A Multi-agent Software Platform Accommodating Location-Awareness for Smart Space

Hongliang Gu, Yuanchun Shi, Guangyou Xu, and Yu Chen

Computer Science Department, Tsinghua University, Beijing 100084, P.R. China
ghl02@mails.tsinghua.edu.cn, shiyc@tsinghua.edu.cn,
xgy-dcs@mail.tsinghua.edu.cn, yuchen@tsinghua.edu.cn

Abstract. Software Platform is a middleware component of Smart Space to coordinate and manage all modules. Location-awareness is a common feature of many modules. Current several typical methods for distributed systems can hardly be competent for both the role of Software Platform and accommodating location-awareness simultaneously. Aiming at this, we present our method: SLAP (Smart Location-awareness-Accommodating Platform). The method, on the basis of OAA (Open Agent Architecture), adopts such new technologies as *Poll-Ack mechanism*, *dual-central coupling model* and *hybrid architecture*. Consequently it not only reserves the advantages of OAA to coordinate multi-modal modules efficiently and flexibly, but also accommodates location-aware computing well.

1 Introduction

Smart Space [1] (or Intelligent Environment) is a working environment integrated with numerous distributed software and hardware, including multi-modal modules and positioning sensors, which is also a system intensively applying pervasive/ubiquitous computing technologies. The Software Platform (also called Software Infrastructure), working as a middleware between OS (Operation System) and application modules, is a fundamental component of Smart Space to coordinate and managing all hardware and software modules.

Nowadays location-awareness is becoming an indispensable characteristic of most modules in Smart Space, which brings about a research field: location-aware computing. In our project, Smart Classroom [2] (a Smart Space on tele-education), location-awareness means that applications or services can modify their own behaviors unobtrusively or non-intrusively to adapt to users' purpose, according to the location (or spatial relationship) of located-objects [3] (including applications or service).

Both accommodating location-aware computing and adapting to Smart Space give the Software Platform dual challenges, which are just all the necessity of Smart Classroom. However, current several representative methods for tradition distributed systems, e.g. DCOM, CORBA, Metaglue and OAA (Open Agent Architecture) etc, can not give both needs a satisfying solution simultaneously. Aiming at this, we present our method: SLAP, a system with our improvement on OAA, which it not only efficiently coordinates and manages all modules according to the demands of Smart Space, but also accommodates location-aware computing very well.

© IFIP International Federation for Information Processing 2005

The contents below are as follows: Section 2 discusses the demands of Smart Space on Software Platform. Section 3 introduces the requirements of location-awareness and the deficiencies of OAA. Section 4 presents our improvement's key technologies of. Section 5 presents the architecture and primitives of SLAP. Section 6 elaborates on the experiments. And section 7 concludes this paper.

2 Criterion and Selection of Software Platform

2.1 Demands of Smart Space on Software Platform

As far as the fact that Smart Space consists of various computing and communication units is concerned, Smart Space is a distributed system in some sense. However, it has some special features different from the normal distributed systems:

1. autonomy and independency
The modules in Smart Space are more autonomous and independent than those in distributed systems. For example, most modules in Smart Space can run or expire independently, which are not in a certain module's control and do not comply with other modules' assignment at all.
2. loose-coupling
A Smart Space system is very dynamic. Modules are restarted or moved to different hosts and System configurations change time to time. The loose coupling of modules will help to cope with this nature of Smart Space, as well as to resile from failure.
3. lightweight
As an underlying component, the Software Platform is to run on the various units in Smart Space which have various abilities of computing and communication spanning from mainframe computers to embedded systems, and to be used by the various module's developers who have uneven IT backgrounds. Thus, the feature of lightweight helps the Software Platform to accommodate various units, and to give various users a facile and simple interface. The light-weighted Software Platform only provides some key services and commits other complex functions to the applications in manner of the end-to-end implementation.

The items above are almost the common demands for all modules of Smart Space, especially for the multi-modal modules.

