Present Scenarios and Future Challenges in Pervasive Middleware

Amitava Mukherjee

School of Computer Science and Engineering
University of New South Wales
Sydney 2052, Australia
amitavam@cse.unsw.edu.au

Debashis Saha

MIS & Computer Science Group
Indian Institute of Management (IIM) Calcutta
Joka, Kolkata 700 104, India
ds@iimcal.ac.in

December 2003

UNSW-CSE-TR- 0337

Abstract- *In order to run applications on pervasive devices, pervasive middleware has to support context-awareness, as pervasive applications need to adapt to variations of context of execution (such as network bandwidth, battery power and screen size), physical change of locations, change of technological artifacts (devices), change of hardware resources of artifacts, and so on. Recent research efforts have primarily focused on designing new mobile middleware systems capable of supporting the requirements imposed by mobility. However, apart from mobility constraint, pervasive middleware will operate under above-mentioned conditions of a radical change. This change is varying from physical components (like network heterogeneity) to functional components (right from heterogeneous devices to context-based applications). Few contemporary researches have indeed focused on some parts of these requirements; but a qualitative difference between intended requirements and practical achievements still remains there. In this article, we discuss some of recent mobile/pervasive middleware systems, focusing on research issues and challenges ahead to bridge the gap. Typically, we highlight the key characteristics of pervasive middleware to support context awareness and service discovery, smartness and adaptation, heterogeneity and integration, and intelligent interfacing.*

Keywords: Pervasive computing, middleware, components, heterogeneity, context-awareness, interfaces.

Introduction

Pervasive computing [1]-[2] is “omni-computing”. It is “all-pervasive” by combining open standards-based applications with everyday activities. Pervasive computing environments increase users’ thought and activities with an all *pervasive* information processing and analysis. It provides an environment to perform day-to-day activities that are enhanced by behavioral contexts of users. In the vision [1] of pervasive computing, the environment is saturated with a host of computing and communication capabilities which are gracefully integrated with daily life so that user will be able to exchange information and control their environments from everywhere using a seemingly invisible infrastructure of various wireline and/or wireless networks and computing devices [2]. Various peer-to-peer computation and communication devices are creating this environment to facilitate users’ everyday tasks and to increase productivity as well. This, in turn, is facilitating the construction of new classes of applications that will be embedded in the physical environments and integrated seamlessly with user’s everyday tasks. These applications pose a number of new challenges for the existing middleware (a bundle of firmware and/or software executing in either client-server or peer-to-peer mode) technology to live up to the desired level of expectation. This is because of the presumption of the following architectural model (Figure 1) for pervasive computing [2]. Similar to the model of distributed computing and mobile computing, in pervasive computing too, a shell of middleware is essential to interface between the pervasive network kernel and the end-user applications running on pervasive devices. This middleware will be responsible for keeping the users immersed in the pervasive computing space and for mediating all interactions with the kernel on behalf of the user.

An Example

A pervasive computing environment may have limited hardware resources, heterogeneity in networks and devices, dynamic changes in physical environment and different applications (services) to run on. To understand such a scenario, we consider an example.

Ray, manager of an international Consulting Company, is attending a conference at Sydney with his PDA. He uses the PDA to access information in a wireless environment dynamically via the conference server. Ray performs several activities during a day of the conference: access the marketing and technical presentations electronically from the conference web site, select the schedule to attend the presentation, check timing of cocktail party and cultural functions of conference in the evening, get some alert messages during the conference to send mail/message to his secretary to run business at his office.

