Software Infrastructure for Sentient Materials

Tomoko Shoji, Nobuyoshi Nakamura, Kaori Fujinami, Tatsuo Nakajima
Department of Information and Computer Science
Waseda University
3-4-1 Okubo Shinjuku Tokyo 169-8555 JAPAN

Abstract

In the future, every material will contain a computer and become smart. We call these materials *sentient materials*. Sentient materials can communicate each other and change its behaviors according to their surrounding environments. However, current infrastructures such as the Internet are not suitable for building the infrastructure for supporting sentient materials. The paper especially describes two issues for building software infrastructures for supporting sentient materials.

1 Introduction

Ubiquitous computing environments create a seamless boundary between atoms and bits. Especially, in the near future, embedding various types of sensors in our daily lives enables an application to customize their behavior according to its user's situation. Also, physical objects in our dairy lives can be used to manipulate information in cyber spaces. However, the vision requires a new software infrastructure that collaborates various dairy objects in a spontaneous way.

One of the most important issues to build ubiquitous computing environments is how to offer context information for composing services. The composition of services according to a user's preference and situation needs various context information about our daily lives. The first step to retrieve the information requires various sensors. However, current sensor technologies are not suitable to retrieve our context information. We believe that future materials will contain many small computers and sensors, and they give various information to us[1, 2]. For example, they calculate geographic relationship among materials, or provide unique identifiers that are useful to know what they are. We call the intelligent materials *sentient materials*. The real sentient materials may not be available soon, but we like to consider the impact of the sentient materials to ubiquitous computing while building the seamless town applications. Also, it is important to create a large-scaled database to store information about our real world. The database stores abstract information extracted from many sentient materials in our world. We are considering to adopt a peer-to-peer technology such as Pastry[4] and Chord[5] to build the scalable context information database.

In this paper, we present two issues to support sentient materials. The first issue is a communication infrastructure for supporting collaboration among sentient materials. The purpose of the infrastructure is to collect information about all sentient materials, and provide these information to application programs. This enables us to remove the boundary between bits and atoms. The second issue is a system support for privacy information. We show how to protect our privacy information from sensors embedded in our environments.

2 Scalable Context Database

We have several problems to build a large-scale database in ubiquitous computing environments. One of the problems is the lookup problem for dynamic resource information. How do you find any given objects in a network? There are three approaches for the lookup in recent years.

2.1 Routing approach

One approach is the centralized routing. This approach stores a database in the centralized servers. But this approach has inherent scalability and resilience problems. The database is a central point of failure. Moreover, the centralized routing does not suit the routing of the dynamic database. Because the precision of search and the time is trade-off, when the information quantity increases, it becomes difficult to cover all information.

The other approach is the hierarchical routing. The Internet's Domain Name System (DNS) uses this lookup approach. Queries start from the top of the hierarchy and, by following forwarding references from peer to peer, traverse a single path down to the peer containing the expected data. The disadvantage of this approach is the failure or removal of the root or a peer in a high level

Proceedings of the IEEE Workshop on Software Technologies for Future Embedded Systems (WSTFES'03)
0-7695-1937-7/03 $17.00 © 2003 IEEE

hierarchy can be catastrophic, and the peers in higher level take load larger than the leaf peer.

These approaches are structured lookups, where each peer has a well-defined set of information about other peer in the system. The advantage of structured lookup methods is that one can usually make guarantees that data can be reliably found in the system once it is stored.

Another approach is the distributed routing like peer-to-peer(P2P). The approach does not have a bottleneck. To overcome the resilience problems of the centralized and hierarchical routing, some P2P systems developed the notion of symmetric lookup algorithms. Unlike the hierarchical approach, no peer is more important than any other peers as far as the lookup process is concerned, and each peer is involved in only a small fraction of the search path. This approach allows the peer to self-organize into an efficient overlay structure. However, to transfer a query dynamically, a lot of wasteful queryies have occurred in the network. Therefore, the query transfer way of the network must be made efficient. Furthermore, this approach does not provide guarantees on object retrieval.

So, these approaches have problem of scalability by increasing of resource informations. Therefore, we use the algorithm called a hash routing such as Chord and Pastry that provide a self-organizing structured P2P overlay network that can serve as a substrate for large-scale P2P applications. This algorithm incorporates techniques that scale well to large numbers of peers, to locate keys with low latency, to handle dynamic peer arrivals and departures, to ease the maintenance of per-peer routing tables, to balance the distribution of keys evenly among the participating peers.

