aggregation.

Some researches motivating our works consider efficient aggregation processing by distributing a context aggregation. [6], [7], [8]. Event driven context interpretation presents the event driven distributed context aggregation model of context aware system [6]. In this work, distributed processing is easily supported through the use of several context providers helping an aggregation task. Moreover, as another approach for distributed aggregation, context fusion network presents graph based context aware middleware [7], [8]. Graph based abstraction make it easy to collect, aggregate, and disseminate context information. This approach increases the reusability of existing operators like context aggregator in context aware middleware. Although these works provide better processing time and increase the reusability of existing context providers, limitations of these works is that they conduct the aggregation processing in a centralized manner on a coordinator device.

## 8 Conclusion

We propose a resource efficient context aggregation scheme in personal smart space. We present the context management architecture to distribute aggregation tasks without predefined sub task rule in personal smart space. With this approach, we achieve reducing the processing overhead on a coordinator device by distributing an aggregation task into several sub tasks.

In this approach, we propose a lightweight context aggregation mechanism using composite event detection. Although it can reduce the aggregation processing overhead, the lightweight aggregator limits to support semantic context reasoning. We have a plan to consider providing semantic context information.

Moreover, as an extension of this work, we plan to consider multi-user environment where all devices are connected in an ad-hoc manner without coordinator device. Currently, we only support personal smart space considering single user environment and including a coordinator. With multi-user environment, we are also investigating some more complex scenarios and planning to present context management architecture in ad-hoc environment.

**Acknowledgments.** This research was partially supported by the Ubiquitous Computing and Network (UCN) Project, the MIC(Ministry of Information and Communication) 21st Century Frontier R&D Program and the KT-ICU Joint Research Center in Korea.

## References

1. Alexandros Karypidis and Spyros Lalis, "Automated context aggregation and file annotation for PAN-based computing",Personal and Ubiquitous Computing(PUC 2006), Oct. 2006
2. Shiva Chetan, Jalal Al-Muthadi, Roy Campbell, and M. Dennis Mickunas, “Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing”, In IEEE Consumer Communications & Networking Conference (CCNC 2005), Jan. 2005.
3. O. Riva, Contory: A Middleware for the Provisioning of Context Information on Smart Phones, in the Proceedings of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06)
4. T. Gu, H. K. Pung, D. Q. Zhang. "A Service-Oriented Middleware for Building Context-Aware Services." Journal of Network and Computer Applications (JNCA), Vol. 28, Issue 1, pp. 1-18, January 2005.
5. Anand Ranganathan, Roy H. Campbell. "An Infrastructure for Context-Awareness based on First Order Logic."Personal and Ubiquitous Computing 7 2003
6. Joo Geok Tan, Daqing Zhang, Xiaohang Wang, and Heng Seng Cheng, "Enhancing Semantic Spaces with Event-Driven Context Interpretation", PERVASIVE 2005
7. Guanling Chen and David Kotz, “Context Aggregation and Dissemination in Ubiquitous Computing Systems”, Dartmouth Computer Science Technical Report TR2002-420
8. Guanling Chen, Ming Li, David Kotz, "Design and Implementation of a Large-Scale Context Fusion Network", Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)
9. Anind K. Dey, Daniel Salber and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, 2001
10. Dongman Lee, Seunghyun Han, Insuk Park, SaeHoon Kang, Kyungmin Lee, Soon J. Hyun, Young-Hee Lee, and Geehyuk Lee, “A Group-Aware Middleware for Ubiquitous Computing Environments”, ICAT 2004
11. Peter R. Pietzuch, Brian Shand, and Jean Bacon "Composite Event Detection as a Generic Middleware Extension". IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks, 2004.
12. Harry Chen, Tim Finin, and Anupam Joshi, "An Ontology for Context-Aware Pervasive Computing Environments", The Knowledge Engineering Review 2003