Resource Management Based on Personal Service Aggregations in Smart Spaces

Peifeng Xiang, YuanChun Shi
Key Laboratory of Pervasive Computing, Ministry of Education
Department of Computer Science and Technology
Tsinghua University, Beijing 100084
xpf97@mails.tsinghua.edu.cn, shiyc@tsinghua.edu.cn

Abstract

This paper introduces a novel resource management approach based on personal service aggregations(PSAs) for smart spaces. As a smart space is usually a sharable system that simultaneously provides services for multi-user but with limited computational resources, the resource collision is inevitable. We propose the concept of PSA that maintains tasks, services, and underlying resources for each user to address the resource collision. By the help of PSAs, the resource management system can analyze what's the impact of a service for the user, and determines how to schedule the underlying resources. The model of a PSA and resource management strategies based on it are introduced, and a simulative example is given to further explain the idea. In the end, the future work is discussed.

1. Introduction

A smart space, for example, a smart meeting room, usually provides services for multi-user at the same time. As the computational resources in the smart space are always limited, the usage style of multi-user will inevitably lead to resource collision. For example, in our Smart Classroom[12], when the teacher is giving a presentation and some students are downloading files both through a wireless AP(Access Point), the bandwidth is not enough. When the system determines how to assign the limited bandwidth, it has to consider who are using the bandwidth, how important the bandwidth is for their tasks, and what are the roles that they are respectively playing in the Smart Classroom.

Our idea of resource management for multi-user is to construct a personal service aggregation(PSA) that maintains tasks, services, user's role, and underlying resources for each user. When the resource collision happens, the system will analyze the importance of related services and determine how to schedule the underlying computational resources by the help of PSAs.

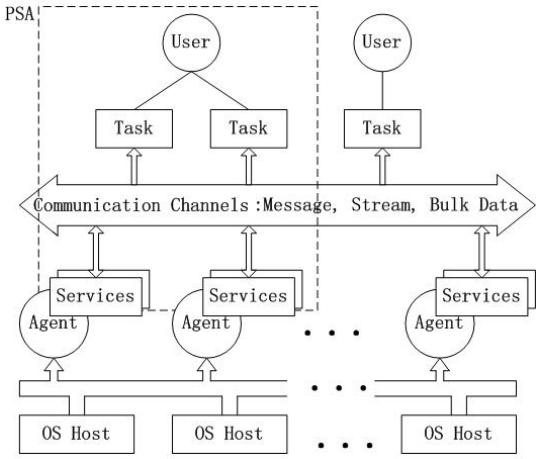

Figure 1. Architecture of smart platform

The resource management is based on a multi-agent system called Smart Platform[8] which is a distributed computing platform for smart spaces. For clarity, as depicted in Figure 1, we explain important terms that are used throughout this paper:

OS host: OS hosts are traditional OS systems that manage low level resources such as memory, CPU, bandwidth and some extended runtime supports for smart spaces, e.g., the nomadic file access.

Agent: Agents are distinct software modules running on OS hosts with its own execution process. Agents interact with users and other agents through interfaces called services.

Proceedings of the Third IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS'05)
0-7695-2357-9/05 $20.00 © 2005 **IEEE**

Service: Services are well defined functionalities provided by agents. One agent can provide one or more services. For example, the record agent in the Smart Classroom provides video capture service, playback service, et al.

Task: A task is a collection of services that are combined to achieve a user's goal. For example, giving a presentation is a task which may comprise the following services: file access, PPT display, location, dynamic audio capture, etc.

PSA: The PSA is a data structure that maintains user's tasks, services, user's role, and underlying resources. There is one PSA for each user.

The purpose of resource management is to schedule agents and OS hosts to support users' tasks by analyzing services in PSAs.

Although the resource management is based on the Smart Platform, we believe the method can be applied to other service-oriented software infrastructure for smart spaces where the tasks or applications are composed by distinct services[10][7][9].

Existing research projects mainly address the resource management on the OS host level [1][5]. Gaia[3] provides support from OS host to application compositions called active-space level functionality[4], but does not address the problem of multi-user explicitly. Rascal[2] focuses on the resource mapping and arbitration between services and agents.

The following sections are organized as follows: section 2 introduces the PSA model; section 3 describes management strategies based on PSAs; section 4 presents a simulative example to further explain the idea; the conclusion and future work are given in the end.

2. PSA Model

We borrow the idea of ConcurTaskTrees[11] and extend it to construct the PSA model. The graphical tree-like representation with relationship operators makes the PSA model easy to build and analyze.