2.1.1 Pastry Overview

Pastry routes a message to a peer whose peer-ID is numerically closest to the key, in a circular peer-ID space. Each peer maintains both a leaf set and a routing table with peer-IDs and IP addresses of other peers. The entries in row n of the routing table refer to peers whose peer-IDs share the first n digits with the present peer's peer-ID; the N+1th peer-ID digit of a peer in column m of row n equals m. Each Pastry peer has a unique, 128-bit peer-ID by a secure hash of the peer's public key or IP address. And ID is a sequence of digits in base 2^b. As a result, only $\log_{2^b} N$ rows are populated in a peer's routing table on average, if there are N peers participating in the overlay. In a normal routing step, a Pastry peer forwards the message to a peer whose peer-ID shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the present peer's ID. If no such peer is known, the message is forwarded to a peer whose peer-ID shares a prefix with the key as long as the current peer, but is numerically closer to the key than the present peer's ID. Pastry is fully self-organizing. A peer join protocol ensures that a new peer can initialize its leaf set and routing table, and restore all system invariants by exchanging $O(\log N)$ messages. In the event of a peer failure, the invariants can likewise be restored by exchanging $O(\log N)$ messages. Pastry constructs the overlay network in a manner that is aware of the proximity between peers in the underlying Internet. As a result, one can show that Pastry achieves an average delay penalty, the total delay experienced by a Pastry message relative to the delay between source and destination in the Internet.

We can think that the delay time depends on the number of the hops than on the physical distance among peer by the improvements of the network band width. We think of the number of the hops in the network that is X's 2^{24} times of numbers of the peers, based on average hops($H_{cent}, H_{ip}, H_{hash}$) at peer total X. In case of centralized routing, the number of hops doesn't depend on network topology. So, the number of the hops does not change. Also, the number of the hops about the hash routing is $\log(2^{24} \times X)$. In other words, it becomes $\log 2^{24}+\log X$. It thinks of the network form of the routing by the IP. The network that is X's 2^{24} times is composed of the continuation of network which the number of peer is X. In other words, the number of the hops has become an enormous number, $H_{ip} \times 2^{24}$.

	$\times 2^{24}$	...	$\times 2^{64}$
Centralized Routing	H_{cent}		
IP Routing	$2^{24} \times H_{ip}$	...	$2^{64} \times H_{ip}$
Hash Routing	$\log 2^{24} + H_{hash}$	...	$\log 2^{64} + H_{hash}$

When the Peers increase, hash search can reduce loads by distributing indexes. And the number of the maximum hops is equal to or less than $O(\log_{2^b} N)$. So, hash search is the scalable searching with the few hops. Also, the searching of the resource information to change dynamically in keeping information in the network at the condition which is always the latest by the communication of P2P becomes possible.

2.2 Discussions

In this section, we give a problem about existing hash routing. So, we explain the way of solving those problems for hash routing to suit ubiquitous computing environments.

2.2.1 Resource Description for Flexibility

At the hash routing, the hash value that is refined at the time of the advertisement of the resource information becomes the value which depends on the entry. Therefore, only one resource information gets to correspond to one

routing key. In other words, the flexible routing can not be done.

This problem can be solved in refining more than one hash value from one piece of resource information. However, the hash value to refine should relate to the resource description. Therefore, a computer needs to understand a resource description, and each divided descriptions change into the hash value. An effective way would be to extract each attribute-value pair from the XML-Tree, and independently map it onto a key. However, this way would lose the richness of hierarchical descriptions and would not allow expressive queries to be performed. For example, traffic in the attribute sequence traffic-of-road would become indistinguishable from traffic in traffic-of-network. Therefore, we must preserve the description structure and supports partial queries.
For example:

$Input$:< target > human < name > Bob < /name >< /target >

$H1 = Hash(<$ target $>$ human $<$ /target $>)$

$H2 = Hash(<$ target $>$ human $<$ name $>< /$ name $>< /$ target $>)$

$H3 = Hash(<$ target $>$ human $<$ name $>$ Bob $< /$ name $>< /$ target $>)$

Output keys: H1, H2 and H3

2.2.2 handling a Peer as a Resolver

As for the hash routing, the load hangs over the small and low performance devices and so on too much.

Therefore, we do not make all terminals a peer and make high performance terminals such as the home server and the street terminal Peer. The low performance devices send request message to the peer to advertise and search. The low performance devices search the nearest peer by the ad-hoc network which depends on the geographical information, and sends a message to discovered peer.

That is, it divides a network into two layers of the ad-hoc network and the hash routing network. In other words, we can do advertisement and search of resources, do not need to think of the performance of the device. Also, the device does not depend on the specific server, and the whole system can realize high availability.

3 Privacy Information Management

In a ubiquitous computing environment, many services, which are suitable for user, are given naturally. However, for this purpose, some personal informations will be needed. Therefore, if the ubiquitous environment is constructed without thinking about privacy protection, the system can be just a surveillance system, which just flowout the personal informations into networks. To protect this happen, we suggest that users manage their personal information by themselves. So, we try to construct the system for privacy protection by keeping the merit of ubiquitous environments.

3.1 Motivation

In ubiquitous environments, there will be a room called intelligent room in which services, which are suitable for the user, who enter this room with some comunication equipments like PDAs, are given without the user control anything. Before thinking about ubiquitous environmets, we try to construct the system like this intelligent room.