When Ray enters the conference hotel, he checks the conference program browsing through PDA to select few presentations to attend. Before attending the presentation, he decides to look at the abstract of those presentations through on-line conference proceedings. While doing so, he meets Robert, manger of one of his client's company, who is also attending the same conference. They move to swimming pool to finish pending discussion that they had on last evening through conference call. As they approach the pool, Ray finds his PDA taking a significant amount of time to access the abstract of the paper because the quality of network connection deteriorates. While *moving to another location to get a better connectivity, they loose connectivity to access the proceedings.*

They come back to conference site to access the contents of the proceeding and attend the talk. In the meantime, a reminder beeps on the screen to send messages to his secretary. After a few minutes, another beep flashes on the screen to remind him to present his talk after 15 minutes. When he prepares to send messages from his PDA, a message flashed on the screen to alert him to proceed to a nearby call-room to get a better hardware infrastructure and network connection. He sends a brief message from call-room using a notebook computer to his secretary to instruct her to prepare the schedule on next day's client meeting. He also has a discussion on day's business activities. A beep flashed on the screen of notebook computer to alert him to present his talk at the conference room.

In the above example, we observe a few important aspects. Ray's working environment has changed very frequently. His location changes dynamically as he moves from the conference site to swimming pool to call-room back to conference room. In his environment, functional components, such as devices, services and resources, are changed frequently. Physical components, such as bandwidth, memory availability of his PDA, and battery power varies on his change of locations. All functional and physical components pointed here are the examples of context that need to be adapted to the dynamism of the user in the environment. As the user is highly mobile, it is impossible to know a priori the uncertainty in user's requirements. So the system would require support for dynamic reconfiguration. When the user enters the conference hotel, his PDA does not know physical and functional components in the environment. So a discovery mechanism provides information about these components. Sometimes the user might need to actively configure the system to adapt the services available in a certain environment. In addition to middleware reconfiguration, software infrastructure associated with the environment is required to discover existing devices and services and learn their facilities offered and reconfigure them when changes are detected (in our case a message flash on PDA to alert Ray to precede for better resources and service).

Pervasive Middleware Characteristics

In particular, future middleware that would support the construction of pervasive applications should include new levels of component interoperability and extensibility, and new dependability guarantees with QoS-based context-aware services, including adaptation to changing environments and tolerance of disconnected operations under heterogeneous environment. Current researches in mobile middleware assume homogeneous network environments. So there remains the problem of interoperating with heterogeneous middleware technologies that provide different asynchronous communication paradigms to cope with frequent disconnections (typically common in a pervasive network environment). Above all, simplicity at the middleware level is important to the deployment and the administration of any pervasive application. Even if each device is equipped with an appropriate interface, the shear multitude of devices will make it virtually impossible for the owner to take on administrative responsibilities manually. When a device gets installed initially, users should be able to just plug the device in, and it should start to work immediately with no hassles. In order to do so, it must be able to self-configure and activate itself.

Researchers in pervasive middleware have focused on similar challenges and pursued them in various research endeavors [2]-[4]. From their works, we can conclude that, in general, a pervasive middleware is characterized by following major paradigm shifts:

- *Context awareness and service discovery*: learn the environment so that the interactions between services and devices are made proper to get a desired service.
- *Smartness and adaptation*: construct, manipulate and display environments, resources and contents for any services.
- *Heterogeneity and integration*: handle different applications on different devices connected to different heterogeneous network environments, and integrate a number of parameters such as QoS, service reliability, invisibility etc.
- *Programming interface*: address issues related to service adaptation and integration
- *Disconnectivity*: protect services and applications from transient failures when users or devices go out of the range of wireless connectivity.
- *Security*: minimize threats to privacy, inspite of being pervasive.

Current generation of mainstream middleware [5] is, to a large extent, heavyweight, monolithic and inflexible, and, thus, fails to properly address the afore-mentioned new requirements of pervasive applications. Traditional middleware systems have been built adhering to the principle of transparency: implementation details are hidden from both users and application designers and are encapsulated inside the middleware itself, so that the distributed system appears as a single integrated computing facility to application developers [5]. Though successful for building traditional distributed systems, this approach suffers from severe limitations, when applied to pervasive computing, where it is neither always possible, nor desirable, to hide all the implementation details from either the user [5], [6] or the developer.

