ARTICLE IN PRESS

Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Middleware for pervasive computing: A survey

Vaskar Raychoudhury [a,*], Jiannong Cao [b], Mohan Kumar [c], Daqiang Zhang [d]

[a] Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, India
[b] Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[c] Department of Computer Science and Engineering, The University of Texas at Arlington, TX, USA
[d] School of Computer Science and Engineering, Nanjing Normal University, Nanjing, China

ARTICLE INFO

Article history:
Received 7 December 2011
Received in revised form 10 June 2012
Accepted 23 August 2012
Available online xxxx

Keywords:
Pervasive computing
Middleware
Context management
Service management
Fault tolerance

ABSTRACT

The rapidly emerging area of pervasive computing faces many challenging research issues critical to application developers. Wide heterogeneity of hardware, software, and network resources pose veritable coordination problems and demand thorough knowledge of individual elements and technologies. In order to ease this problem and to aid application developers, different middleware platforms have been proposed by researchers. Though the existing middleware solutions are useful, they themselves have varied features and contribute partially, for context, data, or service management related application developments. There is no single middleware solution that can address a majority of pervasive computing application development issues, due to the diverse underlying challenges. In this survey paper, we identify different design dimensions of pervasive computing middleware and investigate their use in providing various system services. In-depth analysis of the system services have been carried out and middleware systems have been carefully studied. With a view to aid future middleware developers, we also identify some challenging open research issues that have received little or no attention in existing middleware solutions.

1. Introduction

Pervasive computing (PvC) aims to create a smart environment with embedded and networked computing devices, providing human users with seamless service access. As the primary focus of PvC is human-centricity, autonomous detection of application requirements and automatic service provisioning are the two keys to PvC middleware. Application requirements depend on user context and needs, whereas available resources drive service provisioning capabilities. The goal of PvC middleware is to match application-needs to available resources, while ensuring quality and efficiency. Applications of PvC are wide-spread e.g., smart-spaces [1–6], health-care [7–9], assisted-living [10], social networking, entertainment [11], logistics and intelligent transportation systems. Altogether, PvC concepts are enabling existing computing infrastructure to enhance human quality of life in a truly ubiquitous manner.

The enabling technologies of PvC [12] consist of smart device technologies, wireless communications (Wi-Fi, Bluetooth, Zigbee, LTE, NFC, etc.), software, and enhanced human–computer interaction (HCI) techniques. Developments in smart device technologies include sensors (e.g., UC Berkeley Motes Sensor Network Platform), Radio Frequency ID (RFID) tags, intelligent appliances, embedded processors, wearable computers, handheld computers, smart phones, and many

* Corresponding author.
E-mail addresses: vaskar@ieee.org (V. Raychoudhury), csjcao@comp.polyu.edu.hk (J. Cao), mkumar@uta.edu (M. Kumar), dqzhang@njnu.edu.cn (D. Zhang).

1574-1192/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2012.08.006

others. Tiny intelligent sensors have made it possible to deploy ubiquitous services, and thus create various smart environments. RFID tags allow subtle integration of physical objects (e.g., commodities in a superstore, or books in a library) into the computing environment. So, in summary, the boundaries of PvC have been broadened by the development of new technologies, as well as the extensive use of existing technologies, such as, the Internet, mobile and wireless communications, sensor networks, and RFID technology.

Consider the following motivating scenario to appreciate the benefits of an effective middleware for PvC. *Mr. Smith is 84-year-old and lives alone in a remotely located house. He has common old-age ailments along with chronic heart-related disorders. He often forgets to take his routine medications, and twice he fell down in the shower and remained unconscious for a while. Recently, a smart elderly-care system was installed in Mr. Smith's house. The system monitors the physical conditions of Mr. Smith, and his daily activities through different embedded (pressure, weight, and location sensor), and wearable (blood pressure, heart, and pulse rate monitor) sensor nodes and other smart devices, such as mobile phone, laptop, PDA, etc. The care system also employs a wearable fall-detection system to detect a victim's fall in a two-stage process. One afternoon, Mr. Smith became suddenly ill, fell down from the sofa, and lost consciousness. The following actions would take place if his home is PvC enabled. Initially, an impact sensor detects a sound whose characteristics are that of a crash or fall. Then, a second sensor captures the orientation and movement of Mr. Smith. If he is immobile for a time period higher than a threshold value, an alarm is raised by the collaborating sensors. Then, a number of activities are triggered to check Mr. Smith's heartbeat, pulse rate, and blood pressure, through different wearable sensors. This information is forwarded to the patient's physician, through his smart phone, after verifying the receiver's identity. The physician then tries to identify, remotely, the causes of Mr. Smith's sickness, and the effects. The physician executes services to logically conclude whether Mr. Smith had his lunch or not, by considering his usual lunch time, condition of his dining table captured by a camera, and any post-lunch physical effect in his body, captured through wearable sensors. The physician also needs to know whether Mr. Smith had taken his usual medications, by checking his recorded action of opening medical cabinet along with any dip in the level/decrease in the number of medicines. In order to judge the seriousness of Mr. Smith's fall, the physician may also need to detect chances of bleeding, by analyzing the presence of body fluid around Mr. Smith, and then checking whether that is blood, or not.* Correctly deducing the afore-mentioned decisions is non-trivial, and requires intricate PvC support.

The above *elderly-care system* system faces several challenging issues. Firstly, the system employs multiple wired, as well as networked embedded sensor nodes. Coordination and management of these large numbers of sensors are non-trivial tasks. These resource-constrained hardware entities may suffer frequent failures, due to energy depletion, or processing power and memory size limitation. Secondly, the sensory devices continuously keep collecting huge amounts of raw data about the actions and physical conditions of the user. Managing huge sets of raw data, storing and processing them, are also non-trivial. Moreover, this raw data needs to be processed and reasoned with properly in order to capture meaningful context information about the user. E.g., the sensor nodes embedded in the fall detector can collect raw data and then can logically conclude about a fall of Mr. Smith. While incorrect context-reasoning may generate false alarms of a fall (false positive) it may also fail to detect a true fall (false negative). However, for reliable operation of the fall detection system, fall events should not be missed while false positives should be minimized. Thirdly, the application also needs to detect services required for the user, subject to certain constraints or conditions. E.g, in the above application, the system needs to detect whether Mr. Smith had his lunch or not, or whether he had taken his medicines, or even if he is bleeding due to the impact caused by the fall. However, such a service may be composed of several small or elementary services and require the support of video footage or sensory gadgets to logically deduce the occurrence of an incident (as mentioned in the application scenario). Managing all the elementary services and facilitating service composition based on the user requirements are rather complicated for application developers. Finally, reliability and security are two other core concerns of PvC applications. Failure of hardware entities (sensors and smart devices) or software functionalities (context and service management) can equally affect the system, and can thwart proper functioning, which may lead to serious or fatal consequences for the user. Security concerns are associated with divulging sensitive user information to unauthorized parties. The above application scenario necessitates verifying the identity of the physician before sharing the health information of Mr. Smith.

The above challenges clearly manifest the difficulty of PvC application development by programmers, while addressing all the concerns. Thus, it requires the aid of an efficient middleware platform in order to manage the wide array of devices, technologies and functionalities. Middleware refers to software and tools that can help hide the complexity and heterogeneity of the underlying hardware and network platforms, can ease the management of system resources, and can increase the predictability of application executions. This paper is aimed at conducting a survey of available middleware platforms for PvC.

Two major research objectives of PvC middleware research are about addressing high-level application requirements on one hand, and the complexity of the operations in the underlying devices, networks and platforms on the other. The requirements of pervasive applications are very much application-specific. But usually they include high flexibility, re-usability, reliability, localised scalability, adaptability and context-awareness. The complexity of operations with PvC applications are characterized by resource-constraints (in terms of low computation capability and insufficient battery power) and fault-proneness of devices, dynamicity of network topologies, requirements of interfacing with a myriad of hardware and network protocols, programming of multiple devices and their interactions, and functionalities for context, data and service management. PvC middleware provides a potential solution to address the above issues, while easing the task of PvC application development for the developers.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

Although middleware is a well-established research area in distributed computing systems, PvC poses new challenges to middleware research. The traditional middleware design techniques cannot be applied directly to design PvC middleware. First, PvC environments often showcase far greater heterogeneity, in terms of devices, network protocols, and operating systems, than traditional distributed and mobile environments. PvC middleware must be able to hide the complexity and heterogeneity of underlying entities and their interactions, and abstract them into intuitive and accessible programming constructs. Second, in order to be able to provide situation-aware services to the users, PvC applications often require features such as context-awareness and service-orientation. Special mechanisms are necessary to support those objectives, in the form of raw context collection, storage and processing, higher-level context derivation, and context inconsistency resolution. Service discovery and composition are also equally important to provide users with unconstrained service support, all the time and everywhere. Third, due to their human-centric nature, PvC applications are often safety-critical, and need to support users unobtrusively, without requiring any intervention. This requirement poses many reliability challenges. Elderly-care or automated traffic management applications can hardly afford system failure or malfunctioning. Finally, PvC requires the middleware to be lightweight for implementation in sensor nodes or other embedded and portable devices having limited processing powers and energy backups. PvC also has new requirements on hardware (e.g., various embedded devices), operating systems, routing protocols, and applications.

In recent years, many works have been done on PvC middleware, focusing on different aspects, and for different purposes. Existing PvC middleware systems including, Gaia [13,14], Aura [15–17], PICO/SeSCo [18–20], CORTEX [21], One.World [22,23], Scenes [24], Activity-oriented computing [25], and UIO [26] are popular and showcase the different design choices of the PvC middleware paradigm. Although several survey papers can be found in literature, focusing on context modeling techniques [27–30] and service discovery protocols for PvC environments [31–33], the current paper is a comprehensive survey, which summarizes the general issues, characteristics and design considerations of available PvC middleware solutions. This paper presents a systematic study of recent research on PvC middleware to help identify the key services, challenging issues, and important techniques. The key contributions of the paper are as follows. Firstly, a reference model has been proposed for analyzing the functionalities and key services of PvC-middleware. Secondly, the paper provides a detailed review of the existing work on the most important aspects in developing PvC middleware, covering the major approaches to and corresponding techniques of implementing services. Finally, the paper proposes a feature tree-based taxonomy that organizes PvC-middleware features and their relationships into a framework to help understand and classify the existing work. The paper also discusses open problems and identifies directions in future research.

The remainder of this paper is organized as follows. In Section 2, we describe a reference model of generic PvC middleware and describe the design dimensions identified. In Section 3, we discuss in depth the core system services provided by PvC middleware systems. In Section 4, we discuss the challenges, open problems, and future directions of PvC middleware research. Finally, we conclude this paper in Section 5.

2. Reference model for PvC middleware

Before setting out to propose a reference model to classify existing PvC middleware systems, we have studied a large number of research papers on middleware system survey. There are some noteworthy survey papers on middleware paradigms for mobile computing [34], pervasive computing [35], and wireless sensor networks (WSN) [36] which are closely related to our objective. In their survey on middleware for WSN, Wang et al. [36] have used a reference model based on programming abstractions, system services, runtime support, and QoS mechanisms, among which the first three have been used by us in order to classify PvC middleware solutions as well. Schiele et al. [35], in their survey of PvC middleware systems, have introduced three dimensions for classifying the existing PvC middleware systems—the organizational model, the level of abstractions, and the supported tasks. While the first one is similar to our node and system level abstractions, the second one is more about the benefits of using abstraction, such as, providing transparency, automation, etc. The last one discusses the main services, according to the authors, which are provided by PvC middleware systems, such as support for spontaneous interaction, support for context management, and support for application adaptation. So, in general, middleware systems have traditionally been based on providing some kind of abstractions as well as system services and, keeping that in mind, below we shall propose our reference model to classify existing PvC middleware systems.

2.1. Model overview

Analyzing the generally adopted techniques of system design, we have identified the following three design dimensions (Fig. 1) of a PvC middleware system: (1) programming abstractions, (2) system architectures, and (3) system services and runtime support. Programming abstractions define the interface of the middleware to the application programmer. System services provide implementations to achieve the abstractions. Runtime support serves as an extension of the embedded operating system to support the middleware services. The three design dimensions are drawn as mutually perpendicular axes in order to show their independence or orthogonality. System designers can make any design choice under a particular design dimension, without affecting design choices under other design dimensions. E.g., for designing a new PvC middleware, any programming abstraction can be chosen for any system architecture, and any number of system services can be provided over that new middleware, without mutually affecting the different design choices.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

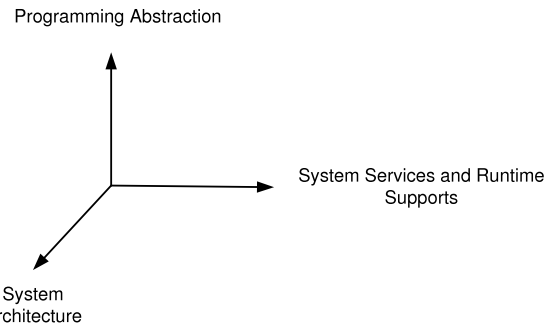

Fig. 1. Design dimensions of a PvC middleware.

By analysing the characteristics and requirements of the elderly-care PvC application mentioned in Section 1, we propose a reference framework, shown in Fig. 2, which identifies the core system services and runtime supports required to be provided by a standard PvC middleware. It should be mentioned that it is not necessary for a specific PvC-middleware to include all the components. Also, functions of several components may be combined together and implemented as one component. Moreover, the choice of underlying system architectures is important for proper functioning of a PvC middleware. In the deployment, the functions of PvC-middleware can be distributed to the sensor nodes, other small computing entities, and high-level application servers. The distributed middleware components, located in different nodes of the network, communicate with each other to achieve some common goals.

2.2. Programming abstractions

Programming abstractions are the foundations of PvC-middleware. They provide high-level programming interfaces to the application programmer, with the goal of separating the development of PvC applications from the operations in the underlying PvC infrastructures. They also provide the basis for developing desirable middleware services. We have identified three aspects that are involved in developing the programming abstractions: *abstraction level*, *programming paradigm*, and *interface type*.

Abstraction level refers to how the application programmer views the system. We have studied many existing PvC middleware solutions and have identified two principal types of abstraction levels—*node level* and *system level*. *Node level abstraction* abstracts the PvC environment as a distributed system consisting of a collection of heterogeneous computing devices, and provides programming support for individual devices for their actions and cooperation. Node level abstractions are used by Aura, PICO/SeSCo, CORTEX and Activity-oriented computing (AoC) middleware. Aura uses *Task* abstraction, which represents user applications composed of multiple abstract services called *Suppliers*. Another type of abstraction called *Connectors* is abstraction of interconnections between the system components. AoC uses a similar abstraction to Task used in Aura, called Activity, which is the abstraction of user actions. Activity as an abstraction has been proposed by the activity-oriented computing paradigm [37] under the purview of PvC. Used in different middleware solutions [38–40], activities are computational abstractions which can be initiated, suspended, stored, and resumed on any computing device (hides heterogeneity of the underlying computing platform), at any point in time in a context-aware manner, and can be handed over to other persons, or shared among several persons. Challenges in activity management include, handling parallel activities, interruptions, mobility, sharing, collaboration, coordination, etc.