2.2 Selection of Software Platform

The Smart Space's features mentioned above are the criterion to select proper framework model for Software Platform. Currently, the representative methods for distributed systems can roughly be divided into two categories: Distributed Component Model (DCM), and Multi-Agent System (MAS).

In essence, DCM model is to encapsulate modules into objects (though someone argue that component is slightly different from object), which abstractly represents the states and behaviors' implementation (also called properties and methods) of modules. The representative DCM models include DCOM, CORBA and EJB etc. In DCM model, there must be a centralized thread of application logic which decides which objects to be invoked (used) and when to invoke (use). However, this premise is difficult to be met in Smart Space, due to the modules' autonomy and independency. For example, in Smart Classroom, a laser-pen-tracking module continuously tracks the position of laser point, which is a projecting point on Smart Board (a large-sized touch screen) corresponding to users' gesture, while a speech-recognition module keeps recognizing the user's voice. In the example, a clear centralized control logic is difficult to be picked up. Instead, there are two parallel application logics simultaneously.

In contrast, MAS model encapsulates each module into an agent, which not only has the same behaviors' implementation as an object, but also owns itself activation logic, executing process and purpose. That is, according to its environment, an agent can itself decide what to do and how to do, which an object can hardly achieve. Thus, in MAS model, the control logic of modules is decentralized, which is more flexible and fitter for Smart Space than that of DCM model. The typical MAS models include Metaglue [4], Hyperglue [5] and OAA [6],[7].

Besides those advantages, MAS model is usually more light-weighted than DCM model. Those representative DCM models, such as DCOM and CORBA, all own many complicated features, e.g. object set, transaction process and currency control etc. In view of those synthetic factors, we select MAS system instead of DCM model as the abstraction model of Software Platform. According to modules' coordination mode, MAS model is divided into two kinds: direct-coupled and meeting-oriented.

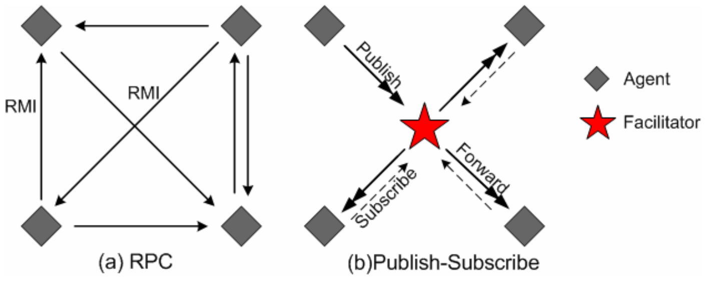

Fig. 1. The inter-module communication mode

The direct-coupled mode is also called RPC-like (Remote Procedure Call) mode. In this mode, each module must know other modules' definite reference (name or ID). As Fig. 1(a) shows, the module on the one side must know who the other side is, and furthermore the modules on the two sides must run at the same time. Both Metaglue and Hyperglue belong to this mode, in which the inter-agent communication is achieved by Java RMI. Undoubtedly, the direct-coupled mode is tight-coupling.

The meeting-oriented mode means the modules achieve the mutual coordination by broadcasting messages in a logic (virtual) meeting room. This mode's feature is that the modules needn't own others' references. The Publish-Subscribe mechanism, which OAA adopts, is typically meeting-oriented. In OAA, when an agent wants a certain kind of message, it will register the messages on a message center: *Facilitator*. This activity is called *subscribe message*, which is also called asking a question in OAA. And if an agent tends to send messages, it needn't know which agent and how many agents need those messages. What it does is only to send Facilitator the messages tagged with the name or category, and then Facilitator forwards all messages to those agents who subscribe them, according to the messages' category and name. The agent's activity above is called *publish message*. The whole process is called "*delegated computing*" in term of OAA, which is skeletally shown in Fig.1 (b).

In comparison with the tight-coupling coordination mode of Metaglue, that of OAA is loose-coupling. Considering this factor, we prefer OAA to Metaglue and Hyperglue as a framework of Software Platform.