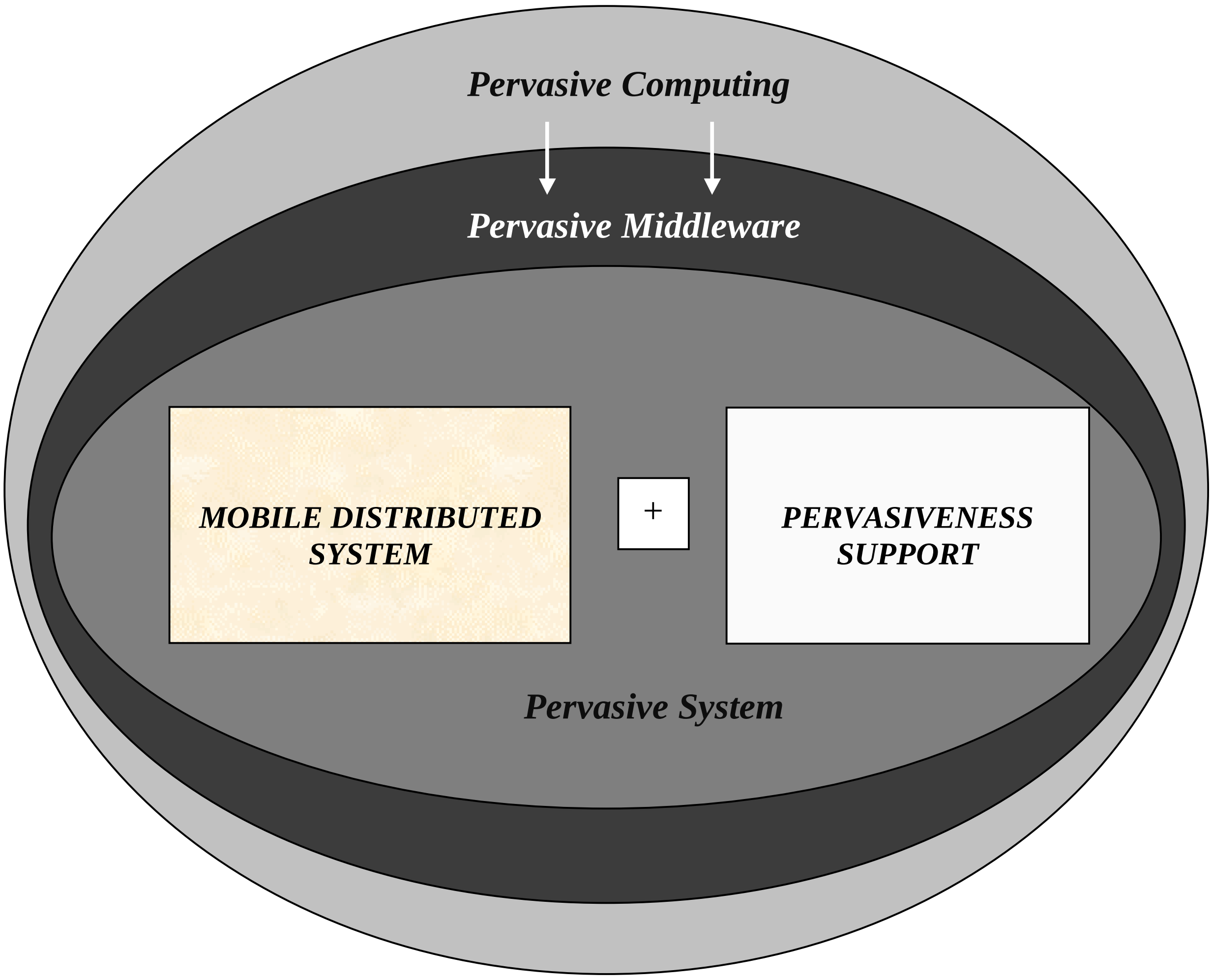

Figure 1**:** ***Pervasive Computing space [2]***

Most of the existing research efforts [2]-[4], [6] have focused on designing middleware capable of supporting only the requirements imposed by mobility. A concept of awareness is introduced recently in [7] to break the high level of abstraction (transparency) for targeted mobile middleware. This approach allows developers to build applications to be aware of their execution context and to adapt their behavior accordingly. This balance between awareness and transparency has added a new dimension to middleware research [7], [8]. However, apart from mobility constraint, pervasive middleware will operate in different conditions of a radical change. This change is varying from physical components (like network heterogeneity) to functional components right from heterogeneous devices to context-based applications [2]. Few contemporary researches [2], [4], [6] have indeed focused on some parts of these requirements, but a qualitative difference between intended requirements and practical achievements still remains there. In this article, we discuss today's mobile/pervasive middleware systems, focusing on research issues and challenges ahead to bridge the gap. Typically, we highlight the key characteristics of pervasive middleware to support context awareness and service discovery, smartness and adaptation, and heterogeneity and integration, and a multitude of programming interfaces in pervasive systems.

Pervasive Middleware Components

As hinted in Introduction, the success of traditional middleware systems is attributed to the principle of offering a distribution abstraction (transparency) to both developers and users, so that a system appears as a single computing facility. These middleware provide built-in mechanisms and policies to support development for fixed distributed systems in wired network environments (not for wireless networks). This high level abstraction of the underlying technology and environment unfortunately makes a little impact on dealing the specific issues, such as heterogeneity and dynamism, of pervasive systems.

In order to make a middleware usable in different pervasive domains, it must have a reusable framework to facilitate services in these domains. We highlight the following three prime design components of a pervasive middleware:

- *Proactive Knowledge on Environment*: To discover proactively network bandwidth, nature of communication, types of devices and their functionalities, such as storage capacity, input/output capability and battery power. Pervasive middleware should facilitate a transparent communication model to applications to flexibly interact with different devices in different network environments. For example, it should notify the appropriate network layers to take actions, when an incompatibility in networks and devices becomes imminent for an application [2].
- *Building Applications on Context-awareness*: To develop systems which determine user tasks in different contexts, such as profile history, preferences, societal behavior and environmental conditions. An application is usually synthesized to suit tasks, associated with components and services. When the application is instantiated with a device (i.e., integration of applications with devices), it should be able to move seamlessly from that device to another device and even from one environment to another environment. Moreover, pervasive middleware should support applications to scale to add new contexts (or, to modify the existing context) for large systems. For instance, pervasive middleware should be able to provide the facility to recover from intermittent network failures.