The nodes used in the PSA tree model are defined as follows:

- ***U***. ***U*** describes a user's information including user's ***name*** and the privilege p_U . Different p_U s are assigned to users according to their roles in the smart space. For example, the teacher has a higher p_U than a student, and all of the students have the same p_U .
- ***FS***, ***IS***. We distinguish two kinds of services: ***FS*** is the final service whose output is not the input of other services. Usually, the output of a ***FS*** is directly for users. ***IS*** is the internal service whose output is the input of other services. For example, the PPT display service on the screen is a final service while the file access service is an internal service that can provides files for the PPT display service. For both ***FS*** and ***IS***, two sets are defined: ***quality set*** that describes the possible choice of quality for a service including the speed of reaction, throughout, or precision, etc.; ***provider set*** that lists agents that are capable to provide this service. Each FS has a privilege p_S .
- ***FS***, ***IS***. Describes a running ***FS*** or ***IS***. Two additional values are assigned: ***quality*** for current service quality and ***provider*** for current running agent that provides the service.
- ***T***. ***T*** identifies a task with the task name.

For clarity, we introduce the relationships between nodes by building an example PSA step by step. As in Figure 2, a PSA is built in four phrases:

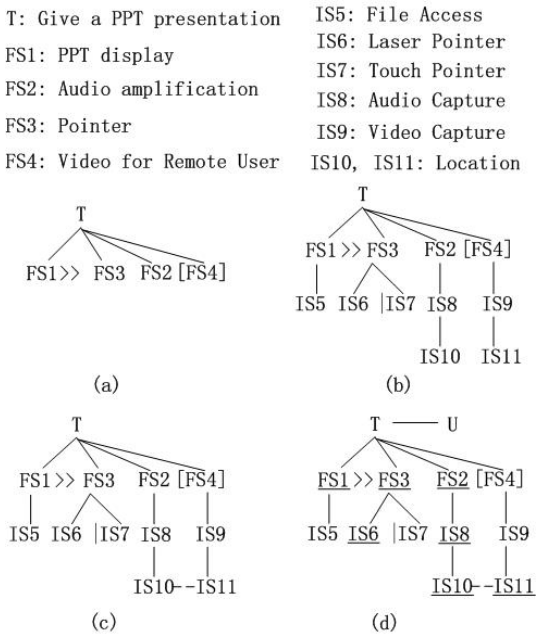

Figure 2. Build a PSA

- Step 1(Figure 2a): building ***FS*** for ***T***. ***FS***'s parent must be ***T***. The line from a ***T*** to ***FS*** means the user can trigger the ***FS*** in this ***T***. Each ***FS*** has a p_S to describe ***FS*** 's importance in this ***T***. For example, ***FS1***. p_S > ***FS2***. p_S .

- Step 2(Figure 2b): constructing ***IS***. The output of a child service is the input of a parent service.
- Step 3(Figure 2c): merging services. The output of a service may be the input for several services at the same time. For example, the location service(***IS10***, ***IS11***) provides the same input(a user's current location) for ***IS8*** and ***IS9***, they are merged into one by the dashed-line.
- Step 4(Figure 2d). Representing a runtime PSA. Active services and the user are identified.

The operators describing dependency relationships between services are only applied to those on the same tree level. Currently, only three operators are defined(***S*** can be ***FS*** or ***IS***):

- ***S1***|***S2***: ***S1*** or ***S2***. For example, in Figure 2b, when the presenter is using the pointer service, he can choose a laser pointer or a touch screen pointer.
- ***S1***>>***S2***: Enable. Although S1 is not a child service of S2, S2 only makes sense when S1 is active. For example, in Figure 2a, the pointer service(that is used to point the content of PPT) only makes sense when the PPT is displayed on the screen.
- [***S***]: ***S*** is optional.

A complex task may comprise several child tasks. Figure 3 illustrates a complex lecture task that includes the PPT presentation task in Figure 2. The relationship operators between tasks follow the definition in [11].

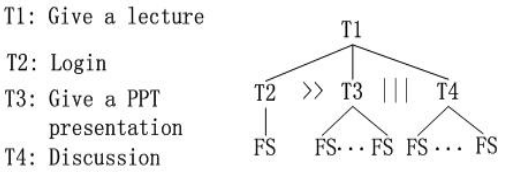

Figure 3 a Complex Lecture Task

3. Resource Management Based on PSAs

When a resource collision is triggered by two or more services, the following management steps are followed:

- Step 1: For ***U*** with different p_U, try to keep ***FS***'s ***quality*** for the ***U*** which has the higher p_U.
- Step 2: For ***U*** with the same p_U, try to lower ***FS***'s ***quality*** to solve the collision.
- Step 3: If some ***FS*** have to be paused, the less ***U*** that have no active FS, the better.
- Step 4: In one ***PSA***, the more active ***FS***, the better.
- Step 5: In one ***PSA***, if some ***FS*** have be paused, the ***FS*** with a higher p_S is preferred to reserve.

The automatic resource management sometimes makes the system behavior seems strange to users. However, with the help of PSAs, the system can inform users how their tasks are affected by other users. For example, the system can inform a user: "Teacher Li is downloading files, your web browsing will slow down" or explicitly ask users how to schedule resources: "Bob want to use the laser pointer service(based on computer vision), would you like to pause your voice recognition service?(The two services cause a collision on CPU resources)".