PICO/SeSCo employs two types of abstractions—*node-based* and *event-based*. *Delegent*, is an abstraction of the mobile software agent, and *device* is an abstraction of computing devices. Delegents are enabled by events that take place in the environment. PICO/SeSCo uses event abstraction to spawn services provided by delegents.

CORTEX uses *sentient object* abstraction which represents smart objects as a single entity. Sometimes, the whole of the smart environment is called a *sentient object*, which then stands for a system level abstraction.

System level abstraction depicts the PvC environment as a single virtual system. It allows the programmer to express a single centralized program (global behavior) into subprograms that can execute on local nodes (nodal behavior), leaving only a small set of programming primitives for the programmer, while making transparent the low-level concerns, such as the distributed code generation, remote data access and management, and inter-node program flow coordination. Gaia and One.World adopt system level abstraction. In Gaia, *Active Space* is used as an abstraction of the smart physical environment, whereas, in One.World, *environment* is an abstraction which incorporates data abstractions, called *tuple*, and *functions* as a container of user applications.

Generally speaking, node level abstraction facilitates the development of applications with more flexibility and energy saving, and less communication and interpretation overheads. On the other hand, system level abstraction is easier to use, because nodal behaviors can be generated automatically, so the programmer can concentrate on the network-level actions, without worrying about how the sensor nodes collaborate with each other to perform the assigned tasks.

The second aspect in developing programming abstractions is *programming paradigm*, which refers to the model of programming the applications. It is often dependent on the applications as well as the system architecture. PvC applications

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

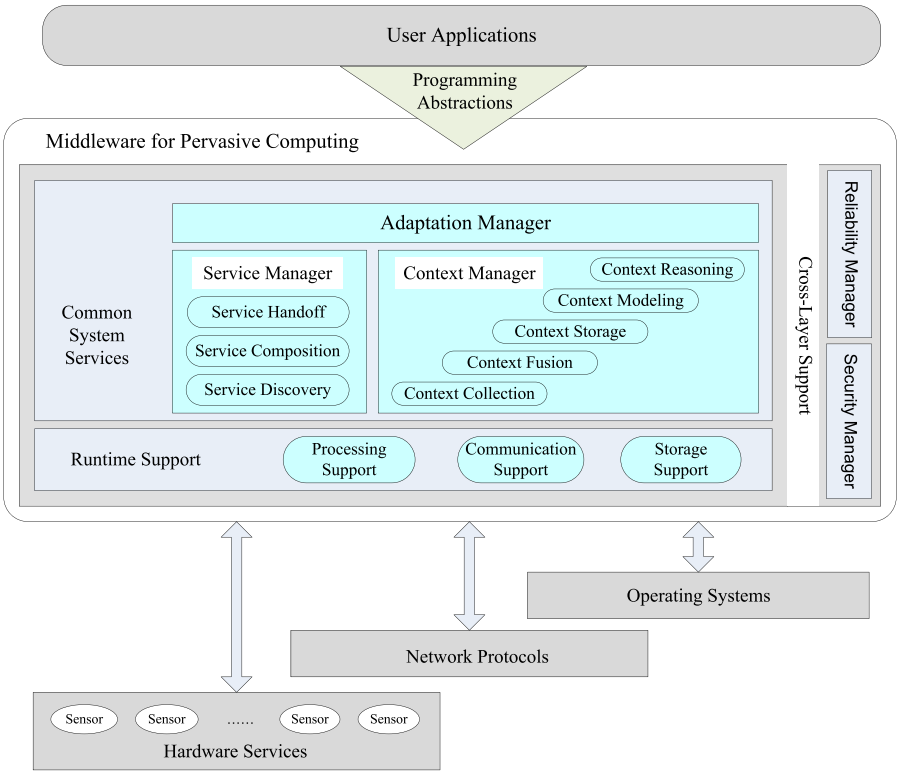

Fig. 2. Reference model for PvC middleware.

Table 1
Comparison of programming abstractions of PvC middleware systems.

		Gaia	Aura	SeSCo	One.World	CORTEX	AoC
Abstraction level	Node level	N	Y	Y	N	Y	Y
	System level	Y	N	N	Y	N	N
Programming model	Component-based	Y	Y	N	N	N	Y
	Context-based	N	N	Y	Y	N	N
	Decentralized	N	N	Y	N	Y	N
API		Active space programming interface	Aura task API	Software agent-device interface	One.World library	Event comm. and TCB model	N/A

can adopt any of the following three programming models: *context-based*, *component-based*, or *decentralized interaction-based*. For the context-based model, event triggering takes place by context change. Context-based models are very much suitable for the PvC environment, as the system can be adapted to context changes in a dynamic manner. For the component-based model, an application is composed of several component modules. And for the decentralized interaction-based model, multiple programmable smart entities interact using some rules.

Gaia, Aura and AoC use a component-based programming model. The Gaia application infrastructure is composed of distributed components organized similarly to the traditional MVC model. In Aura, each user's personal aura is composed of *task manager*, *environment manager*, and *context observer* components. Also, user tasks are represented by a coalition of abstract services. Similar to Aura, activities in AoC are composed of different service components. PICO/SeSCo uses both context-based and decentralized programming models. Mission-oriented community formation is triggered by events generated through context changes. Distributed coordination among the community members is observed similar to autonomous multi-agent interactions. One.World also uses a context-based model, where event-triggering takes place by context change, and the context change is made evident to the application. CORTEX uses a decentralized programming model with autonomous decentralized collaboration among sentient objects (see Table 1).

The third and final aspect in developing programming abstractions is *interface type*, which refers to the style of the application programming interface (API). As a matter of fact, programming abstraction is embodied as the programming interface. PvC applications require different programming interfaces based on the underlying system architecture and functionalities. Gaia has a standard programming interface of the active space model which enables developers to program the active space as a single entity. Aura has separate interfaces for *suppliers* and *connectors*. PICO/SeSCo provides resource-

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

service abstraction through a graph model which helps existing devices to adapt. One.World has a common single API for service discovery and communication. They also provide One.World library for programmers. CORTEX provides APIs for event communication as well as API for timely-computing based (TCB) model.

The consideration of adopting a particular abstraction level, and selecting an appropriate programming paradigm and applicable interface, depends on the specific application requirements and the underlying PvC environment infrastructure. Different middleware systems providing similar paradigms may share the implementation techniques as well.

2.3. System services and run-time supports

System services embody the functionalities and form the core of PvC-middleware. They are exposed to the application programmer through the abstraction interface, and provide support for application deployment and execution, as well as devices and network management. System services are the basic services shared by all PvC applications. They help manage application information and PvC environment infrastructure. The functionalities provided by the common services include:

- Context management: responsible for contextual data acquisition, processing, and derivation of higher-level contexts, context dissemination, context inconsistency detection and resolution;
- Service management: responsible for service discovery, service composition, and service handoff in PvC environment;
- Reliability and security management: responsible for ensuring correct functioning of the system through several hardware and software related faults and also ensures protection of sensitive user information.

In Section 3, we shall explain context and service management functions along with the related reliability and security issues associated with them.

Various runtime supports are necessary for the underlying execution environment while running PvC applications, and they can be seen as an extension of the embedded operating system, which provides functions of scheduling of tasks, inter-process communication (IPC), memory control, and power control (in terms of, voltage scaling and component activation and inactivation). The need for runtime support in PvC middleware comes from the fact that the hardware and firmware of the sensor nodes or other small embedded devices may not always provide enough support for the implementation of the middleware services described above. Runtime support in PvC middleware includes support for local processing, communication, energy management, and storage. More specifically, the support is provided for multi-thread processing, smart task scheduling, and synchronization of memory access.

2.4. System architecture

PvC middleware architectures adopt either top–down or bottom–up design choices depending on the system model and application requirements.

Top–down approach is a software design technique, which describes the system functionality at a very high level, and then partitions it repeatedly into more detailed levels, gradually refining the design at each step, until the detail is sufficient to allow coding. Bottom–up design approach is the reverse, where the design process starts with several small parts and gradually composes them to build the whole system. Many existing PvC middleware, such as Gaia, Aura, One.World, CORTEX, and AoC, are built in a top–down model. PICO/SeSCo middleware supports both top–down and bottom up approaches.

System architectures mostly focus on two important aspects—*mode of system control* and *mode of interaction among system components*. Mode of system control can be either *centralized* or *decentralized*. For centralized control, there will be some central component which controls the rest of the PvC devices and makes decisions on their behalf. In case of centralized systems, certain functions in the central entity are responsible to free resources in other entities. Centralized systems have a single point of failure. In case of dynamic PvC environment, maintaining data in the central entity is quite costly. For decentralized control, no single device takes any final decision, instead, various devices are supposed to work collaboratively in order to reach a global decision.

While Gaia, Aura, One.World and AoC assume a centralized control, CORTEX chooses the decentralized control model. PICO/SeSCo, however, uses a hybrid approach. In Gaia, the Gaia kernel is the management and deployment system for Gaia components. Aura task manager coordinates the task migration, monitors quality of service (QoS), and adapts user tasks. AoC has an Environment Manager (EM) to manage abstract models of the environment, and a Task Manager (TM) to capture the knowledge about user needs and preferences for each activity. CORTEX allows interaction of a very large number of autonomous components in a wired environment. System control in PICO/SeSCo is hierarchical in nature, in which resource rich devices perform management functionalities for resource poor devices.

Mode of interaction among system components can be facilitated using any available communication primitives. The usual choices made by existing PvC middleware are message passing [41], tuple space [42], and publish/subscribe (pub/sub) [43,44]. Message passing is a direct interaction paradigm in which communication is made by the sending of messages to recipients. Tuple space and publish/subscribe are indirect interaction paradigms. In tuple space, entities communicate with each other through the tuple space (see Table 2).

Aura, CORTEX and AoC use the pub/sub approach. In Aura, context observers report relevant context to the Task Manager and the Environment Manager. The CORTEX environment surrounding the application may act either as a producer or as a consumer of information while interacting with smart components. State or state changes of the environment are

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

Table 2
Comparison of system architecture of PvC middleware systems.

	Structure design	Behavior	
		Control	Interaction
Gaia	Top–down	Centralized	Message passing, Pub/sub
Aura	Top–down	Centralized	Pub/sub
PICO/SeSCo	Top–down and Bottom up	Hierarchical	Message passing
One.World	Top–down	Centralized	Tuple space
CORTEX	Top–down	Decentralized	Pub/sub
AoC	Top–down	Centralized	Pub/sub

considered as events. They are captured by sensors and disseminated to interested sentient objects in the system. Gaia system components use both pub/sub and message passing approaches. Events are used to notify entities in the active space about added or removed resources, error conditions, file system changes, and application state changes. Message passing-based interaction is used for communication among components using CORBA [45]. It is possible to port Gaia to other communication mechanisms including SOAP [46], RMI [47], or customized implementations. PICO/SeSCo uses message passing-based communication among software agents, called delegents. One.World uses a tuple space-based approach, where events serve to explicitly notify applications of changes in their runtime context.

2.5. Reliability and security supports

PvC applications are generally human-centric (elderly or child care, health care, etc.) and aim to provide people with unfaltering service support all the time. Depending on the nature of the PvC application, the sensitivity of the reliability issue may vary. Consider cases of health-care or elderly care, which may even lead to loss of lives if the system fails to detect the fall of an old person or the sudden deterioration of a patient's condition, and thereby refrains from informing the doctor. The same goes for structural health monitoring and traffic accident detection applications. So, reliability or availability issues cannot be compromised when human safety is concerned.

Security is another major concern of PvC, where multiple users interact with each other over the network. In different PvC applications, such as service discovery, users express their preferences and the devices owned by different users interact with each other while discovering services. So, there is always a high chance of giving away user's personal information. Also, many times, it is required to deduce user intent based on available contextual information, which is also an equally vulnerable security issue and needs to be addressed during system design.

Reliability and security in the PvC environment can be considered as cross-layer supports (Fig. 2) that span through the different layers of the PvC middleware from bottom to top. In this paper, we shall discuss the reliability and security issues associated with context and service management functions, as they are the core system services and have the maximum effect on the system operations.

3. Pervasive computing middleware services

In this section, we shall describe the core services provided by PvC middleware systems, in general. Section 3.1 will discuss context management service in detail, and Section 3.2 will discuss service management service. In Section 3.3, we shall elaborate the reliability and security supports available for context and service management functions in existing PvC middleware systems.

3.1. Context management service

Context has been defined [48] as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Contexts can be separated into several categories, such as: physical contexts, such as noise, light and temperature; computing contexts, such as network condition, CPU, memory, etc.; and user contexts, such as location, activity, status, social relationships, etc.

Research in context-awareness can be classified [49] into the following four primary areas—*context acquisition*, *context storage*, *context modeling*, and *context reasoning*. Context acquisition is the process of collecting raw context data from various context sources. Raw context data can be collected either from a single sensor node, or by aggregating context data available from multiple sensor nodes. The collected context data then need to be stored for further processing. The raw context tends to be noisy and inconsistent, which calls for proper context pre-processing, inconsistency detection and resolution mechanisms. After processing, context is represented using particular pattern or design descriptions, called context models. Context modeling is about constructing high-level abstraction of contextual data and building relationships among them. Context reasoning is the process of deriving high-level contexts from elementary raw context data and also about deriving implicit contexts from explicit contexts.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

In the elderly care PvC application scenario described in Section 1, we found the use of multiple sensor nodes to collect user context information for making situation-aware decisions. Several wearable sensors, such as heart and pulse rate monitors and blood pressure sensors periodically collect data about the user's physical condition and process it to detect possible abnormalities. Collected data are stored for a future study of the user's long-term physical condition. The wearable fall detector combines inputs from more than one sensor. The first is the impact sensor, which detects an impact over a threshold value which may have been caused by the user falling down, and the second is a posture sensor, which detects a reclined posture of the user which remains unchanged over a time frame. After detecting these contexts correctly, the two sensors logically deduce that the user might have fallen down and lost consciousness. So, the usefulness of context acquisition and context reasoning are crucial in the functioning of PvC applications. Below, we shall briefly describe various existing techniques of context acquisition, modeling, and reasoning.

3.1.1. Context acquisition

Context acquisition, which is a prerequisite for context-aware applications, refers to a process of gathering contexts from various context sources. Contexts can be captured through sensing devices (e.g., sensors, actuators and agents) and virtual sources (e.g., inference and prediction). *Physical contexts* are collected using various physical sensors, such as temperature, heat, magnetism, and pressure sensors, or chemical, mechanical, sound, and light sensors. Also there are motion, flow (for liquid and gas), and orientation sensors. *Computing contexts* are collected through operating system primitives (Iostat, Vmstat, Netstat) and user-level modules (SNMP, Odyssey, Congestion Manager), and *User contexts* are captured via computer vision, signature matching, etc. Though different types of contextual information are necessary for different applications, the most used context in PvC applications is location.