- *Appropriate Programming Interface*: To express different activities, preferences of users, and different characteristics of physical and functional computing components. Future programming languages will support for expressing context-awareness on a conceptual level that will be different from existing programming languages. In essence, the semantic modeling in pervasive middlewares should provide a uniform

and common way to express context-awareness for users' various activities in their applications.

Additionally, a pervasive middleware need some more common capabilities, such as lightweight design and low energy-consumption [4], [8], typically found with a mobile middleware.

Recent Research Endeavors

In order to fix the limitations of legacy middleware, following research projects have focused on middleware technologies to support some scenario of pervasive computing. This results in a group of new/upgraded middleware that has been categorized below in terms of their research objectives and the key issues addressed by them.

Projects	***Objectives***	***Key Issues***
RCSM [8]	To develop a middleware that facilitates applications those require context awareness in mobile ad hoc communications.	**Context awareness in applications during development and runtime operation**: It combines the characteristics of context awareness and ad hoc communications in a way to facilitate running complex applications on devices.
X-Middle [9]	To develop a mobile middleware which supports building applications that use both replication and	**Disconnected operations in mobile applications:** It allows mobile users to share data when they are connected, or replicate the data and perform operations on them off-line when they are

	reconciliation over ad-hoc networks.	disconnected; data reconciliation takes place when user gets reconnected.
Gaia [10]	To build a distributed infrastructure where a middleware coordinates software entities and heterogeneous devices.	**Dynamic adaptation to the context of mobile applications:** It supports the development and execution of portable applications in active spaces.
Environment Awareness Notification Architecture [11]	To develop a middleware for event notification by a mobile computing environment to applications.	**Scarce resources of mobile devices and dynamicity of the mobile environment:** It models the environment as an asynchronous event that includes the information related to the change.
UIC [6]	To develop a reflective middleware (composed of a pluggable set of components) for mobile devices.	**Heterogeneity of devices and networks:** It helps users to specialize to the particular properties of different devices and network environments.
Nexus [12]	To develop a middleware that supports location-aware applications for mobile users.	**Heterogeneity in networks**: It provides an infrastructure that supports communication in heterogeneous network environments.
Lime [13]	To develop a	**Programming constructs which are**