We argue that this kind of interactions between the system and users are important. As the smart space is becoming more and more complex, the relationships between services and the underlying resources is difficult to understand for users. For example, in a smart space with the nomadic file access service, when a user hopes to speed up his file downloading, he probably couldn't know what tasks and services occupies the bandwidth by only asking other users. ***PSA***s can provide help in such situations.

4. An Example

A complete implementation of the resource management also depends on a manager to monitor and dispatch agents and OS resources. Currently, this manager is still under work. Here we present a simulative example(Figure 4).

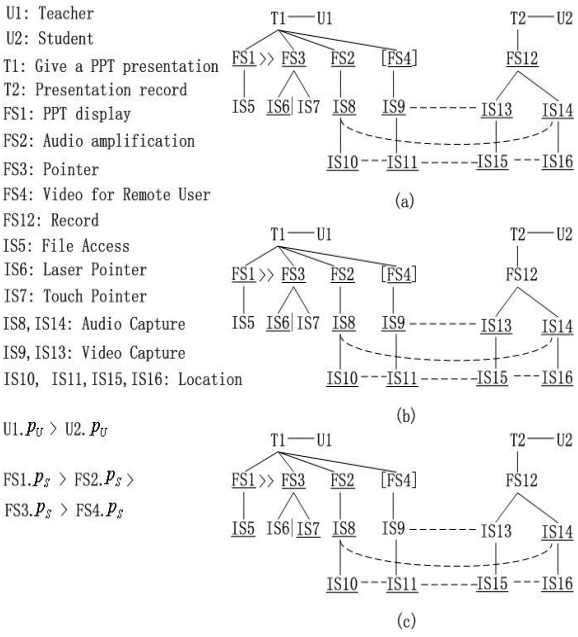

Figure 4 An Example

From Figure 4a to Figure 4b, when the collision of the network bandwidth is detected, the record service for video is paused.

From Figure 4b to Figure 4c, when the collision of the CPU resource is detected(a big video is displayed in PPT), remote video is paused and the system informs the teacher to use touch pointer instead of laser pointer which consumes more CPU resource.

5. Conclusion and Future Work

In this paper, we introduce a high level resource management approach which is based on the multi-user feature in smart spaces. For each user, a ***PSA*** that maintains user's tasks, services, user's role, and underlying resources is built to support the resource management.

Future work on ***PSA***s will include extending operators between services so as to describe more complex tasks and providing a more efficiency algorithm for resource management. In addition, we are developing a low level resource manager so that the complete resource management can be deployed in the Smart Classroom.

6. Acknowledgements

This research is supported by NCET(Program for New Century Excellent Talents in University, China) and NSFC(Natural Science Foundation, China)

7. References

[1] X.H. Gu, Messer A., Greenberg I., Milojicic D., et al. Adaptive Offloading for Pervasive Computing. IEEE Pervasive Computing, vol. 3, no. 3, 2004, pp. 66-73.

[2] Krzysztof Gajos. Rascal – a Resource Manager for Multi Agent Systems in Smart Spaces. CEEMAS01, Krakow, Poland, 2001.

[3] Manuel Román, Christopher K. Hess, Renato Cerqueira, et al. Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing, vol. 1, no. 4, 2002, pp. 74-83.

[4] Manuel Roman, Brian Ziebart, and Roy Campbell. Dynamic Application Composition: Customizing the Behavior of an Active Space. PerCom03, Dallas-Fort Worth, Texas, 2003.

[5] Duran-Limon H.A., Blair G.S., Coulson G. Adaptive Resource Management in Middleware: A Survey. IEEE Distributed Systems Online, vol. 5, no. 7, 2004.

[6] Garlan D., Siewiorek D.P., Smailagic A., et al. Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, vol. 1, no, 2, 2002, pp. 22-31.

[7] Cheyer A, Martin D. The Open Agent Architecture. Journal of Autonomous Agents and Multi-Agent System. vol. 4, no. 1, 2001, pp. 143-148.

[8] Xie W K, Shi Y C, Xu G Y, et al. Smart Platform - A Software Infrastructure for Smart Space (SISS). The Fourth International Conference on Multimodal.
Interfaces(ICMI2002), Pittsburgh, USA, 2002.

[9] Coen M, Phillips B, Warshawsky N, et al. Meeting the Computational Needs of Intelligent Environments: The Metaglue System. In: Proc. MANSE, Dublin, Ireland, 1999.

[10] Choonhwa Lee, Nordstedt D., Helal S. Enabling Smart Spaces with OSGi. IEEE Pervasive Computing, vol. 2, no. 3, 2003, pp. 89-94.

[11] F. Paterno, C. Mancini, S. Meniconi. Concur- TaskTrees: A Diagrammatic Notation for Specifying Task Models. Proc. Interact, Sydney, 1997.

[12] Yuanchun Shi; Weikai Xie; Guangyou Xu, et al. The Smart Classroom: Merging Technologies for Seamless Tele-Education. IEEE Pervasive Computing, vol. 2, no. 2, 2003, pp. 47-55.