Context acquisition.

In the sensor layer, there are two primary methods to gather contexts: *event-driven* and *query-based*. The event-based method allows applications to specify their interest in certain state changes of the data, called an event. Upon detecting such an event, the middleware sends an event notification to applications interested in that event. This method uses asynchronous communication, and is suitable for resource constrained PvC environments. This method is applied widely in TinyDB [50], DSWare [51], Mires [52] and Impala [53]. In the elderly care scenario, the context acquisition for user fall detection is considered as an event-based one, where a user's fall event detection process is kicked off by an impact (Mr. Smith hitting the floor) trigger.

For the query-based method, applications send queries to sink nodes asking for certain kinds of contexts and the sink nodes then collect and aggregate sensor readings to satisfy the query. The query-based method allows database-style query operations to be executed among sensor nodes and sink nodes through a declarative, SQL-like but distributed interface [54]. This method is used by projects such as TinyDB [50], Cougar [55], SensorWare [56] and CACQ [27,57].

Context fusion.

Raw context data collected by multiple sensor nodes are eventually transmitted to the base station for processing. Since sensor nodes are energy constrained, it is inefficient for all the sensors to transmit the data directly to the base station. Data generated from neighboring sensors is often redundant and highly correlated. In addition, the amount of data generated in large sensor networks is usually enormous for the base station to process. Hence, we need methods for combining data into high-quality information at the sensors or intermediate nodes which can reduce the number of packets transmitted to the base station, resulting in conservation of energy and bandwidth. This can be accomplished by data aggregation or fusion. Data aggregation [58] is the process of aggregating data from multiple sensors to eliminate redundant transmission and provide fused information to the base station in an energy-efficient manner while reducing latency. Data aggregation usually involves the fusion of data from multiple sensors at intermediate nodes and transmission of the aggregated data to the base station (sink).

Data aggregation in sensor networks has been investigated extensively, and there exist several well-written survey papers [58,59]. PvC middleware can adopt prevailing data aggregation techniques used in WSNs. Data aggregation often requires resolving inconsistency between data from multiple sources and also deriving multi-dimensional relationships between different data items. Inconsistency between data items can be detected using several probabilistic and logic-based methods, such as Bayesian Networks [60,61], First-order Logic [62], Ontological [63,64], and Dempster–Shafer theory-based techniques [65,66].

3.1.2. Contextual data storage

Past research on mobile data management was based on the client/proxy/server model, where mobile/wireless devices acted as consumers of data stored on servers typically residing on the wired network. However, mobile devices in PvC environments consider not only centrally placed data sources, but even the other mobile peers which are connected in an ad hoc manner and act simultaneously as producers and consumers of data. Data management in these highly dynamic setting [67] must consider issues such as frequent change of data and data source based on location and time, lack of a global data catalogue, frequent disconnection among peers, and data synchronization and coherence problems. In this section, we discuss some of the aforementioned issues which are related to contextual data management in PvC environments. Storing raw context data collected from numerous sensor nodes is a non-trivial issue in designing any PvC application. Existing data storage techniques developed for WSN cannot be directly applied for data management in PvC due to the following reasons.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

Firstly, data in PvC comes from multiple sources (e.g., different types of sensors and other smart entities), and to store and maintain such a vast amount of heterogeneous data is rather difficult. Since, there can be multiple sources of same data, resolving redundancy and inconsistency is often necessary. Different data sources may use different data formats and/or media formats. Also contextual data is often spatially distributed, and hence, deciding where to store contextual data (people, space, and objects) is important. Different policies for maintaining privacy and security may be required while collecting context data at different sources.

Secondly, since the data values in PvC change frequently, the relationships between different data items also require updates. This is especially challenging as data items can share multiple contextual relations with other data items in terms of spatial, temporal, social, and logical contexts. Therefore, in order to achieve low update delay, it may be necessary to store related data items at physically close locations. Users of PvC applications often want to navigate through the high-level relationships among the data items. Moreover, change in data values may start cascading contextual changes in other smart objects if they are contextually connected.

Finally, the data sources and consumers are extremely dynamic in PvC systems. They can join and leave at any point of time, which changes data sources randomly. Also, the user's interest in particular data can change with the context.

So, the data management system for a PvC environment must guarantee ubiquitous data access for users. The user must have a seamless data access experience and the QoS of data access should be adequate in terms of low latency, high availability and correctness and relevance. Also, the system should understand the user's intention and provide the required data proactively.

Data storage in a PvC environment must take care of two issues—*managing large volumes of data* and *managing contextual relations between data objects*. A large volume of data can be managed by distributed and hierarchical storage systems. Hierarchical system organization is an efficient tri-stage data processing technique, where tags, sensors, and actuators are attached to physical objects that are at the lowest level. The sensory devices contain a textual description of the physical object and perform specified functions and additional actions. The intermediate level contains multiple static or mobile sub-stations associated with a physical location. They collect and maintain data from nearby objects and possess larger storage capacity than tags. Sub-stations forward aggregate object information to the base-Station, which has large storage and processing capacity, and is usually connected to a constant power source.

Distributed and hierarchical storage systems often support massive data storage and can cope with frequent changes of data. They allow P2P search and query. One such example is Context Distributed Database Management System (CDDBMS) [68], where for each piece of context a master copy resides at a central point of access (home node), and the replicas roam and operate in a remote mobile node. The mobile object handling mechanism is based on a chain of pointers from the home node to the remote mobile node. Another example is Spacelog [69], which enables contextual data storage for smart spaces such as laboratories, classrooms, libraries, homes, clinics, shops, restaurants, construction sites, etc., and can be regarded as a special database for a physical environment. The context data collected can be about people, facilities, artifacts and spatial contexts such as temperature distribution, air quality, noise level, sound source, etc.

P2P Context Storage Technique [70] is another technique where multiple context producers, known as ContextPeers, are interconnected into self-organized semantic P2P network. Peers with semantically similar contexts are grouped together. Context producers ('*ContextPeers*') produce contexts and store context data as RDF triplets. Context-query is parsed using the RDQL (RDF Data Query Language)—based query engine. The SCS protocol undertakes overlay construction and maintenance, and query routing.

3.1.3. Context modeling

Context model is a pattern or description to represent context. After context acquisition has been done, context modeling and representation is necessary to validate the integrity, structure and data of the collected context information. Efficient context models can support—querying current and historical contextual information, deriving high-level contexts from raw context data—and are easy to extend, so that new context types can be added without affecting existing information. Many context models have been developed over the past decade, ranging from simple to sophisticated complex models. Based on the data structure used for expressing and exchanging contextual information, papers [28–30,49] have summarized context models that have been proposed earlier. Here we introduce the recent advances in context modeling.

(1) *Key-value pair context model*: This is the simplest type of context modeling technique, which uses key-value pairs to represent attributes and corresponding values describing contextual information, e.g., <temperature, 25 °C>. It has achieved widespread success in early distributed service frameworks to represent location information or the capabilities of services, e.g., in [71,72]. Then, service discovery is performed by matching key attributes. Though tuple-based techniques are easy to implement and manage, they are inefficient for complex and sophisticated context representation and support only simple context queries based on text string matching. Also they are not very well suited for context reasoning.

(2) *Logic-based context model*: A variety of logic-based context models [73–76] have been proposed by researchers. Cabot middleware project [77–79], uses a first-order logic-based context model which is designed for checking and resolving context inconsistency. Many PvC middleware (e.g., Gaia) use context model based on first-order logic and Boolean algebra. In this form, context is usually represented as (<context type>, <subject>, <predicate>, <object>). E.g., (Temperature, Room A, is, 25 °C), or more complex constructs like the following one, as used in GAIA—Context (Number

of people, Room 2401, >, 4) AND Context (Application, PowerPoint, is, Running) ⇒ Context (Social Activity, Room 2401, Is, Presentation).

(3) *Object-oriented context model*: Object-oriented techniques for context representation make use of the merits of the object-oriented approach (e.g., encapsulation, inheritance, and reusability) to represent contexts and can be accessed and modified through *accessor* and *mutator* methods. Several representative context models for the object-oriented technique include, cue [80] at the TEA project [81], and Active Object Model at the GUIDE project [82]. However, object-oriented context models are difficult for developers as they must systematically analyze and design the entire context-aware system. In addition, it is nontrivial to update and optimize the context-aware system as time goes on.

(4) *Markup context model*: The markup model represents context using a hierarchical data structure consisting of markup tags with attributes and content. The tags can be interpreted by devices to control how contexts should be organized and exchanged among devices. The two most popular markup context models are User Agent Profile (UAProf) standard [83], which captures capabilities and preferences for wireless devices, and Composite Capabilities Preferences Profile (CCPP), which describes device capabilities and user preferences and is often used to instruct the adaptation of content presented to devices [84]. However, they are limited by their pre-defined hierarchy and restricted overriding mechanisms. Some other markup context models are Context Extension [85], Comprehensive Structured Context Profiles (CSCP) [86], Pervasive Profile Description Language based on XML [87], ConteXtML [88], Centaurs Capability Markup Language [89], and the note-tags of the stick-e notes [90]. However, markup context models suffer from several problems in capturing various types of contexts, capturing relationships, dependencies, timeliness and QoS of contexts, inconsistency checking, context reasoning, and removing uncertainty from contexts [71,91].

(5) *Ontology-based context model*: The ontology-based context model makes use of Resource Description Framework (RDF) and Ontologies recommended by W3C, which incorporate semantics into an XML-based representation. A variety of RDF-based and ontology-based context models have been proposed [92–98]. Ontologies and RDF are promising techniques to model contexts because of their formal expressiveness and the ability to infer contexts. However, they often represent incomplete ontologies (i.e., construct deficit) and ambiguity of ontologies for given contexts [99]. They are also seriously limited by inexact reasoning [100].

In the elderly care scenario proposed earlier, the acquired context information can be modeled as simple key-value pair or by using first order logic. While simple context elements, such as the body temperature, blood pressure, and heart rate of Mr. Smith can be modeled using key value pairs, the fall detection requires more advanced logic-based context modeling. The fall detection logic can be represented as (Mr. Smith, has, Physical impact > Threshold) AND (Mr. Smith, Posture: reclined, >, T time) ⇒ (Mr. Smith, state, fallen and unconscious). In the following sub-section we shall study the existing techniques of context reasoning.

3.1.4. Context reasoning

Context reasoning refers to a process of inferring high-level implicit context information based on the existing low-level explicit context information. Existing context reasoning techniques can be classified into two types—*exact reasoning* and *inexact* or *fuzzy reasoning*. Exact reasoning techniques incorporate Bayesian network based [60,61], logic-based [62], case-based [101,102], ontology-based [95,103], and rule-based [104] techniques, whereas, inexact reasoning techniques include evidence-based [100,105], and fuzzy-based [106,107] techniques.

(1) *Exact context reasoning: Bayesian networks* can be regarded as a canonical reasoning technique, which employs probability graphs to represent contexts. Though it can be easily tailored to different context models, yet, it is seriously limited by its exponential computation overhead, and requirement of exhaustive and exclusive hypotheses. *Case-based* reasoning infers contexts based on the past cases. However, it cannot accurately measure the similarities among cases. *Logic-based* exact context reasoning is inefficient for incomplete, imprecise, and fuzzy contexts, often available in PvC environments. Similarly, *Rule-based* reasoning infers contexts using pre-defined rules. An open issue in the rule-based reasoning technique is dynamic generation of rules depending on varying contexts. *Ontology-based* reasoning schemes, backed by powerful software supports [63,64], incorporate semantics into context representation and reasoning, and are quite popular. They implicitly assume that all ontologies related to a specific domain are pre-defined. However, this supposition does not always hold in PvC environments.

(2) *Inexact context reasoning: evidence theory*, also known as Dempster–Shafer theory [65,66], is used for inexact context reasoning. Evidence theory is used to construct ground truth from anecdotal evidence and to provide inaccurate predictions. This technique incurs heavy computation overhead and is inefficient in representing context semantics. Another technique to deal with imprecise and incomplete contexts is *fuzzy context reasoning*, which employs fuzzy set theory. This technique provides a manner of combining captured data with expert knowledge. In [108], a fuzzy reasoning scheme based on Mamdani inference systems is proposed, and it assists mobile applications in fuzzy context reasoning. Compared with exact reasoning, inexact reasoning schemes cannot get the accurate implicit contexts and thus they may be unsuitable for PvC applications which require accurate contexts.

In the elderly care scenario depicted in Section 1, context reasoning can be used to deduce the fall event of Mr. Smith, and to carry out the required operations regarding his treatment and hospitalization. Once all the context information is available, the application can use pre-defined rule-based reasoning to decide on the course of actions. E.g., (IF Mr. Smith fell

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

on the floor) AND (IF Mr. Smith is unconscious) AND (IF Mr. Smith has abnormal heart rate) THEN inform the physician about Mr. Smith's condition. After the physician has been contacted, he may require deductive reasoning to conclude if Mr. Smith had his lunch and whether he took his medicine before he fell down. Also, he may need to know if Mr. Smith is bleeding due to the fall. The application requires discovering simple or composite services specific to those needs. A dedicated service management module in a PvC middleware can take care of these issues. In the following section, we describe the functions of such service management modules.

3.2. Service management service

PvC applications are mainly concerned about services, instead of the individual nodes that provide them. Services can come from a single node or can be composed of a set of nodes. Due to their service-based nature, pervasive systems require strong supporting mechanisms for service management. Service management comprises three main operations—*service discovery*, *service composition*, and *service access*, and in the following paragraph we shall define them according to our understanding.

Services are central to the service management techniques, and they can be defined as *any hardware or software functionality (resources, data or computation) of a device that can be requested by other devices for usage*. For example, the media player on a mobile device can be considered as a service which provides music and movie playing functionalities for that device, and can be requested by peer devices. Service discovery is *a process by which any potential user (human or device) requiring a service, can find services on peer devices, and can determine how to access or utilize the discovered services*. Service composition, on the other hand, is *a process of identifying and combining component services to compose a higher-level service*. Service access ensures that users requesting particular services can keep using them despite user or service provider mobility, device failure, or network disconnection.

3.2.1. Service discovery

Service discovery enables devices and services to properly discover, configure, and communicate with each other with minimal or no human intervention. However, service discovery has been investigated in traditional distributed as well as enterprise environments. Service discovery in enterprise networks is restricted, as they consider mostly static and resource-rich computing devices connected through wired or infrastructure-based networks. Moreover, services in enterprise networks operate within a fixed scope, and hence, they can be protected by firewalls and can be managed by system administrators on a centralized basis.