3.2 Design

To let the user manage their personal information, we suggest to make two files called a policy file and a preference file. The policy file is made by a service provider and which defines how they handle the datum which they collect. On the other hand, the preference file is made by the user side, and which defines how much the user allows their datum to be handled.

3.2.1 Intelligent Room

When a user enters into an intelligent room, the equipment, which the user keeps, takes contact with the room's sensor called 'privacy beacon'. This beacon gets hold of all service's policy file information. When this beacon detects a user's entrance, a beacon request to the user to give the information about the place of a preference file. After getting the information, the beacon sends the information about the policy file to preference's application. Then the preferences's application gives their data to policy's application and starts to compare each file's datum.

3.2.2 System

After preparing the policy file and preference file, we put these datum into a data base. Then we construct a Java program which use the JDBC driver to call the database. After the datum are called, the darum are analyzed, and they are compared by that programm, and finally, how the data is handled is output.

How the data is handled is separated into three categories. These categories are 'accept', 'inform', and 'block'. The data which is positioned in the category 'accept', the data is forwarded to the service without a user noticing it. Then for the category 'inform', the data's action can be changed according to circumstances, so a service application gives a notification to a user and the user gives a decision whether the data is allowed to give or not for that purpose. Finally, for the category 'block', the data is refused to be given, so the notification, which says the user refuses to give the service, is sent to the service application.

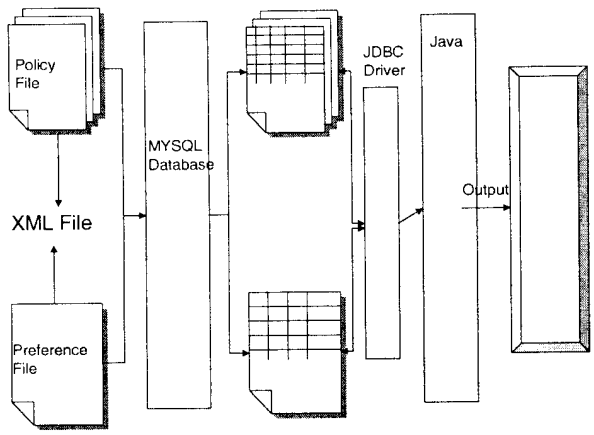

Figure 1: System Overview

3.2.3 Implementation

The policy file and preference file are written by XML. To write XML, we use the standard data schema. This data schema is defined by the project which is made for the privacy protection of the world of the Internet. This project is made by W3C(World Wide Web Consourtium)which is a party of standization of application for the Internet. The project is called P3P(Platform for Privacy Preferences Project). Adding that, we make the preference file also by using the idea of this project, this project defines a language called APPEL(A P3P Preference Exchange Language). Therefore, to write the preference file, we use the method of this APPEL.

3.2.4 Data Comparison

For this research, we use MySQL Database because it is good for reference speed. Using the JDBC driver, it becomes possible to call the database from a Java program. Then, called datum are put into arrangement. After each file's data is called from that arragement, the same field data from each file is compared by the contents. After comparing the contents, each data is distributed to suitable behavior which is defined by the preference file.

3.3 Discussions

It will be important to think about privacy protection when ubiquitous environments are actualized. However, technical idea can not be enough, and legal issue will also be important. Even if we can construct a privacy protection system as hard as possible, some malicious people will try to break it. Therefore, this system must be protected legally. However, the problem is that in the present circumstances, only few countries take care of internet privacy protection problems. Actually, to make this system goes through, some agreements, which define that the service provider must define thier use correctly, must be settled legally and when they betray some punishment must be settled.

3.3.1 The Law

Actually, a law for privacy protection in the Internet is not settled in most of country. Though some countries start to think about the privacy protection's law, there is still a big problem. That is that the world of the Internet is no boundry world. The law for the Internet relevace must be unique for all of the world. Now many projects are trying to standardize the arrangement and many countries start to agree theses arrangement. By the time the ubiquitous environments are actualize, we hope that many countries join in the idea of standardization.

4 Conclusions and Future Directions

The paper describe two system supports for sentient materials. The first one is a communication infrastructure to support to build a scalable database for context information. The second one is a privacy information management system. The system protects our privacy information from sensors embedded in our environments.

Currently, we are working on a system support to provide high level abstraction about low level sensors[3]. Also, the middleware supports to compose several sensors. In the future, we like to integrate these system supports to support sentient materials.

References

[1] W. Butera, "Programming a Paintable Computer", MIT Ph.D Thesis, 2002.

[2] L.H.Lifton, "Pushpin Computing: A Platform for Distributed Sensor Networks", MIT Master Thesis, 2002.

[3] K.Fujinami, T.Nakajima, "A Framework for Development of Context-Aware Applications in a Ubiquitous Computing Environment", IPSJ Computer System Synposium, (In Japanese), 2002.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.

[5] I.Stoica, R.Morris, D.Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", ACM SIGCOMM 2001, 2001.