3 Deficiencies of OAA on Location-Awareness

3.1 Requirements of Location-Awareness on Software Platform

In Smart Space, the location-aware computing system consists of three parts: location-aware applications, location server and position system. The position system of our project is Cricket V2.0 [8], in which each positioning unit (a PDA with Cricket Mote) knows its own geometric coordinate location and then sends its location to the location server by a wireless network. The location server, on the one hand, takes charge of storing and managing all units' location; on the other hand, provides the location-related services for the applications. In Smart Classroom, the location server adopts an implementation method called ASMod [9] to provide two kinds of service: query service and spatial event service. The former asks the applications' spatial query, which is like a SQL service; the latter tracks the varying of located-objects' spatial relationship to emit the relevant event notification.

To support the location-aware computing system, Software Platform encounters two new issues: one is how to efficiently organize the communication of position system, namely the communication between the location server (also an agent) and positioning agents (which correspond to positioning units); another is how to organize the communication between location-aware applications according to their locations (or spatial relationship) which is also called *location-based communication*. Unfortunately, neither of the issues is OAA competent for.

3.2 Deficiencies of OAA on Supporting Location-Aware Computing

First, OAA does not excel at organizing the communication between the location server and positioning agents efficiently, due to its Publish-Subscribe mechanism. In the mechanism, when to publish messages and how many messages to publish only depend on the agent itself, which we call *free-publishing* characteristic. This characteristic adapts to such Smart Space's demands as modules' autonomy and independency and the system's loose coupling, meanwhile it also brings about two problems:

One problem is the difficulty in controlling the communication between the location server and positioning agents. In Smart Space, the location server usually needs to obtain the location from various positioning agents at various frequencies in different time according to its data's state, which is essentially the location server's data update policy. For example, in a time, if the location server infers that a positioning agent is moving quickly (maybe attached to a mobile person), it will get data from the agent twice per second. Likewise, in another time, if the location server infers the positioning agent seldom moves, it will get data from the agent only once per minute.

Another problem is that the disorderly contentions on the wireless network's channel increase, which results in the degradation of performance and throughput. Because each positioning agent publishes its data (namely location) only according to its own willing, despite the others and the location server's need, the disorderly contentions are inevitable, which will become more intensive with the increasing of the positioning agents' number and the frequency of publishing in each agent.

Secondly, OAA is also incompetent for organizing the location-based communication between applications. In Smart Space, much communication between agents is not constantly sustaining from beginning (subscribing) to end (unsubscribing), but varies according to their spatial relationship. The kind of communication, namely the location-based communication, is different from that of multi-modal modules, which OAA excels at. For example, when a PDA enters the service scope of Smart Board, the communication between the PDA agent and the Smart Board agent will emerge; and when the PDA leaves the service scope, the communication will also be broken off. Unfortunately, OAA is incompetent for the location-based communication. The cause is that neither Facilitator nor the source agents (which publish messages) cares the agents' location and changes their behaviors according to the varying of location.

4 Key Technologies of Our Improvement

Aiming at the deficiencies of OAA on supporting location-aware computing, we present our solution to Software Platform: SLAP (Smart Location-awareness-Accommodating Platform). Here we first introduce the Key technologies of SLAP, which are to solve the two issues brought by location-aware computing.

4.1 Poll-Ack Mechanism

Aiming at the incompetence of Publish-Subscribe mechanism for organizing position system communication, we present an appropriative inter-agent communication mechanism: Poll-Ack mechanism. This mechanism is described as follows:

As Fig. 2 illustrates, the communication consists of Poll-Ack cycles. And each cycle is initiated by a broadcast message from the location server, which is called *Poll*. A poll indicates which agent to publish its location. On receiving the Poll, the positioning agent indicated in the Poll, replies an acknowledgement message called *Ack* (including ID and location) in a fixed time. The location server stores all agents' location, and assigns poll number to each agent in the unit time according to the agent's velocity. An agent's velocity is its adjacent location difference divided by the interval