PvC environments, on the other hand, are far more dynamic and heterogeneous. When combined with a purely ad hoc wireless network structure, the chances of occurrences of faults increase greatly. Moreover, due to their ad hoc nature compounded with node mobility, it is not possible to centrally control PvC systems through system administrators. Devices must act through localized coordination in order to discover services and use them reliably. As services in pervasive networks are scattered in the ambience, network-wide service discovery poses a significant challenge. However, the most important challenge regarding service discovery in the PvC environment lies in figuring out how to satisfy human users seamlessly. Integrating people in smart environments requires robust security and reliability supports for service discovery applications.

Several well-organized survey papers [31–33] on service discovery systems in mobile and PvC environments exist in the literature. Prevailing service discovery protocols (SDP) are usually classified based on their *underlying network types*, *discovery models*, and *discovery processes* (see Fig. 3). Based on the network structure, PvC environments can be either *wired* or *wireless*.

Service discovery protocols employ either a *directory-based* or a *directory-less* discovery model. For the directory-based model, there is a dedicated directory node along with the service providers and service consumers. The directory node maintains service information and processes queries and announcements. Some directories provide additional functionality. For instance, Ninja SDS [109] directories support secure announcements and queries. Directory-based systems are usually more efficient and scalable than directory-less ones. Based on the number of services and the size of the network, directory-based systems can use a single centralized directory or multiple directory nodes distributed across the network at strategically important locations. Depending on the organization of the directory nodes, directory-based models can be distinguished as either *flat* or *hierarchical*. In a flat directory structure, subdirectories have peer-to-peer relationships. For example, within an INS [110] sub-domain, directories have a mesh structure: a directory exchanges information with all other directories. Salutation [111] and Jini [112] can also adopt flat structure. On the other hand, a hierarchical directory structure follows the DNS model in which information, advertisements, and queries are propagated up and down through the hierarchy. Parent nodes store information of their children, and thus in this type of system, the root node may possibly become a bottleneck. Examples include Ninja SDS [109] and Rendezvous [113], both of which have a tree-like hierarchy of directories.

Discovery processes are of two types—*active/pull-based/query-based* discovery and *passive/lazy/push-based/announcement-based* discovery. In the query-based approach, a party receives an immediate response to a query and does not need to process unrelated announcements. In the announcement-based approach, interested parties listen on a channel for periodic announcement from service providers. Many protocols support both approaches.

Based on the nature of the underlying network environment, service discovery protocols adopt different designs. Wired networks consist of resource-rich and static computing devices connected through high-bandwidth network cables.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

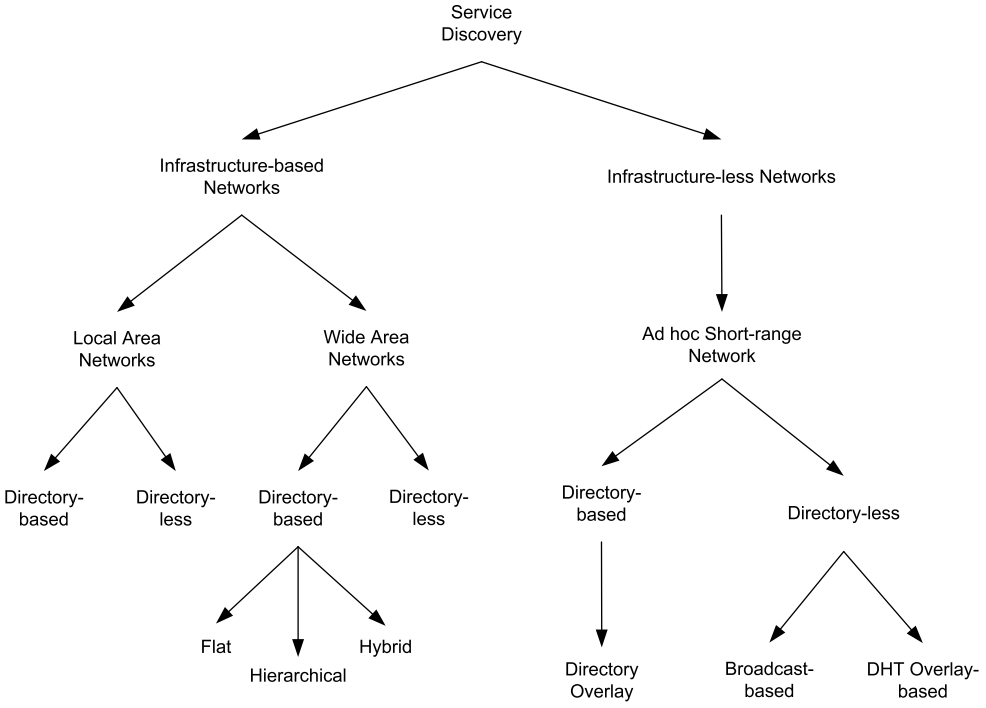

Fig. 3. Classification of existing service discovery protocols.

Examples of these types of systems are enterprise networks. Service discovery protocols for wired and infrastructure-based environments [109,112,114–120] are usually unsuitable for usage in PvC environments. Wireless networks, on the other hand, can be employed either in static or in mobile settings. Static wireless networks mostly contain high bandwidth network backbone infrastructure, which supports the networking needs of the participating computing entities. Mobile wireless networks, however, lack any such infrastructure support. They create an ad hoc composition of multiple resource-constrained portable and handheld devices connected by unreliable and intermittent wireless connectivity. PvC systems are characteristically closer to the ad hoc wireless environment due to their limited resource availability, extreme dynamism and unreliable network connections.

Service discovery in ad hoc wireless environments consist of multiple resource-constrained and possibly mobile devices which provide services to their peer devices. Existing service discovery protocols for ad hoc environment can be directory-based or directory-less.

In the directory-less service discovery protocols, the service information is stored with the service providers themselves. There are two distinctly identified methods for directory-less service discovery in ad hoc networks. The first method works by broadcasting service information as well as service requests and the second approach works by building DHT-based P2P overlay [121–123,109,124,125] for mobile ad hoc networks (MANET).

Broadcast-based service discovery can adopt either the push or pull model. In a push-based discovery model, service advertisements are distributed by the service providers to all the nodes in the network. A pull-based discovery model, however, necessitates a service requestor to broadcast their service request to other nodes until a matching service is found. The broadcasting nature of this model is grossly unsuitable for the mobile ad hoc networks due to their high demand of bandwidth and energy. So, these protocols can only be used in small scale networks. Some of the protocols which use broadcast policy are Bluetooth [126], DEAPspace [127], Allia [128], GSD [129], DSD [130], and Konark [131].

Among DHT overlay-based service discovery protocols, there are several MANET-oriented DHT systems [123,124,132] that integrate DHT with different ad hoc routing protocols to provide indirect routing in MANET. Ekta [123], CrossROAD [124] and MADPastry [132], each integrates Pastry [133] with DSR [134], AODV [135], and OLSR [136], respectively, to share routing information between the network layer and application layer. An opposite approach is adopted by virtual ring routing (VRR) [125], which is a network-layer routing protocol inspired by overlay routing on DHTs, and can significantly reduce traffic compared with broadcast-based schemes.

The directory-based service discovery in ad hoc networks is thwarted by the resource-poor nature of devices and their mobility, which makes it difficult to choose single centralized directory nodes. To cope with this limitation, directories are dynamically selected from mobile nodes, considering their available resources, such as processing power, memory size, battery life, or node coverage, etc. It is true that dynamic directory assignment incurs extra overhead to the network, because directories should be selected and their identities should be informed to the rest of the network nodes. Moreover, directory nodes must be constantly available on the face of node failures and dynamic topology changes and network partitions. Even

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

with these difficulties, a directory-based system proves to be more scalable and fault tolerant than a directory-less one in these environments. Moreover, using directories will most certainly decrease the discovery delay and will enhance load balancing among service providers, so as to reduce the load on individual services, and enhance the overall service discovery performance. Examples of directory-based service discovery protocols for ad hoc wireless networks are—Service Rings [137], Lanes [138], DSDP [139], Splendor [140], and some other schemes without any proposed scheme-name, like [141–143]. The basic approach followed by these protocols is the same. They select some nodes as directories based on some parameters and then form an overlay or backbone network connecting those nodes.

In this section, we described the service discovery actions performed by existing systems and protocols for different types of network environments with a special focus on those being used for PvC environments. Based on the presence or absence of a service discovery registry, we have also classified the service discovery protocols as directory-based and directory-less. Service discovery in real life is often subject to several constraints based on user requirements, and hence, a single service provider may not be able to meet all the needs of a user. In certain cases it may be necessary to compose a service by combining the elementary services provided by multiple providers. In the following sub-section we shall study the service composition functions.

3.2.2. Service composition

Basic functions of service composition include—*means to specify the preferences by users and service descriptions by service providers*, *finding service providers in the environment matching user preferences*, *selecting the best suited service*, and *combining elementary services to compose a higher-level service*.

Describing services.

Service providers describe their services in standard formats and the service requestors also need to specify their requests for composed service following standard techniques. Below we summarize different service descriptions adopted by service providers and service requestors.

Service providers often describe their services as atomic functionalities. E.g., in the elderly care example scenario described earlier, for the fall detection there are two types of atomic services—"fall sound detection" and "body orientation detection". During a service composition process, service providers may have the simple knowledge of their atomic services or the additional knowledge to decide whether it can be a part of a service composition workflow. E.g., the two atomic services for fall detection must be able to decide that they have roles in the service composition workflow—"detect the fall of Mr. Smith". Both these processes are dependent on the service description process adopted by the service provider.

Users or service requestors can specify their service request either in high level or in low level. In the high-level description [144], the requested service is specified as a goal to be achieved, e.g., the user queries the room saying "determine the cause of Mr. Smith's fall". In this case, the intelligent devices in the room have to figure out their roles and need to perform accordingly. High-level service description can use natural languages or ontology terms. In the low-level description [145], the requested service is specified as a workflow, given the set of atomic services to be composed. This is easier for the participating devices, as they know their roles directly from the workflow specified by the user.

Specifying composition plan.

Service composition system tries to provide the user requested service by composing the available services. It tries to generate one or more composition plans with the same or different services available in the environment. It is quite common to have several ways to compose a service, as the numbers of available services in PvC environment are abundant.

One critical challenge in developing a service composition plan is regarding how to describe appropriately the services provided by multiple service providers, and how to represent services requested by users. Many service modeling languages have been developed towards achieving that objective, and they can describe functional and non-functional attributes of services. Existing service composition approaches can be characterized by the different expressiveness of the modeling language and can be classified as below.

- *Low-level requested service description*—In this approach [137–139,143], the requested service is specified as a workflow, given the set of atomic services to be composed.
- *High-level requested service description*—In this approach [115,120], the requested service is specified as a goal to be achieved.
- *Workflow-provided service description*—While the majority of solutions assume that service providers have limited knowledge and can only describe atomic functionalities that they provide, in this approach [135] it is assumed that service providers are able to specify workflows in which they can take part.

In the existing literature, there are mainly three different techniques of drafting a composition plan. The first one utilizes the classical *planning* technique used in artificial intelligence (AI). In this approach [127,139,142], the composition of atomic services into composite service is viewed as planning. Atomic services are mapped into planning operators, a planning algorithm links them, and the generated plan constitutes a composite service. The second technique uses *workflow*, in which a composite service is broken down into a sequence of interactions between atomic services [137,138,143]. The system generates a customized workflow that describes how various services should interact with one another as well as with the requestor. Finally, there is the *historical data-based* composition technique [117], which uses data mining in order to discover previous service composition and usage patterns which can help future service composition.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

Selecting service providers.

It is usual to have many services providing the same or similar functionalities. In that case, the services are evaluated by their overall utilities using the information available from the non-functional attributes. Service selection is done based on user specifications. Either, all the matching services are sent to the user for manual selection [146], or the system automatically chooses the best service that matches user requirements [145,147]. Selection is usually performed based on some application context [148], quality of the network, or service QoS parameters [149]. An example of context-based service provisioning is *finding all Japanese restaurants in the area close to the user's current location*. QoS-based service selection is carried out with the help of several metrics, such as cost of usage, time of availability, etc.

Service composition architectures and techniques.

Service composition architectures can be *centralized*, *distributed*, or *hybrid*, the last one being a composition of the previous two approaches. In the centralized approach [147], there is a central coordinator assigned for performing service compositions. The coordinator can be fixed, or can be chosen at run-time, and can even be changed during the execution [150], if need arises. In the decentralized approach, no central coordinator is present and the composition is achieved by peer-to-peer interactions among the participating services. In the hybrid approach [19], distributed coordinators are used for service composition. While the centralized approach is prone to a single-point-failure, the distributed approach is quite difficult to implement in the highly heterogeneous PvC environment. So, the hybrid approach may strike a balance between these two extremes.

Siebert et al., have proposed a fully decentralized service composition approach [151], in which service composition is carried out through localized interactions between service providers. The problem of identifying a composite service graph, similar to a request graph input by the user, has been modeled as a sub-graph isomorphism problem, which is proven to be NP-complete. Unlike previous algorithms that rely on global knowledge, in this algorithm each device only maintains local state information about its physical neighbors and builds an overlay network. Services reside on the device and component services construct sections of the composition graph by interacting with neighbors. Finally, all such fragmented sections are interconnected to come up with the requested composite service.

Service composition techniques can be either static or dynamic, based on whether the solution provides provision of dynamically changing the composition plan in the middle of an ongoing service composition process. Even after a composition has been successfully carried out and the user is accessing the composed service it may be required to replace a component service due to failure or unavailability caused by some other reasons. If the service composition solution can perform this replacement automatically, and can add or remove component services without halting the ongoing composition or service execution process, then the composition technique is dynamic, otherwise it is static.

Service discovery and composition are crucial services required to be provided by a PvC middleware. In the elderly care application scenario introduced in Section 1 of this paper, we describe an example service composition operation, which requires the application to discover whether Mr. Smith had his medications before his fall. This high-level service is actually composed of three elementary services, such as, (1) detect whether Mr. Smith opened the medical cabinet before his fall (by video footage), (2) detect whether the medicine has decreased by the daily usual amount (by level counter), and (3) (if possible) detect whether Mr. Smith has actually taken the medicines (by video footage). The results of these elementary services can then be composed to deliver a higher-level service.

3.3. Reliability and security support

In this sub-section, we shall discuss fault tolerance and security support available for the context and service management modules. Faults in PvC environments can be of two types—*infrastructure-related* and *software and service-related*. While the *infrastructure-related* faults cover hardware failures, such as devices and network failures, *software and service-related* faults are concerned with failures of pervasive software and system services.

Privacy and security issues have some conflicts with the pervasiveness property of a smart application. Sometimes PvC applications require user's location for providing location-based services, such as being discovered by friends in the neighbourhood. However, divulging user's location may have severe privacy concerns. Similarly, in the elderly-care scenario depicted in this paper, when Mr. Smith is unwell, his crucial health information will have to be shared with multiple medical care-givers, and it may not be possible to verify their identity all the time. So, there is a chance that secured personal data is leaked to unauthorized third parties during the chaos. We can see that ensuring privacy and security of sensitive user information often requires striking a balance with the application requirements.