	middleware that supports physical mobility of hosts, logical mobility of agents, or both.	**sensitive to the mobility constraints:** It explores the idea by providing programmers with a global virtual data structure and a tuple space (Tspace), whose content is determined by the connectivity among mobile hosts.
Tspaces [14]	To develop a middleware that support communication, computation and data-management on hand-held devices.	**Asynchronous messaging-based communication facilities without any explicit support for context-awareness:** It explores the idea of combination of tuple space (Tspace) and a database that is implemented in Java. Tspace targets nomadic environment where server contains tuple databases, reachable by mobile devices roaming around.
L2imbo [15]	To develop a middleware that emphasizes on QoS support in mobile applications.	**QoS monitoring and control by adapting applications in mobile computing environment**: It provides the facilities of multiple spaces, tuple hierarchy, and QoS attributes.
Aura [16]	To develop a pervasive infrastructure involving wireless networks,	**Distraction-free pervasive computing:** It develops the system architecture, algorithms, interfaces and evaluation

	wearable or handheld devices and smart spaces.	techniques to meet the goal of pervasive computing.

Research Challenges

A critical review of the above projects helps us identify the following immediate challenges facing the researchers in this domain [2]-[4].

Service discovery

Service discovery dynamically locates a task that matches a user's requirements. Traditional naming and trading service discovery techniques, developed for fixed distributed systems, where intermittent (rather than continuous) network connection is the practice, cannot be used successfully in pervasive computing environments. It might solve the association problem to interact components (tasks) with services. But, the research challenge would be to make use of this task (needs) to discover services in an entire pervasive computing environment that would be able to give services to users based on QoS-aware specifications. In QoS-aware service discovery, application needs are made explicit and are used to decide how a service would be delivered to users in the current context. For example, L2imbo [15] has developed a middleware to support QoS in mobile applications. However, currently these needs are taken into account locally only. The real issue is to resolve the following concern: "Hundreds or even thousands of devices and components (tasks) might exist per cubic meter; with which of these, if any, is it appropriate for the arriving component to interact" [4].

Context awareness

To accommodate context-awareness [2] of pervasive computing, pervasive middleware must have facilities for both deployment-time and run-time configurability. Context (or extension) with respect to an application is to bind and re-bind a number of pervasive devices to facilitate the continuity of applications running on. The research challenge will be that N classes of applications will have to adopt M numbers devices (N-to-M) instead of doing a single new application to a group of devices (1-to-M). The development of an appropriate program model becomes a challenge to express common semantics to develop tasks (activities), validate tasks on different of physical environments and devices, and finally share tasks by different applications (services). To achieve this goal, in recent research projects [8], [13], [14], task components interact with services by sharing a tuple space or an event service [11], or data-oriented services [9]. Some researchers have pointed out that data-oriented interaction might be a promising model that has shown its value for spontaneous interaction inside the boundaries of individual environments. It seems that this requires ubiquitous data standardization for it to work across environment boundaries.

Adaptation

To accommodate dynamic requirements and preferences of users, a set of services and polices need to be installed and uninstalled spontaneously. Different adaptation schemes need different system configurations that vary over time. The changing interactions among distributed services and policies may alter the semantics of the applications built on top of the middleware. Most of the pervasive devices have limited and dynamically varying computational resources. These embedded components, used in pervasive devices, are small, and limit to constrained resources. Additionally, as portable devices run on batteries, a trade-off exists between battery life and computational ability (or network communication). Here, a research challenge arises because adaptation must often take place without human intervention to achieve *calm computing* [1]. Possible extensions of existing mobile middleware are to include transformation and adaptation for content and human interface in terms of context in pervasive applications.