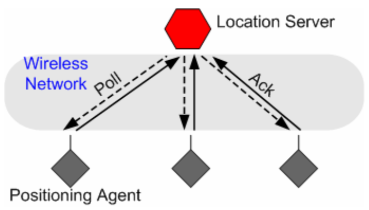

Fig. 2. The Poll-Ack mechanism

of its adjacent Ack. The higher velocity an agent is at, the more polls the location server assigns to it. Hence, not only this mechanism doesn't produce channel contention, but also it is a velocity-directed bandwidth assignment in some sense.

4.2 Dual-Central Coupling Model

Aiming at the incompetence of OAA for organizing location-based communication, we present the dual-central coupling structure and the Spatial-event-directed Publish-Subscribe mechanism.

In OAA there is a unique coupling center, Facilitator, to organize message communication. In contrast, in SLAP there are two coupling centers: LAMD (Location-Aware Message Dispatcher) and LocServ. The former provides the analogous function of Facilitator, and the latter plays the role of location server. LocServ has a component, *Spatial Event Generator*, which tracks the agent's moving and translates location into spatial events. LAMD owns a dispatching engine, *Forward-Valve*, which decides messages whether to forward indeed according to the event notification from LocServ. The structure of two coupling centers is shown in Fig. 3.

The dual-central coupling structure adopts a new communication mechanism called Spatial-event-directed Publish-Subscribe. The mechanism is based on Publish-Subscribe with some modification. The modification is as follows:

1. When an agent subscribes a kind of messages, it is demanded to submit a spatial condition of the messages to LocServ at the same time. The spatial condition indicates the premise the communication needs, and the premise is express as a spatial relationship, such as, the publisher' location must be contained in the subscriber's scope. The step is called *spatial condition's customization*.
2. LocServ keeps on obtaining the latest location of all agents from the position system (namely tracking agents' moving), and judges whether the spatial conditions are met by its Spatial Event Generator. When the spatial conditions become met or unmet, LocServ sends LAMD the event notifications: *message-forward-enable* or *message-forward-disable*.
3. According to the notifications, LAMD will decide whether to forward the subscribed messages to the subscriber (agents).

If an agent wants the received messages to be irrelevant to the location, it submits a command to LocServ to abolish the spatial condition of messages. The whole process of this mechanism is shown in Fig. 4.

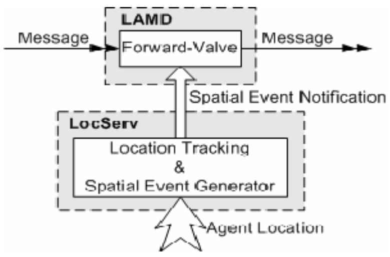

Fig. 3. The structure of two coupling centers

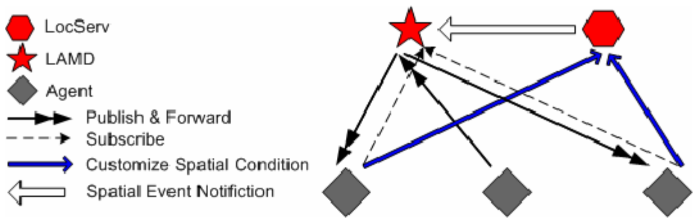

Fig. 4. The spatial-event-directed publish-subscribe mechanism

5 The Architecture of SLAP

As a Software Platform, SLAP is a middleware between OS (Operation System) and applications (agents). An overview of SLAP is shown in Fig. 5.