Below, in Section 3.3.1, we shall briefly describe various fault tolerance techniques in context management, in order to ensure quality of context information. Section 3.3.2 discusses techniques to ensure security and privacy of crucial context data. Similarly, fault tolerance as well as security related issues of service management in PvC have been discussed in Sections 3.3.3 and 3.3.4, respectively. Finally we shall compare several PvC middleware services with respect to a set of middleware solutions.

3.3.1. Fault tolerance in context management

The context management module is considered as the brain of a PvC system and any type of failure in this module may severely disrupt the application functionalities. Usually failures in context management services are mostly due to incorrect

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

context sensing and reasoning. One way to improve the context management service operations is to introduce context pre-processing as well as detecting and resolving context inconsistencies.

Context preprocessing.

Contexts are often noisy and incomplete in PvC environments [152,153] due to incompetent sensing technology. This produces errors in collected data and their interpretation as contexts. For example, RFID readers just capture 60%–70% of the tags in their vicinity [154] and the rest goes undetected. Context preprocessing is a function that processes raw contextual information before context reasoning and interpretation. This function, if implemented, can significantly improve context quality, particularly when collected contextual information is too coarse-grained.

Context preprocessing on the sensor level is performed either in a *centralized*, *distributed*, or *hybrid* manner. In the centralized manner, a central node or a centralized module is required to preprocess contexts captured and delivered by sensing devices. In the distributed manner, sensors and the aggregating nodes have to filter and preprocess their collected context by themselves. The hybrid design combines the strengths of the former two, in which several nodes are selected to aggregate their nearby contexts and then combine their results and report them to applications or users. Various schemes [27,155–158] have been proposed to preprocess raw context in the sensor layer, using Bayesian network, Kalman filter, linear regression, and other statistical and probabilistic techniques.

In [157], Extensible Sensor stream Processing (ESP) is a data cleaning infrastructure between sensor devices and applications. The cleaning infrastructure translates raw sensor data into clean data, so that applications are unaffected by erroneous data. ESP consists of a programmable pipeline of cleaning stages intended to operate on-the-fly as sensor data are streamed through the system. ESP is designed to be easy to configure and be able to evolve over time.

Context preprocessing in the context level [159,160] has not been studied well. Most of the existing works on context-aware research implicitly assumes that raw data is accurately interpreted, which is often not true [161].

Context inconsistency detection and resolution.

Context inconsistency detection and resolution is often necessary due to the noisy context data collected at the sensor layer. Various schemes [77–79,162–168] have been proposed towards that objective.

(1) *Context inconsistency detection*: In [78], a context-aware middleware modeled context constraints by tuples, and checked context consistency by semantic matching and inconsistency triggers among elements in tuples. This work was extended in [79], which proposed ICIA—an Incremental Checking Inconsistency Algorithm, which works using a consistency computation tree (CCT). In [162,163], ontologies and assertions are used to model contexts and context consistency, respectively. However, ontology-based schemes have limited capability of removing noise in contexts and they require that all ontologies related to a specific domain be pre-defined, which is not always possible for common users. In [168], CEDA evaluates the temporal relationships among collected contexts in asynchronous environments, and checks for inconsistency among them using *happen-before* relationships.
(2) *Context inconsistency resolution*: In [169], a scheme is proposed to resolve inconsistency by following user preferences or polices in situation evaluations. In [77], a "drop-bad" heuristics-based context inconsistency resolution strategy has been introduced, which overcomes the shortcomings from "drop-latest" [170] and "drop-all" [171] resolution strategies proposed earlier, and thus achieves more reliable results. This strategy keeps track of contextual information and gets rid of outliers using historical information. In addition, [172] proposes a Quality-of-Context (QoC) based solution for removing context inconsistency and conflict resolution by incorporating quality, which refers to any information describing contextual information and its worth for a specific application.

3.3.2. Security and privacy in context management

Maintaining privacy is necessary in order to protect user's personal information. Conventional mechanisms to ensure security in distributed systems may not be suitable for PvC environments due to extreme dynamism and device resource constraints. Privacy guarantees how contextual information will be used or passed on. Since users expose plenty of personal information in a PvC environment, as a result, maintaining absolute privacy is difficult. Security and privacy in context management has not been thoroughly studied, except for location privacy research which also remains largely unexplored. Among the existing approaches there are access control, encryption and cryptographic techniques.

(1) *Access control*: In order to achieve security and privacy, Access Control (AC) is adopted in the majority of projects, such as Gaia [13] and Context Toolkit [48]. There are several well-recognized AC models—Attributed-Based AC (ABAC), Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC), among which ABAC and RBAC are two most popular ACs.
(2) *Encryption and cryptography*: Conventional cryptographic and encryption methods are inappropriate for PvC environments, because of the extensive computation overhead incurred by the existing encryption algorithms. Pseudonymity [173] could be an important solution, using which users can frequently change pseudonyms in order to avoid being identified by the locations they visit. This can solve the privacy problem discussed in the second paragraph of Section 3.3, where users do not want to divulge their location while willing to use several location-dependent services. Another solution is anonymity, and can work well to hide the user's identity in some of the location-based services. E.g., if the user wants to be discovered by his friends when he is in a shopping mall, he must use his identity, however, if the same user wants to be notified about the special pizza price while he is passing through a pizza place, then identity exposure is not necessary.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

3.3.3. Fault tolerance in service management

Reliability and availability of service management systems are affected, either by hardware failure, or due to user or service provider mobility. So, ensuring reliability in service management requires those issues to be addressed. Below, we present an overview of existing fault tolerance and mobility management techniques used in wireless network-based service management solutions, as they are the most suitable ones for PvC environments.

Fault tolerance and mobility management for service discovery in wireless networks.

Fault tolerance in ad hoc wireless environments is challenging compared to traditional wired distributed environments. The dynamic nature of the environment, device resource-constraints, and unreliable wireless connections, make it hard to design robust fault tolerant mechanisms. Fault tolerance requires withstanding failure of directory nodes. INS [110] and LANES cope with directory failure by maintaining multiple copies of service information at different nodes. Most of the directory-based protocols replace failed directory nodes with newly selected ones.

Mobility management, however, is very challenging in ad hoc environments. Frequent node mobility renders the topology unstable, and disconnections give rise to inconsistency in service information. In order to maintain consistency of service and route to service information, either a proactive or a reactive method has been adopted by existing protocols. In a proactive approach, the participating nodes periodically exchange messages to update information, e.g., a service provider may send periodic advertisements to update its location. A reactive method, however, updates information based on events. For example, if a user discovers that a service previously cached by him is unreachable, then he seeks new service information.

Directory-less service discovery protocols cope with node mobility by adjusting service advertisement rate and the diameter of announcements. For example, GSD [129] implements a small advertisement time interval for highly dynamic environments, opposed to a larger value for comparatively stable networks. The advertisement diameter (in number of hops) is also regulated depending on different mobility situations. Similarly, Allia [128] controls the frequency of advertisements and the diameter of the alliance considering the mobility of nodes.

A majority of directory-based protocols for ad hoc service discovery require special mechanisms to maintain the directory structure, called backbone or overlay. These algorithms try to ensure smooth operation by handling node joining or leaving scenarios, broken connections, network partitions, and partition merges. Service Rings [137], Lanes [138], DSDP [139], Tyan and Mahmoud [141], and Sailhan and Issarny [142], all proposes similar mechanisms.

Raychoudhury et al. [174], have proposed an efficient and fault tolerant service discovery protocol for MANETs. A virtual backbone of directory nodes, called directory community, is first constructed by selecting the top K nodes considering higher resource and lower mobility, to ensure that they are more reliable and less fault-prone. They have modelled the directory community formation problem as a top-K weighted leader election in mobile ad hoc networks [175], and develop a distributed algorithm to achieve the objective. Here, the weight indicates available node resources in terms of memory, processing power or energy. Using the aforementioned directory community, a quorum-based fault-tolerant service discovery protocol has been proposed. The elected directory nodes are divided into multiple quorums. Services registered with a directory are replicated among its quorum members, so that, upon the failure of a directory, services can still be available. This approach guarantees network-wide service availability using the quorum intersection property, and reduces replication and update costs by minimizing the quorum size.

Fault tolerance and mobility management for service composition.

Given the dynamic and fault-prone nature of PvC environments, service provisioning needs to be dynamic and adaptable to unpredictable changes. As already discussed in Section 3.2.2, existing service composition techniques can either be static or dynamic. In static systems, if any of the services fail, service composition needs to start all over again. The majority of proposed service composition solutions assume static service composition. However, dynamic service composition approaches [148,176–179] support re-planning of the composed service during the execution of the composition. Services can be replaced, added, or removed if necessary without starting the service composition processes afresh. Dynamic service composition is more difficult to implement than static service integration, since every service component of the dynamic service is being monitored and should be replaced immediately in the case of failure. Dynamic adaptation to contextual changes [147] may require service re-binding or service execution state transfer as and when necessary.

3.3.4. Security and privacy in service management

Ninja Secured Service Discovery Service (SSDS) [109,180] is one of the earliest proposed methods of securing service discovery applications. SSDS uses a hierarchical directory-based approach in which a user uses his public key to authenticate with a local server in order to discover services. Service providers specify user privileges and register services with local servers. Furthermore, communications among different parties are encrypted.

Security in Bluetooth [181] and Universal Plug and Play (UPnP) [182] protocols is based on the traditional authentication and authorization approaches, where a user has to provide correct credentials to discover and access services. In the trusted discovery mode of Bluetooth Security, services only interact with a pre-known device that shares a common secret code. UPnP Security features include several authorization methods, including access control lists, authorization servers and certificates, and group definition certificates. Though these approaches protect the privacy of the service provider, they overlook the privacy of the service user.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

Table 3
Comparison of PvC middleware services.

		Gaia	Aura	PICO/SeSCo	One.World	CORTEX	AoC
Context management	Sensing	√	√			√	√
	Modeling	√	√			√	√
	Reasoning	√			√	√	
Service management	Discovery		√	√	√		√
	Composition		√	√			√
	Hand-off	√		√			
Fault tolerance	Device	√		√	√		√
	Application	√	√		√	√	√
	Network		√	√			
	Service	√		√			
Security and privacy	Authentication and access control	√			√		
	Secured service discovery	–	–	–	–	–	–

Zhu et al. [140,183–186] have carried out significant research in ensuring security and privacy of both the user and the service provider in service discovery. They have proposed Splendor [140], a secured service discovery protocol for nomadic users in public environments, which has provisions for protecting user, data, and location privacy. Location-awareness in Splendor can support location-dependent service discovery with reduced security risks. Splendor can protect users against eavesdropping and replay attacks. Splendor offers mutual authentication among components, simplifies service authorization, provides communication confidentiality and message integrity, and supports non-repudiation.

PrudentExposure [183,185], also proposed by Zhu el al., does not require users to actively identify existing services and service providers. Instead, a user's program discovers service providers from whom he acquires credentials via code words specified in the Bloom filter format [187]. This format enables users and service providers to identify each other in one round of message exchange. After identifying each other, a user and a service provider establish an encrypted channel between themselves to exchange requests and service information. The approach can protect sensitive information, such as service information, user identities, user's presence information, domain identities, service provider's presence information, and service requests.

An extension of the PrudentExposure scheme, named as *progressive approach*, was proposed in [184,186], which is similar to the former one, in which a user utilizes a program to manage all his credentials, and users and service providers exchange code words. However, progressive approach uses a different method to exchange code words and service information, and protects sensitive information by denying unnecessary exposure of the same even to the legitimate parties. Thus, when a user needs a service, he only authenticates service providers providing a matching service and excludes other known service providers. Similarly, a service provider exposes his information only after verifying the legitimacy of both the users and the service. In the progressive approach, users and service providers can establish mutual trust through multiple rounds of communication while exchanging partial and encrypted forms of their identities and service information. Any mismatch during the procedure stops the communication and indicates that further interaction is unnecessary. Because only partial information gets exposed in the unnecessary cases, it is highly likely that sensitive information is not exposed to inappropriate participants. By using simple strategies, users and service providers know the number of communication rounds and the number of bits to exchange in each round to reach mutual trust.

Comparison of middleware services.

Earlier we discussed core system services commonly implemented by PvC middleware systems. The Table 3 compares the functionalities available in many of the existing PvC middleware services. We compare the context management, service management, fault tolerance, and security and privacy features of some well-known PvC middleware systems.

4. Challenging open issues

PvC middleware research has many challenging problems [188,189] that need to be addressed for designing more efficient middleware systems.

4.1. General PvC research issues

PvC systems are different from existing computing paradigms. Though they bear many similarities with distributed and mobile computing technologies, still there are many new requirements.

PvC applications aim to assist people with computation support everywhere and all the time. This often requires support for user and application migration across different PvC environments. Due to the resource constrained nature of PvC devices, supporting user and application migration in PvC advocates using available resources in the local environment. This approach is known as resource opportunism. Aura middleware applies this idea in order to support user task migration

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

across different environments. Resource opportunism requires dynamic discovery of useful resources matching application requirements (e.g. for computation and data storage) for smooth, seamless and transparent migration. Also, policies need to be developed for adapting applications to different environments, both local and remote.

PvC algorithms and protocols must be device agnostic, i.e., the focus should be on information/resources available in the environment, rather than devices. Applications should be designed at an upper level, where the device and the user interface has been abstracted. Applications and users in PvC environments are only interested in services or data, and not on devices.

Most of the existing PvC middleware follow a centralized co-ordination. But, this top–down approach of design requires re-thinking, given the presence of a very large number of low-power sensory devices and no guarantee of having a central entity capable of managing all other devices. What is required is an efficient decentralized and bottom–up approach of system design which will work through spontaneous interoperation and de-coupled coordination among participating devices. Researchers should think of a design paradigm which can consider the pros and cons of either design technique and can study how the bottom–up and top–down approaches can co-exist and possibly conflict in future systems [190].

Also, the PvC environment is extremely dynamic in nature and devices frequently join and leave the environment. In order to enable the PvC middleware to cope with the dynamicity, the middleware must implement adaptive functionality with distributed and dynamic deployment and auto-configuration support for newly joined devices.

4.2. Context management issues

PvC applications should be context-aware and adaptive. These objectives can be achieved using situated interactions which require deciding on a particular situation depending on the surrounding environment (e.g. location, proximity, physical conditions, social setting, etc.). Situated interaction needs to study the situation context before triggering actions. Developing smart environments (home, office, meeting rooms) are very much dependent on these types of interactions. Another challenge in developing context-aware applications is to correctly detect user intention based on the situational knowledge. This area has many open challenges and requires further investigation. Conflict resolution among the data sensed by multiple sensor nodes is also open for research.

Another very important challenge concerns storage of contextual data used in PvC applications. Smart systems use sensors, RFID tags, cameras, etc., and the data produced is becoming overwhelmingly large for the existing infrastructure to store and manage. So, developing the means to store as well as meaningfully access and query the collected data is essential.