Heterogeneity

Future pervasive systems will be heterogeneous in the sense that many different devices will be available on the market, with possibly different operating systems and user interfaces. Network connectivity will also be heterogeneous, even if an effort towards complete convergence through different connection technologies will be made. To accommodate this variety of heterogeneities, pervasive middleware must have a facility (in terms of programming interfaces) to adapt to the jitter in environments at both start-up time and run-time. For example, RCSM [8] has facilitated applications to adapt to network heterogeneity by providing development and run-time supports. On another front, the message-passing communication paradigm has already been tried to support disconnections; for example, X-middle [9] has categorically addressed this issue to support disconnections of devices with networks.

Semantic modeling

Semantic modeling addresses the challenge of programming interfaces in applications for various user activities. The challenge is to develop a rich programming semantic to handle different ontologies for various tasks in application domains. This high-level semantics would describe the dynamic preferences, characteristics and services. Ontologies [3] describe different task environments to adapt the changes in user environments (for example, Roy moves around to adapt different environments to run his different activities during attending the conference).

Conclusion

Recent research efforts in middleware technology have addressed only a few scenario of pervasive computing in their works. Truly speaking, researchers have focused on some specific middleware contexts to meet typical aspects of mobile wireless networks. Moreover, their prototypes have their own unique architectures and semantics, which rarely lead to a generic framework. These existing implementations have not directly addressed most of the characteristics (mentioned in the earlier sections) of pervasive computing. Further, heterogeneity in these middleware may pose difficult interoperability hurdles in future to a pervasive computing practitioner engaged in developing real-world applications in the present structure. Nevertheless, in order to reach Mark Weiser's vision [1] of "*calm computing*", future pervasive middleware must take a key role to integrate physical and functional components to run different applications seamlessly so that, when Ray, in our example, would visit the next conference in 2005, he would not face hassles and social uncomfort any more.

References

1. Weiser, M. The computer for the 21st Century. *Scientific American* (Sept.1991).
2. Saha, D. and Mukherjee, A. Pervasive Computing: A Paradigm for the 21st Century. *IEEE Computer Magazine* (Mar 2003).
3. Banavar, G et al. Software Infrastructure and Design Challenges for Ubiquitous Computing Applications. *CACM* (Dec 2002).

4. Kindberg, T. and Fox, A. System Software for Ubiquitous Computing. *IEEE Pervasive Computing* (Jan 2002).
5. Geihs, K. Middleware Challenges Ahead. *IEEE Computer Magazine* (Jun 2001).
6. Mascolo, C et al. Middleware for Mobile Computing. *Advanced Lecture Notes, Springer LNCS 2497* (2002).
7. Mascolo, C et al. Middleware for mobile computing: awareness vs transparency. *Proceeding 8th Workshop Hot Topics in Operating Systems, IEEE CS Press* (May 2001).
8. Yau, S S et al. Reconfigurable Context-Sensitive Middleware for Pervasive Computing. *IEEE Pervasive Computing* (Jul 2002).
9. Mascolo, C et al. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *International Journal on Wireless Personal Communications* (2002).
10. Roman, M et al. A middleware Infrastructure for active spaces. *IEEE Pervasive Computing* (Oct 2002).
11. Welling, G. and Badrinath, B.R. An architecture for exporting environment awareness to mobile computing applications. *IEEE Trans on Software Engineering* (May 1988).
12. Fritsch, D et al. NEXUS Positioning and Data Management Concepts for Location Aware Applications. *Proceedings of the 2nd International Symposium on Telegeoprocessing, France* (2000).
13. Murphy, A. L et al. Lime: A Middleware for Physical and Logical Mobility. *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)* (May 2001).

14. TSpaces Project, IBM Research, *http://www.almaden.ibm.com/cs/TSpaces/* (2000).

15. Davies, N et al. L2imbo: A Distributed Systems Platform for Mobile Computing. *ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks* (1998).

16. Garlan, D et al. Project Aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing* (Apr 2002).