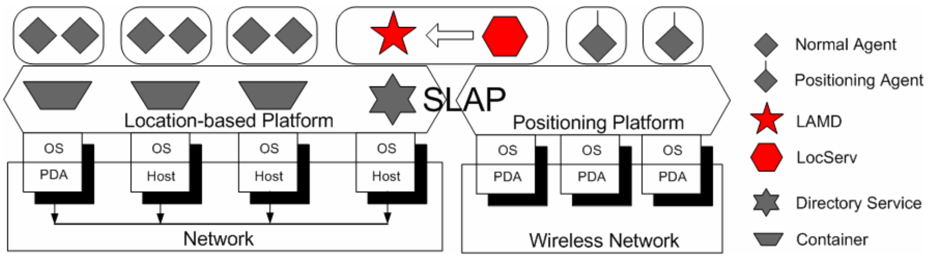

Fig. 5. The architecture of SLAP

SLAP is a *hybrid architecture* composed of two parts, which respectively correspond to two kinds of communication environment. The right part, *positioning platform*, is used for the position system to coordinate LocServ and the positioning agents, which adopts the Poll-Ack communication mechanism. And the left part, *location-based platform*, is to organize the location-based communication between agents, which adopts the dual-central coupling model. The host containing LAMD and LocServ spans two network environments: one connects to the position system' network (wireless network), another connects to the network all normal agents share. To enhance some functions of SLAP, we add in some components that OAA doesn't own. For example, the *containers*, acting as mediators under the agent layer, are to shield heterogeneous OS and accommodating different developing languages, such as C++ and Java. The *Directory Service* is used for the service's discovery.

6 Performance Analysis

To evaluate the performance of SLAP, we compare the Poll-Ack mechanism (which SLAP adopts) with the Publish-Subscribe mechanism (which OAA adopts) on the communication efficiency of position system. Define:

T_L = Average time for a positioning agent to calculate its location

D_L = Transmission duration of a location message (which is in the form of Ack message in the Poll-Ack mechanism)

D_p = Transmission duration of a poll message

Now we first investigate the performance of Publish-Subscribe mechanism. Providing a positioning agent publishes its location message at once after calculating its location, the probability that the positioning agent publishes the location:

$$p = {D_L}/{T_L} \tag{1}$$

For a successful publishing exactly one of n positioning agents should be publishing at a given time. Hence the probability that only one given positioning agent is publishing at a particular time:

$$P_1 = p(1-p)^{n-1} \tag{2}$$

When there are n positioning agents, the probability that exactly one positioning agent is publishing at a given time is the channel utilization of wireless network U .

$$U = np(1-p)^{n-1} \tag{3}$$

For maximum utilization of publish-subscribe mechanism, there exists:

$$\frac{dU}{dp} = n(1-p)^{n-1} - np(n-1)(1-p)^{n-2} = 0 \Rightarrow p = \frac{1}{n} \Rightarrow T_L = nD_L \tag{4}$$

Hence, in the case above, the optimum utilization of Publish-Subscribe mechanism:

$$Uo_{p-s} = (1-\frac{1}{n})^{n-1} \tag{5}$$

As for the Poll-Ack mechanism, the channel is occupied by Polls and Acks in turn. Hence, the channel utilization U' is:

$$U' = \frac{D_L}{D_L + D_P} \tag{6}$$

Because the equation (4) exists in the case of channel's maximum utilization, the optimum utilization of Poll-Ack mechanism Uo_{p-a} is:

$$Uo_{p-a} = \frac{T_L}{T_L + nD_p} \tag{7}$$

As for a given position system, the average time of calculating location T_L is fixed, which only depends on the hardware's intrinsic functionality. In contrast, the poll's transmission duration D_p is determined by the concrete Poll-Ack mechanism.

Both Publish-Subscribe mechanism and Poll-Ack mechanism are simulated using the ns-2 network simulator with suitable extensions [10], which is guided by the CMU wireless extensions. In the simulation experiment, T_L is set to 100ms, D_p is set to 1ms, 2ms and 4ms, which are corresponding to the curve Poll-Ack (1), (2) and (4) in Fig. 6 respectively. And the performance of Publish-Subscribe mechanism is labeled by the curve Pub-Sub in Fig. 6.