4.3. Service management issues

PvC applications are service-based and service discovery and access are common operations to serve user needs. Service handoff is also necessary to provide mobile users seamless service access by proactively finding new matching services if the original service becomes unavailable due to any reason [191,192]. Also, many applications may require dynamic service composition in order to build higher-level services composed of several atomic or lower-level services. Reliable service management operations for multi-application service provision are absolutely essential for PvC environments. In order to manage cooperation among a large number of heterogeneous devices and to enable them to adapt to rapidly changing contexts and scenarios, PvC should enforce self-$*$ capabilities, such as autonomic, self-managing, self-organizing, and self-adaptive behaviour in both infrastructure management and service provisioning [193]. Also, ensuring privacy and security of services is also important in such dynamic and open environments.

Providing human-centered services (location-based, or proximity-based services) based on the social context of the user introduces a new class of application in PvC systems [194]. Such services study the user's physical context (location, movement, proximity, etc.) and provide services automatically (push-based, depending on user preference) or on a request–response basis (pull-based).

4.4. Social network-related issues

Nowadays, a new type of application is on the rise, which uses context information of the user's physical environment (location, movement, proximity information, etc.) to study user's individual and social behavior and predict future actions based on that [195,196]. This approach of using PvC related tools and technologies for studying human dynamics and social networks can pose many challenging issues, such as developing suitable algorithms to facilitate large scale data management, providing support for data driven adaptability, managing user's privacy and security, etc.

5. Conclusion

In this survey, we investigated a large number of disparate PvC middleware solutions and infer that PvC middleware helps the programmer develop applications in several ways. First, it provides appropriate system abstractions, so that the application programmer can focus on the application logic without caring too much about the lower-level implementation

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

details. Second, it provides reusable code services, such as code update, and data services, including, data filtering, so that the application programmer can deploy and execute the application without being troubled with complex and tedious functions. Third, it facilitates system resource management and adaptation by providing efficient control services, e.g., data management, service management and context management. Fourth, it supports strong system integration by ensuring tight coupling between the computational and physical elements of the system. Last, but not the least, it enables system monitoring, and ensures security and privacy as well as fault tolerance and reliability support for developed PvC applications.

However, middleware research for PvC environment has not quite matured yet. Many researchers propose different system structures and methodologies, tools and techniques and system support services to help application developers to design PvC middleware systems. In this paper, we have classified existing techniques adopted by PvC middleware systems into different classes (e.g., abstractions, programming models, system architectures, runtime-supports, and system services) and identified their characteristics. We have presented a typical elderly care PvC application scenario in order to highlight common research issues and challenges in designing such applications. We have then critically analyzed those challenges and pointed out the need for designing efficient middleware solutions to address diverse and complex developmental problems. At the end of this survey, we identified some challenging problems and indicated possible future directions of research.

Though many PvC middleware systems have been proposed in recent times, we hardly see widespread use of any in real life. Most middleware systems mentioned in this paper have hardly been used outside the academic research labs. Generally speaking, for the growth of middleware research we must be able to reuse services and functionalities provided by existing middleware systems while developing support for new and hitherto unattended issues. We must also have bridging functionality between different custom PvC middleware systems in order to facilitate interactions between them. E.g., different smart device makers usually employ dedicated middleware systems for their product and this may create obstacles for device interoperation.

Table of acronyms

Acronym	Meaning
AC	Access Control
AI	Artificial Intelligence
AoC	Activity-oriented Computing
API	Application Programming Interface
CCPP	Composite Capabilities Preferences Profile
CORBA	Common Object Request Broker Architecture
DHT	Dynamic Hash Table
HCI	Human–Computer Interaction
IPC	Inter-process Communication
LTE	Long Term Evolution
MANET	Mobile Ad-hoc Network
MVC	Model–View–Controller
NFC	Near-field Communications
P2P	Peer-to-peer
Pub/Sub	Publish/ Subscribe
PvC	Pervasive Computing
QoC	Quality-of-Context
QoS	Quality of Service
RDF	Resource Description Framework
RDQL	RDF Data Query Language
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
SDP	Service Discovery Protocol
SOAP	Simple Object Access Protocol
UAProf	User Agent Profile
UPnP	Universal Plug and Play
WSNs	Wireless Sensor Networks
XML	eXtensible Markup Language

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

References

[1] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, E. Jansen, The Gator Tech Smart House: a programmable pervasive space, IEEE Computer 38 (3) (2005) 50–60.
[2] B. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer, Easyliving: technologies for intelligent environments, in: Proc. of the 2nd International Symposium on Handheld and Ubiquitous Computin, HUC'00, 2000, pp. 12–29.
[3] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, F. Liu, The smart classroom: merging technologies for seamless tele-education, IEEE Pervasive Computing (PERVASIVE) 2 (2) (2003) 47–55.
[4] A. Chen, R.R. Muntz, S. Yuen, I. Locher, S. Park, M.B. Srivastava, A support infrastructure for the smart kindergarten, IEEE Pervasive Computing (PERVASIVE) 1 (2) (2002) 49–57.
[5] H. Chen, F. Perich, D. Chakraborty, T. Finin, A. Joshi, Intelligent agents meet semantic web in a smart meeting room, in: Proc. of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems, AAMAS 2004, July, 2004.
[6] C. Santoro, F. Paternò, G. Ricci, B. Leporini, A multimodal mobile museum guide for all, in: Proc. of the Mobile Interaction with the Real World, MIRW 2007, Workshop at MobileHCI 2007, Singapore, September 11–14, 2007.
[7] M. Bang, A. Larsson, H. Eriksson, NOSTOS: a paper-based ubiquitous computing healthcare environment to support data capture and collaboration, in: Proc. of the 2003 AMIA Annual Symposium, 2003, pp. 46–50.
[8] M. Rodriguez, V. Gonzalez, P. Santana, J. Favela, A home-based communication system for older adults and their remote family, Computers in Human Behaviour Journal 25 (2009) 609–618.
[9] C.D. Kidd, R. Orr, G.D. Abowd, C.G. Atkeson, I.A. Essa, B. MacIntyre, E.D. Mynatt, T. Starner, W. Newstetter, The Aware Home: a living laboratory for ubiquitous computing research, in: Proc. of the Second International Workshop on Cooperative Buildings, CoBuild'99, 1999, pp. 191–198.
[10] D.J. Patterson, O. Etzioni, D. Fox, H. Kautz, Intelligent ubiquitous computing to support Alzheimer's patients: enabling the cognitively disabled, in: Proc. of the First International Workshop on Ubiquitous Computing for Cognitive Aids UniCog, 2002.
[11] J. Bohn, The smart jigsaw puzzle assistant: using RFID technology for building augmented real-world games, in: Proc. of the Pervasive Games Workshop 2004.
[12] M. Kumar, S.K. Das, Pervasive computing: enabling technologies and challenges, in: A. Zomaya (Ed.), Handbook of Nature-Inspired and Innovative Computing, Section III, Springer, 2006, pp. 613–631.
[13] M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt, Gaia: a middleware infrastructure to enable active spaces, IEEE Pervasive Computing (PERVASIVE) (2002) 74–83.
[14] S. Chetan, J. Al-Muhtadi, R.H. Campbell, M.D. Mickunas, Mobile Gaia: a middleware for ad-hoc pervasive computing, in: IEEE Consumer Communications and Networking Conference, CCNC 2005, Las Vegas, January, 2005.
[15] J.P. Sousa, D. Garlan, Aura: an architectural framework for user mobility in ubiquitous computing environments, in: Proc. of the 3rd Working IEEE/IFIP Conference on Software Architecture, WICSA, 2002.
[16] D. Garlan, D.P. Siewiorek, A. Smailagic, P. Steenkiste, Project Aura: toward distraction-free pervasive computing, IEEE Pervasive Computing (PERVASIVE) 1 (2) (2002).
[17] M. Satyanarayanan, Pervasive computing: vision and challenges, IEEE Personal Communications (2001).
[18] M. Kumar, B. Shirazi, S. Das, B.Y. Sung, D. Levine, M. Singhal, PICO: a middleware framework for pervasive computing, IEEE Pervasive Computing (PERVASIVE) (2003) 72–79.
[19] S. Kalasapur, M. Kumar, B. Shirazi, Seamless service composition (SeSCo) in pervasive environments, in: Proc. of the First ACM international Workshop on Multimedia Service Composition, Hilton, Singapore, November 11, 2005. MSC'05, ACM, New York, NY, 2005, pp. 11–20.
[20] S. Kalasapur, M. Kumar, B. Shirazi, Dynamic service composition in pervasive computing systems, IEEE Transactions on Parallel and Distributed Systems 18 (7) (2007) 907–918.
[21] P. Veríssimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday, J. Kaiser, CORTEX: towards supporting autonomous and cooperating sentient entities, in: Proc. European Wireless 2002, Florence, Italy, February, 2002.
[22] R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, D. Wetherall, System support for pervasive applications, ACM Transactions on Computer Systems 22 (4) (2004) 421–486.
[23] R. Grimm, J. Davis, B. Hendrickson, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, D. Wetherall, Systems directions for pervasive computing, in: Proc. of the 8th Workshop on Hot Topics in Operating Systems, HotOS-VIII, May, 2001, pp. 147–151.
[24] S. Kabadayi, C. Julien, A local data abstraction and communication paradigm for pervasive computing, in: Proc. of the of the 5th Annual IEEE International Conference on Pervasive Computing and Communications, March, 2007, pp. 57–66.
[25] J.P. Sousa, V. Poladian, D. Garlan, B. Schmerl, P. Steenkiste, Steps toward activity-oriented computing, in: Next Generation Software Program Workshop, 2008.
[26] M. Wang, J. Cao, J. Siebert, V. Raychoudhury, J. Li, Ubiquitous intelligent object: modeling and applications, in: Proc. of the 3rd International Conference on Semantics, Knowledge and Grid, SKG'07, October 29–31, 2007.
[27] S. Madden, M. Franklin, Fjording the stream: an architecture for queries over streaming sensor data, in: Proc. of the Intl. Conf. on Data Engineering, ICDE'2002, 2002, pp. 555–566.
[28] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: Workshop on Advanced Context Modelling, Reasoning and Management as Part of UbiComp 2004.
[29] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, International Journal of Ad Hoc and Ubiquitous Computing 2 (4) (2007) 263–277.
[30] H. Truong, S. Dustdar, A survey on context-aware web service systems, International Journal of Web Information Systems 5 (1) (2009) 5–31.
[31] C. Cho, D. Lee, Survey of service discovery architectures for mobile ad hoc networks, Unpublished Term Paper, Mobile Computing, CEN 5531, Computer and Information Sciences and Engineering Department, University of Florida Gainesville, USA, 2005.
[32] A.N. Mian, R. Baldoni, R. Beraldi, A survey of service discovery protocols in multihop mobile ad hoc networks, IEEE Pervasive Computing (PERVASIVE) 8 (1) (2009) 66–74.
[33] F. Zhu, M.W. Mutka, L.M. Ni, Service discovery in pervasive computing environments, IEEE Pervasive Computing (PERVASIVE) 4 (4) (2005) 81–90.
[34] C. Mascolo, L. Capra, W. Emmerich, Mobile computing middleware, in: Advanced Lectures on Networking, in: Enrico Gregori, Giuseppe Anastasi, Stefano Basagni (Eds.), Lecture Notes in Computer Science, vol. 2497, Springer-Verlag New York, Inc., New York, NY, USA, 2002, pp. 20–58.
[35] G. Schiele, M. Handte, C. Becker, Pervasive computing middleware, Springer US Handbook of Ambient Intelligence and Smart Environments, Boston, MA, 2010.
[36] M.M. Wang, J. Cao, J. Li, S.K. Das, Middleware for wireless sensor networks: a survey, Journal of Computer Science and Technology 23 (3) (2008) 305–326.
[37] H.B. Christensen, J.E. Bardram, Supporting human activities—exploring activity-centered computing, in: Proceedings of Ubicomp 2002, Springer Verlag, 2002, pp. 107–116.
[38] J.E. Bardram, Activity-based computing: support for mobility and collaboration in ubiquitous computing, Personal and Ubiquitous Computing (2005) 312–322.
[39] J.E. Bardram, C. Bossen, Mobility work—the spatial dimension of collaboration at a hospital, Computer Supported Cooperative Work 14 (2) (2005) 131–160.
[40] J. Bardram, J. Bunde-Pedersen, M. Soegaard, Support for activity-based computing in a personal computing operating system, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2006, ACM Press, New York, New York, USA, 2006.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