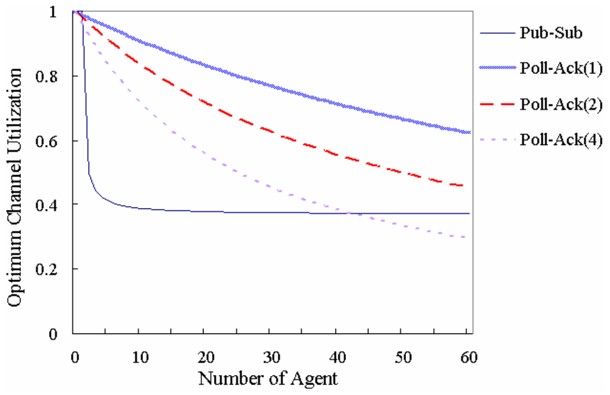

Fig. 6. The communication performance of SLAP versus OAA in position system

As Fig. 6 shows, in most cases, the optimum channel utilization of the Poll-Ack mechanism is superior to that of the Publish-Subscribe mechanism. The few exceptional cases occur on Poll-Ack (4) with the agent number of about 45, where the total transmission duration of polls (180ms) is greater than the average positioning time (100ms) by far. These cases are very extreme, which rarely appears in practice. Another trend seen from Fig. 6 is that, the smaller the poll's transmission duration is, the larger improvement of channel utilization the Poll-Ack mechanism achieves on the Publish-Subscribe mechanism.

7 Conclusion

On the one hand, as a Software Platform for Smart Space, being based on OAA, SLAP reserves the main characteristics of OAA, a loose-coupling multi-agent system. Those characteristics conform to Smart Space's demands on Software Platform better than other distributed system methods, which highly ensure modules' autonomy and independency, inter-module loose-coupling and system's lightweight. Hence, as far as accommodating Smart Space is concerned, SLAP, as well as OAA, is an excellent Software Platform, especially for coordinating most multi-modal modules.

On the other hand, SLAP overcomes the shortcomings of OAA on accommodating location-awareness. By introducing in the dual-central coupling model, SLAP realizes inter-agent location-based communication that OAA used to not be able to provide. And by introducing the Poll-Ack communication mechanism into position system, SLAP achieves higher channel utilization and more efficient communication performance than OAA. Thus SLAP not only is competent for Software Platform of Smart Space, but also accommodates location-aware computing well.

References

1. http://www.nist.gov/smartspace/
2. Y. C., Shi, et al.: The smart classroom: merging technologies for seamless tele-education. Pervasive Computing, IEEE press, Vol 2, No 2, 2003, pp. 47-55
3. B. Schilit, N. Adams, and R. Want: Context-aware computing applications. IEEE Workshop on Mobile Computing Systems and Applications, IEEE CS Press, 1995, pp. 85-90
4. M.H. Coen, B. Phillips, N. Warshawsky, et al.: Meeting the computational needs of intelligent environments: The Metaglue system. Proc 1st International Workshop Managing Interactions in Smart Environments (MANSE'99), 1999, pp.210-213
5. Peters S, Look G, Quigley K.: Hyperglue: Designing High-Level Agent Communication for Distributed Applications. Technical Report, Laboratory of CS and AI (CSAIL), Massachusetts Institute of Technology, 2002.
6. SRI., OAA web site: http://www.ai.sri.com/~oaa
7. Adam Cheyer, David Martin: The Open Agent Architecture. Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publisher, Vol 4, No 1-2, 2001, pp.143-148
8. Adam Smith, Hari Balakrishnan, Michel Goraczko, Nissanka Priyantha: Tracking Moving Devices with the Cricket Location System. Proc 2nd International conference on Mobile systems, applications, and services(MobiSys'04), 2004, pp.190-202
9. Hongliang Gu, et al.: A core model supporting location-aware computing in Smart Classroom, Proc 4th International Conference on Web-based Learning, 2005, pp.1-13
10. NS-2 network simulator. http://www.isi.edu/nsnam/ns/