[41] W.W. Chu, V. Cerf, T.C. Chen, R.S. Gains, B. Lampson, Some considerations for a high performance message-based interprocess communication system, in: Proceedings of the ACM SIGCOMM/SIGOPS Workshop on Interprocess Communications, 1975.
[42] D. Gelernter, Generative communication in Linda, ACM Transactions on Programming Languages and Systems (TOPLAS) 7 (1) (1985) 80–112.
[43] F. Fabret, H.A. Jacobsen, F. Llirbat, J. Pereira, K.A. Ross, D. Shasha, Filtering algorithms and implementation for very fast publish/subscribe systems, in: Proceedings of ACM SIGMOD, vol. 30, 2001, pp. 115–126.
[44] K.P. Birman, T.A. Joseph, Exploiting virtual synchrony in distributed systems, Operating Systems Review (1987) 123–138.
[45] http://www.omg.org/spec/CORBA/.
[46] http://www.w3.org/TR/soap12-part1/.
[47] http://java.sun.com/developer/onlineTraining/rmi/RMI.html.
[48] A.K. Dey, Understanding and using context, Personal and Ubiquitous Computing 5 (1) (2001) 4–7.
[49] D. Zhang, M. Guo, L. Liu, M. Zhong, X. Liu, K. Ota, X. Zhu, A reference model for context-awareness in pervasive computing environments, in: Pervasive Computing, Nova Publisher, 2011 (Book Chapter).
[50] S. Madden, M. Franklin, J. Hellerstein, W. Hong, TinyDB: an acquisitional query processing system for sensor networks, ACM Transactions on Database Systems 30 (1) (2005) 122–173.
[51] S. Li, Y. Lin, S. Son, J. Stankovic, Y. Wei, Event detection services using data service middleware in distributed sensor networks, Telecommunication Systems 26 (2) (2004) 351–368.
[52] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, J. Kelner, Mires: a publish/subscribe middleware for sensor networks, Personal and Ubiquitous Computing 10 (1) (2005) 37–44.
[53] T. Liu, M. Martonosi, Impala: a middleware system for managing autonomic, parallel sensor systems, in: Proc. of the 9th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming, PPoPP'2003, 2003, pp. 107–118.
[54] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, Tag: a tiny aggregation service for ad-hoc sensor networks, in: Proc. of the 5th Symposium on Operating Systems Design and Implementation, OSDI'2002, pp. 131–146.
[55] P. Bonnet, J.E. Gehrke, P. Seshadri, Towards sensor database systems, in: Proc. of the Second International Conference on Mobile Data Management, MDM'01, Hong Kong, January, 2001.
[56] A. Boulis, C. Han, M. Srivastava, Design and implementation of a framework for efficient and programmable sensor networks, in: Proc. of the 1st Intl. Conf. on Mobile Systems, Applications and Services, MobiSys'2003, 2003, pp. 187–200.
[57] S. Madden, M. Shah, J. Hellerstein, V. Raman, Continuously adaptive continuous queries over streams, in: SIGMOD'2002: Proc. of the 2002 ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 49–60.
[58] R. Rajagopalan, P.K. Varshney, Data aggregation techniques in sensor networks: a survey, IEEE Communications Surveys and Tutorials 8 (4) (2006) 48–63.
[59] M. Wang, J. Cao, Jing Li, S.K. Das, Middleware for wireless sensor networks: a survey, Journal of Computer Science and Technology 23 (3) (2008) 305–326.
[60] A. Ranganathan, J. Al-Muhtadi, R.H. Campbell, Reasoning about uncertain contexts in pervasive computing environments, IEEE Pervasive Computing (PERVASIVE) 3 (2) (2004) 62–70.
[61] M. Mamei, R. Nagpal, Macro programming through Bayesian networks: distributed inference and anomaly detection, in: Proc. of the 5th Annual IEEE Int. Conf. on Pervasive Computing and Communications, Percom'2007, March, 2007, pp. 87–96.
[62] A. Ranganathan, R.H. Campbell, An infrastructure for context awareness based on first order logic, Personal and Ubiquitous Computing 7 (6) (2003) 353–364.
[63] Stanford Center for Biomedical Informatics Research, Protégé. http://protege.stanford.edu/.
[64] IBM China Research Lab, Integrated ontology development toolkit. http://www.alphaworks.ibm.com/tech/semanticstk, December 2007.
[65] A.P. Dempster, A generalization of Bayesian inference, Journal of the Royal Statistical Society, Series B 30 (1968) 205–247.
[66] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, 1976.
[67] F. Perich, A. Joshi, T. Finin, Y. Yesha, On data management in pervasive computing environments, The IEEE Transactions on Knowledge and Data Engineering 16 (5) (2004) 621–634.
[68] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, N. Liampotis, Optimising context data dissemination and storage in distributed pervasive computing systems, Pervasive and Mobile Computing 6 (2) (2010) 218–238.
[69] J. Ma, Spacelog concept and issues for novel u-services in smart spaces, in: Keynote Speech, Int'l Conference on Future Generation Communication and Networking, FGCN'08, December, 2008.
[70] T. Gu, H.K. Pung, D. Zhang, A P2P context lookup service for multiple smart spaces, in: Proc. of the 4th International Conference on Mobile Systems, Applications, and Services, Mobisys'06, Uppsala, Sweden, June, 2006.
[71] Bill Schilit, Norman Adams, Roy Want, Context-aware computing applications, in: Proceedings of the Workshop on Mobile Computing Systems and Applications, 1994.
[72] M. Samulowitz, F. Michahelles, C. Linnhoff-Popien, Capeus: an architecture for context-aware selection and execution of services, in: K. Zielinski, K. Geihs, A. Laurentowski (Eds.), New Developments in Distributed Applications and Interoperable Systems, 2001, pp. 23–40.
[73] C. Ghidini, F. Giunchiglia, Local models semantics, or contextual reasoning = locality + compatibility, Artificial Intelligence 127 (2) (2001) 221–259.
[74] J. McCarthy, Notes on formalizing context, in: Proc. of the 13th Int. Joint Conf. on Artificial Intelligence, vol. 13, 1993, pp. 555–560.
[75] J. McCarthy, S. Buvac, Formalizing context (expanded notes), Computing Natural Language 81 (1998) 13–50.
[76] P. Gray, D. Salber, Modelling and using sensed context information in the design of interactive applications, in: Proc. of 8th IFIP Intl. Conf. on Engineering for Human–Computer Interaction, 2001, pp. 317–336.
[77] X. Chang, S.C. Cheung, W.K. Chan, Y. Chunyang, Heuristics-based strategies for resolving context inconsistencies in pervasive computing applications, in: Proc. of the 28th Intl. Conf. on Distributed Computing Systems, ICDCS'2008, 2008, pp. 713–721.
[78] C. Xu, S.C. Cheung, W.K. Chan, Incremental consistency checking for pervasive context, in: ICSE'2006, Proc. of the 28th Int. Conf. on Software Engineering, 2006, pp. 292–301.
[79] C. Xu, S.C. Cheung, Inconsistency detection and resolution for context-aware middleware support, in: Proc. of the 10th European Software Engineering Conf. Held Jointly with 13th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering, 2005, pp. 336–345.
[80] A. Schmidt, K.A. Adoo, A. Takaluoma, U. Tuomela, K.V. Laerhoven, W.V.D. Velde, Advanced interaction in context, in: Proc. of 1st Int. Symposium on Handheld and Ubiquitous Computing, 1999, pp. 89–101.
[81] A. Schmidt, K. Van Laerhoven, How to build smart appliances? IEEE Personal Communications 8 (4) (2001) 66–71.
[82] K. Cheverst, K. Mitchell, N. Davies, Design of an object model for a context sensitive tourist guide, Computers and Graphics 23 (6) (1999) 883–891.
[83] World Wide Web Consortium (W3C), User agent profile (uaprof). http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20011020-a.pdf, 2001.
[84] World Wide Web Consortium (W3C), Composite capability/preference profiles (CC/PP): structure and vocabularies 2.0—W3C working draft. http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/, 2007.
[85] J. Indulska, R. Robinson, A. Rakotonirainy, K. Henricksen, Experiences in using CC/PP in context-aware systems, in: Proc. of the 4th Int. Conf. on Mobile Data Management, MDM'2003, 2003, pp. 247–261.
[86] K. Henricksen, J. Indulska, A. Rakotonirainy, Modeling context information in pervasive computing systems, in: Proc. of the 1st Int. Conf. on Pervasive Computing, Pervasive'2002, 2002, pp. 167–180.
[87] E. Chtcherbina, M. Franz, Peer-to-peer coordination framework (P2Pc): enabler of mobile ad-hoc networking for medicine, business, and entertainment, in: Proc. of Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet, 2003, pp. 22–29.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

[88] N. Ryan, ConteXtML: exchanging contexual information between a mobile client and the fieldnote server, Technical Document, Computing Laboratory, University of Kent at Canterbury, 1999. http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.
[89] L. Kagal, V. Korolev, H. Chen, A. Joshi, T. Finin, Project Centaurus: a framework for indoor services mobile services, in: Proc. of the 21st Intl. Conf. on Distributed Computing Systems Workshops, ICDCSW'2001, 2001, pp. 195–201.
[90] P. Brown, J. Bovey, X. Chen, Context-aware applications: from the laboratory to the marketplace, IEEE Personal Communications 4 (1997) 58–64.
[91] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, A survey of context modeling and reasoning techniques, Pervasive and Mobile Computing 6 (2) (2010).
[92] X.H. Wang, D.Q. Zhang, T. Gu, H.K. Pung, Ontology based context modeling and reasoning using OWL, in: Proc. of the 2nd IEEE Annual Conf. on Pervasive Computing and Communications Workshops, PercomW'2004, 2004, pp. 18–22.
[93] I. Pandis, J. Soldatos, A. Paar, J. Reuter, M. Carras, L. Polymenakos, An ontology-based framework for dynamic resource management in ubiquitous computing environments, in Proc. of the 2nd Intl. Conf. on Embedded Software and Systems, 2005, pp. 195–203.
[94] N.B. Behlouli, C. Taconet, G. Bernard, An architecture for supporting development and execution of context-aware component applications, in: Proc. of the ACS/IEEE Int. Conf. on Pervasive Services, ICPS'2006, 2006, pp. 57–66.
[95] D. Ejigu, M. Scuturici, L. Brunie, An ontology-based approach to context modeling and reasoning in pervasive computing, in: Proc. of the 5th IEEE Annual Conf. on Pervasive Computing and Communications Workshops, PercomW'2007, 2007, pp. 14–19.
[96] M. Siadaty, C. Torniai, D. Gasevic, J. Jovanovic, T. Eap, M. Hatala, m-LOCO: an ontology-based framework for context-aware mobile learning, in: Proc. of the 6th Int. Workshop on Ontologies and Semantic Web for Intelligent Educational Systems at the 9th Int. Conf. on Intelligent Tutoring Systems, 2008, pp. 21–35.
[97] L. Seremeti, C. Goumopoulos, A. Kameas, Ontology-based modeling of dynamic ubiquitous computing applications as evolving activity spheres, Pervasive and Mobile Computing 5 (5) (2009) 574–591.
[98] B. Hu, B. Hu, J. Wan, M. Dennis, H.-H. Chen, L. Li, Q. Zhou, Ontology-based ubiquitous monitoring and treatment against depression, Wireless Communications and Mobile Computing 10 (10) (2010) 1303–1319.
[99] M. Rosemann, P. Green, M. Indulska, A reference methodology for conducting ontological analyses, in: ER'2004, Proc. of 23rd Intl. Conf. on Conceptual Modeling, 2004, pp. 110–121.
[100] D. Zhang, M. Guo, J. Zhou, D. Kang, J. Cao, Context reasoning using extended evidence theory in pervasive computing environments, Future Generation Computer Systems 26 (2) (2010) 207–216.
[101] D. Zhang, J. Cao, J. Zhou, M. Guo, Extended Dempster–Shafer theory in context reasoning for ubiquitous computing environments, in: IEEE Int. Conf. on Computational Science and Engineering, vol. 2, 2009, pp. 205–212.
[102] A. Kofod Petersen, M. Mikalsen, Context: representation and reasoning: representing and reasoning about context in a mobile environment, Revue D'Intelligence Artificielle 19 (3) (2005) 479–498.
[103] J. Nieto, M. Gutierrez, B. Lancho, Developing home care intelligent environments: from theory to practice, in: Proceedings fo the 7th Int. Conf. on Practical Applications of Agents and Multi-Agent Systems, 2009, pp. 2–12.
[104] X. Wang, D. Zhang, T. Gu, H. Pung, Ontology based context modeling and reasoning using OWL, in: Proc. of the 2nd IEEE Annual Conf. on Pervasive Computing and Communications Workshops 2004, 2004, pp. 18–22.
[105] A. Bikakis, T. Patkos, G. Antoniou, D. Plexousakis, A survey of semantics-based approaches for context reasoning in ambient intelligence, in: Proc. of the Workshop Artificial Intelligence Methods for Ambient Intelligence, Springer, 2007, pp. 15–24.
[106] P. Delir Haghighi, S. Krishnaswamy, A. Zaslavsky, M.M. Gaber, Reasoning about context in uncertain pervasive computing environments, in: EuroSSC'2008, Proc. of the 3rd European Conf. on Smart Sensing and Context, 2008, pp. 112–125.
[107] C.B. Anagnostopoulos, P. Pasias, S. Hadjiefthymiades, A framework for imprecise context reasoning, in: Proc. of IEEE Int. Conf. on Pervasive Services, ICPS'07, Istanbul, Turkey, July, 2007, pp. 181–184.
[108] A.A. Eldin, J. van den Berg, R. Wagenaar, A fuzzy reasoning scheme for context sharing decision making, in: ICEC'2004, Proc. of the 6th Int. Conf. on Electronic Commerce, 2004, pp. 371–375.
[109] T.D. Hodes, Steven E. Czerwinski, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, An architecture for secure wide-area service discovery, ACM Wireless Networks Journal 8 (2–3) (2002) 213–230.
[110] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, The design and implementation of an intentional naming system, in: Proc. of Seventeenth ACM Symposium on Operating Systems Principles, SOSP'99, ACM Press, Charleston, SC, 1999, pp. 186–201.
[111] The Salutation Consortium, Salutation architecture specification version 2.0c. Available online at: http://www.salutation.org/, June, 1999.
[112] Jini Technology core platform specification, v. 2.0, Sun Microsystems. www.sun.com/software/jini/specs/core2_0.pdf, June 2003.
[113] S. Cheshire, M. Krochmal, DNS-based service discovery, IETF Internet Draft. http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt, September 2008.
[114] E. Guttman, C. Perkins, Service location protocol, version 2, June 1999.
[115] UPnP device architecture 1.0, UPnP Forum. www.upnp.org/resources/documents/CleanUPnPDA10120031202s.pdf, December, 2003.
[116] V. Sundramoorthy, J. Scholten, P.G. Jansen, P.H. Hartel, Service discovery at home, in: Proceedings of Fourth International Conference on Information, Communications and Signal Processing and Fourth IEEE Pacific—Rim Conference on Multimedia, ICICS/PCM, IEEE Computer Society Press, Singapore, 2003, pp. 1929–1933.
[117] C. Lee, S. Helal, A multi-tier ubiquitous service discovery protocol for mobile clients, in: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'03, Montréal, Canada, 2003.
[118] M. Balazinska, H. Balakrishnan, D. Karger, INS/Twine: a scalable peer-to-peer architecture for intentional resource discovery, in: Proceedings of International Conference on Pervasive Computing 2002, August, 2002.
[119] K. Arabshian, H. Schulzrinne, GloServ: global service discovery architecture, in: Proceedings of MobiQuitous, June 2004, pp. 319–325.
[120] R. Robinson, J. Indulska, Superstring: a scalable service discovery protocol for the wide area pervasive environment, in: Proceedings of the Eleventh IEEE International Conference on Networks, Sydney, September 2003.
[121] E. Kang, M.J. Kim, E. Lee, U. Kim, DHT-based mobile service discovery protocol for mobile ad hoc networks, in: Proceedings of the Fourth International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications—with Aspects of Theoretical and Methodological Issues, ICIC'08, September, 2008.
[122] H.J. Yoon, E.J. Lee, H. Jeong, J.S. Kim, Proximity-based overlay routing for service discovery in mobile ad hoc networks, in: Proceedings of Nineteenth International Symposium on Computer and Information Sciences, ISCIS, 2004.
[123] H. Pucha, S. Das, Y. Hu, Ekta: an efficient DHT substrate for distributed applications in mobile ad hoc networks, in: Proceedings of Sixth IEEE Workshop on Mobile Computing Systems and Applications, WMCSA, 2004.
[124] F. Delmastro, From pastry to CrossROAD: CROSS-layer ring overlay for ad hoc networks, in: Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communication Workshops, March, 2005, pp. 60–64.
[125] M. Caesar, M. Castro, et al. Virtual ring routing: network routing inspired by DHTs, in: Proceedings of ACM SIGCOMM, 2006, pp. 351–362.
[126] Bluetooth SIG. Specification. http://bluetooth.com/.
[127] M. Nidd, Service discovery in DEAPspace, IEEE Personal Communications (2001) 39–45.
[128] O.V. Ratsimor, D. Chakraborty, A. Joshi, T. Finin, Allia: alliance-based service discovery for ad hoc environments, in: Proceedings of ACM Workshop on Mobile Commerce, WMC'02, September, 2002.
[129] D. Chakraborty, A. Joshi, T. Finin, Y. Yesha, GSD: a novel group-based service discovery protocol for MANETs, in: Proceedings of Fourth IEEE Conference on Mobile and Wireless Communications Networks, MWCN, September, 2002.
[130] D. Chakraborty, A. Joshi, Y. Yesha, T. Finin, Toward distributed service discovery in pervasive computing environments, IEEE Transactions on Mobile Computing (2006).

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

[131] S. Helal, N. Desai, V. Verma, C. Lee, Konark—a service discovery and delivery protocol for ad hoc networks, in: Proceedings of the Third IEEE Conference on Wireless Communication Networks WCNC, March, 2003.
[132] T. Zahn, J. Schiller, MADPastry: a DHT substrate for practicably sized MANETs, in: Proceedings of Fifth Workshop on Applications and Services in Wireless Networks, ASWN, June, 2005.
[133] A. Rowstron, P. Druschel, Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms, Middleware, in: Lecture Notes in Computer Science (LNCS), vol. 2218, Heidelberg, Germany, November, 2001, pp. 329–350.
[134] D. Johnson, D. Maltz, J. Broch, DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, Addison-Wesley, 2001, pp. 139–172 (Chapter 5).
[135] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings of Second IEEE Workshop on Mobile Computer Systems and Applications, IEEE Computer Society, 1999, pp. 90–100.
[136] T. Clausen, P. Jacquet, Optimized link state routing protocol (OLSR), RFC 3626, October, 2003.
[137] M. Klein, B. Konig-Ries, P. Obreiter, Service rings—a semantic overlay for service discovery in ad hoc networks, in: DEXA Workshops, 2003, pp. 180–185.
[138] M. Klein, B. Konig-Ries, P. Obreiter, Lanes—a light weight overlay for service discovery in mobile ad hoc networks, Technical Report 2003-6, University of Karlsruhe, May 2003.
[139] U.C. Kozat, L. Tassiulas, Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues, Ad Hoc Networks 2 (1) (2004) 23–44.
[140] F. Zhu, M. Mutka, L. Ni, Splendor: a secure, private and location-aware service discovery protocol supporting mobile services, in: Proceedings of the First International Conference on Pervasive Computing and Communication PerCom'03, 2003, pp. 235–242.
[141] J. Tyan, Q.H. Mahmoud, A comprehensive service discovery solution for mobile ad hoc networks, ACM/Kluwer Journal of Mobile Networks and Applications (MONET) 10 (8) (2005) 423–434.
[142] F. Sailhan, V. Issarny, Scalable service discovery for MANET, in: Proc. of IEEE PerCom'05, 2005, pp. 235–246.
[143] M.J. Kim, M. Kumar, B.A. Shirazi, Service discovery using volunteer nodes in heterogeneous pervasive computing environments, Pervasive and Mobile Computing 2 (2006) 313–343.
[144] J. Robinson, I. Wakeman, T. Owen, Scooby: middleware for service composition in pervasive computing, in: Proc. of the 2nd Workshop on Middleware For Pervasive and Ad-Hoc Computing (Toronto, Ontario, Canada, October 18–22, 2004) MPAC'04, Vol. 77, ACM, New York, NY, 2004, pp. 161–166.
[145] M. Vallee, F. Ramparany, L. Vercouter, Flexible composition of smart device services, in: Proc. of the International Conference on Pervasive Systems and Computing, PSC-05, Las Vegas, USA, June 2005, pp. 27–30.
[146] M.W. Newman, J.Z. Sedivy, C.M. Neuwirth, W.K. Edwards, J.I. Hong, S. Izadi, K. Marcelo, T.F. Smith, Designing for serendipity: supporting end-user configuration of ubiquitous computing environments, in: Proc. of the 4th Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques, London, England, June 25–28, 2002. DIS'02, ACM, 2002, pp. 147–156.
[147] A. Bottaro, A. Gérodolle, P. Lalanda, Pervasive service composition in the home network, in: Proc. of the 21st International Conference on Advanced Networking and Applications, AINA'07, IEEE Computer Society, Washington, DC, pp. 596–603.
[148] S.B. Mokhtar, D. Fournier, N. Georgantas, V. Issarny, Context-aware service composition in pervasive computing environments, in: Proc. of the Rapid Integration of Software Engineering Techniques, RISE'05, in: LNCS, vol. 3943, Springer, Berlin, Germany, 2006, pp. 129–144.
[149] S.B. Mokhtar, J. Liu, N. Georgantas, V. Issarny, QoS-aware dynamic service composition in ambient intelligence environments, in: Proc. of the 20th IEEE/ACM international Conference on Automated Software Engineering, ASE'05, pp. 317–320.
[150] D. Chakrabory, A. Joshi, T. Finin, Y. Yesha, Service composition for mobile environments, Mobile Networks and Applications 10 (4) (2005) 435–451.
[151] J. Siebert, J.N. Cao, L. Cheng, E. Wei, C. Chen, J. Ma, Decentralized service composition in pervasive computing environments, in: International Wireless Communications and Mobile Computing Conference, IWCMC 2010.
[152] C. Xu, S.C. Cheung, Inconsistency detection and resolution for context-aware middleware support, in: SIGSOFT ESEC/FSE-13, Proc. of the 10th European Software Engineering Conference, 2005, pp. 336–345.
[153] S. Jeffery, M. Garofalakis, M. Franklin, Adaptive cleaning for RFID data streams, in: Proc. of the 32nd Int. Conf. on Very Large Data Bases, 2006, pp. 163–174.
[154] C. Floerkemeier, M. Lampe, Issues with RFID usage in ubiquitous computing applications, in: Proc. of the 2nd Int. Conf. on Pervasive Computing, 2004, pp. 188–193.
[155] E. Elnahrawy, B. Nath, Cleaning and querying noisy sensors, in: WSNA'2002, Proc. of the 2nd ACM Int. Conf. on Wireless Sensor Networks and Applications, 2003, pp. 78–87.
[156] P. Bonnet, J. Gehrke, P. Seshadri, Querying the physical world, IEEE Personal Communications 7 (5) (2000) 10–15.
[157] S. Jeffery, G. Alonso, M. Franklin, W. Hong, J. Widom, Declarative support for sensor data cleaning, in: Lecture Notes in Computer Science, 2006, pp. 83–100.
[158] Z. Yongzhen, C. Lei, X.S. Wang, L. Jie, A weighted moving average-based approach for cleaning sensor data, in: ICDCS'2007, Proc. of the 27th Int. Conf. on Distributed Computing Systems, 2007, pp. 38–48.
[159] R. Reichle, M. Wagner, M. Khan, K. Geihs, M. Valla, C. Fra, N. Paspallis, G. Papadopoulos, A context query language for pervasive computing environments, in: Percom'2008, Proc. of the 6th IEEE Int. Conf. on Pervasive Computing and Communications, 2008, pp. 434–440.
[160] S.-H. Eo, W. Zha, B.-S. You, D.-W. Lee, H.-Y. Bae, Intelligent Context-Awareness System Using Improved Self-Adaptive Back Propagation Algorithm, Vol. 4761, Springer, Berlin, Heidelberg, 2007, pp. 329–338 (Chapter 9).
[161] B. Marie-Luce, Handling uncertainty in multimodal pervasive computing applications, Computer Communications 31 (18) (2008) 4234–4241. 1465801.
[162] Y. Bu, S. Chen, J. Li, X. Tao, J. Lu, Context consistency management using ontology based model, in: Int. Conf. on Extending Database Technology, 2006, pp. 741–755.
[163] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, J. Lu, Managing quality of context in pervasive computing, in: QSIC'06, Proc. of the 6th Int. Conf. on Quality Software, 2006, pp. 193–200.
[164] A. Bikakis, G. Antoniou, Local and distributed defeasible reasoning in multi-context systems, in: RuleML'08: Proc. of the Int. RuleML Symposium on Rule Interchange and Applications, Springer, 2008, pp. 135–149.
[165] A. Ranganathan, R. Campbell, A. Ravi, A. Mahajan, Conchat: a context-aware chat program, IEEE Pervasive Computing (PERVASIVE) 1 (3) (2002) 51–57.
[166] A. Bikakis, F. Antoniou, Distributed reasoning with conflicts in a multi-context framework, in: D. Fox and C. P. Gomes (Eds.), AAAI'08: Proc. of the 23rd AAAI Conf. on Artificial Intelligence, 2008, pp. 1778–1779.
[167] I. Park, D. Lee, S. Hyun, A dynamic context-conflict management scheme for group-aware ubiquitous computing environments, in: COMPSAC'05: Proc. of the 29th Annual Int. Computer Software and Applications Conference, vol. 1, 2005.
[168] Y. Huang, X. Ma, J. Cao, X. Tao, J. Lu, Concurrent event detection for asynchronous consistency checking of pervasive context, in: Percom'2009: Proc. of the 5th IEEE Int. Conf. on Pervasive Computing and Communications, 2007, pp. 37–46.
[169] A. Ranganathan, R. Campbell, A. Ravi, A. Mahajan, Conchat: a context-aware chat program, IEEE Pervasive Computing (PERVASIVE) 1 (3) (2002) 51–57.
[170] J. Chomicki, J. Lobo, S. Naqvi, Conflict resolution using logic programming, IEEE Transactions on Knowledge and Data Engineering 5 (1) (2003) 244–249.
[171] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, J. Lv, Managing quality of context in pervasive computing, in: Proc. the 6th International Conference on Quality Software, Beijing, China, October, 2006, pp. 193–200.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006

[172] A. Manzoor, H. Truong, S. Dustdar, Using quality of context to resolve conflicts in context-aware systems, in: Lecture Notes in Computer Science, vol. 5786, Springer, Berlin, Heidelberg, 2009, pp. 144–155 (Chapter 4).
[173] A.R. Beresford, F. Stajano, Location privacy in pervasive computing, IEEE Pervasive Computing (PERVASIVE) 2 (1) (2003) 46–55.
[174] V. Raychoudhury, J. Cao, W. Wu, Y. Lai, *K*-directory community: reliable service discovery in MANET, Journal of Pervasive and Mobile Computing (JPMC) 7 (1) (2011).
[175] V. Raychoudhury, J. Cao, W. Wu, Top *K*-leader election in wireless ad hoc networks, in: Proc. of the 17th International Conference on Computer Communication Networks, ICCCN08, St. Thomas, US, Virgin Islands, August 4–7, 2008.
[176] K. Carey, D. Lewis, S. Higel, V. Wade, Adaptive composite service plans for ubiquitous computing, in: Proc. of the 2nd International Workshop on Managing Ubiquitous Communications and Services, MUCS, December 2004.
[177] W.L.C. Lee, S. Ko, S. Lee, A. Helal, Context-aware service composition for mobile network environments, in: Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing, UIC2007, 2007.
[178] U. Bellur, N.C. Narendra, Towards service orientation in pervasive computing systems, in: Proc. of the International Conference on Information Technology: Coding and Computing, ITCC, 2005, pp. 289–295.
[179] A. Ranganathan, R.H. Campbell, Autonomic pervasive computing based on planning, in: Proc. of the International Conference on Autonomic Computing, 2004, pp. 80–87.
[180] S. Czerwinski, B.Y. Zhao, T. Hodes, A. Joseph, R. Katz, An architecture for a secure service discovery service, in: Proc. MobiCom, 1999.
[181] Bluetooth Security, White Paper, Bluetooth SIG Security Expert Group. http://grouper.ieee.org/groups/1451/5/Comparison%20of%20PHY/Bluetooth_24Security_Paper.pdf, 2002.
[182] C. Ellison, UPnP security ceremonies V1.0, Intel Co. October 2003. http://www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf.
[183] F. Zhu, M. Mutka, L. Ni, Prudent exposure: a private and user-centric service discovery protocol, in: Proc. of the IEEE Conference on Pervasive Computing and Communications, Percom'04, March, 2004.
[184] F. Zhu, W. Zhu, M. Mutka, L. Ni, Expose or not? a progressive exposure approach for service discovery in pervasive computing environments, in: Proc. of the IEEE Conference on Pervasive Computing and Communications, Percom'05, March, 2005.
[185] F. Zhu, M. Mutka, L. Ni, A private, secure and user-centric information exposure model for service discovery protocols, IEEE Transactions on Mobile Computing 5 (4) (2006) 418–429.
[186] F. Zhu, W. Zhu, M. Mutka, L. Ni, Private and secure service discovery via progressive and probabilistic exposure, IEEE Transactions on Parallel and Distributed Systems 18 (11) (2007) 1565–1577.
[187] B. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (1970) 422–426.
[188] A. Schmidt, S. Spiekermann, A. Gershman, F. Michahelles, Real-world challenges of pervasive computing, IEEE Pervasive Computing (PERVASIVE) 5 (3) (2006) 91–93.
[189] M. Conti, S.K. Das, C. Bisdikian, M. Kumar, L.M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, F. Zambonelli, Looking ahead in pervasive computing: challenges and opportunities in the era of cyber—physical convergence, Pervasive and Mobile Computing 8 (1) (2012) 2–21.
[190] B.H.C. Cheng, et al., Software engineering for self-adaptive systems: a research roadmap, in: Self-Adaptive Software, in: LNCS, vol. 5525, Springer-Verlag, 2009, pp. 1–16.
[191] O. Riva, T. Nadeem, C. Borcea, L. Iftode, Context-aware migratory services in ad hoc networks, IEEE Transactions on Mobile Computing 6 (12) (2007) 1313–1328.
[192] V. Raychoudhury, J. Cao, W. Wu, C. Hui, Service handoff for reliable and continuous service access in pervasive computing, in: Proceedings of 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing, PDP, Ayia Napa, Cyprus, February 9–11, 2011.
[193] S. Dobson, S. Denazis, A. Fernández, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, A survey of autonomic communications, ACM Transactions on Autonomous and Adaptive Systems 1 (2) (2006) 223–259.
[194] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, J. Crowcroft, Recommending social events from mobile phone location data, in: 10th IEEE International Conference on Data Mining, ICDM 10, Sydney, Australia 2010.
[195] V. Kostakos, E. O'Neill, A. Penn, G. Roussos, D. Papadogkonas, Brief encounters: sensing, modeling and visualizing urban mobility and copresence networks, ACM Transactions on Computer and Human Interaction 17 (1) (2010).
[196] F. Calabrese, J. Reades, C. Ratti, Eigenplaces: segmenting space through digital signatures, IEEE Pervasive Computing (PERVASIVE) 9 (2010) 78–84.

Please cite this article in press as: V. Raychoudhury, et al., Middleware for pervasive computing: A survey, Pervasive and Mobile Computing (2012), doi:10.1016/j.pmcj.2012.08.006