ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

A Survey on Context-Aware Middleware

Chitra G. Sahu[1], **Dr. D. S. Adane**[2]

Research Scholar, Department of Computer Science and Engineering, ShriRamdeobaba College of Engineering and Management, Nagpur, India.[1]

Head of Department, Department of Information Technology, ShriRamdeobaba College of Engineering and Management, Nagpur, India.[2]

Abstract: The middleware deals with different functionality such as service discovery, asynchronous messaging, publish/subscribe event management,managingand storage of context information, constructing the user interface and management of the local and network resources. Also, it allows controlling the connectivity of the device; the middleware is proficient to switch traffic from one network connection to another.
In this literature review paper, we are presenting the experience of several research papers related to the next generation of middleware systems. Our work is concerned with providing basic understanding about the current requirements of mobile computing systems and how they can be achieved by modifying the existing middleware systems. Initially, we have defined Mobile Computing, its functions and different Mobile Platforms. Further, we have defined middleware and described its common features. Purposely, we have indicated the major challenges in mobile computing systems and analyzed the requirements of mobile computing. Following this, we review the different categories of mobile middleware technologies and show their relative strengths and weaknesses. On a broader perspective, we have tried to identify the major requirements for context-aware middleware systems and discussed how context-aware middleware can resolve the major issues in the development of application for pervasive computing. Also, we have highlighted the latest trend in context-aware computing; Internet of Things (IoT).

Keywords: Mobile Computing; Middleware; Context-Aware Middleware; Internet of Things.

I. INTRODUCTION

The availability of lightweight, portable computers and wireless technologies has created a new class of applications called mobile applications, presenting challenging problems to the designers. These applications often run on scarce resource platforms such as personal digital assistants, notebooks, and mobile phones, each of which have limited CPU power, memory, and battery life. They are usually connected to wireless links, which are characterized by lower bandwidths, higher error rates and more frequent disconnections. To support designers building mobile applications, research in the field of middleware systems has proliferated. Middleware aims at facilitating communication and coordination of distributed components, concealing complexity raised by mobility from application developers as much as possible. Most distributed applications and services were designed with the assumption that the terminals were powerful and connected to fixed networks. Conventional middleware technologies distribution and thus, have focused on concealingthe problems of heterogeneity to enable the development of distributed systems. They let the application developers to focus on application functionality rather than on handling explicitly with distribution issues. Traditional middleware systems such as CORBA, DCOM and Java RMI have shown their suitability for standard client-server applications.
However, under the highly variable computing environmental conditions that characterize mobile platforms, it is believed that existing traditional middleware systems are not capable of providing adequate support for the mobile wireless computing environment.
There is a great demand for designing modern middleware systems that can support new requirements imposed by mobility and as context aware middleware deals with the current context of the user, we found in our survey that it is the only middleware that can meet the changing requirements as it can adapt itself according to the changing context. Thus we have focused our survey on context-aware middleware as it is the most efficient middleware for mobile computing systems.
In this paper Section II defines middleware and gives common features and functions of middleware. In Section III we present a detailed description of context-aware middleware, defines context describes its life cycle and highlights main characteristics of context information; Section IV focuses on the issues related to context-aware middleware. Section V points out future directions of research in the area of context-aware middleware for mobile computing and Section VI concludes the paper.

II. MIDDLEWARES

Middleware plays a vibrant role in hiding the complexity of distributed applications. These applications usually operate in an environment that may include heterogeneous computer architectures, operating systems, network protocols, and databases. It is unpleasant for an application developer to deal with such heterogeneous plumbing. Middleware's primary role is to conceal this complexity from developers by deploying an isolated layer of APIs. This layer bridges the gap between application program and platform dependency.
Middleware is defined as follows by Linthicum D.[1] –

“Middleware is an enabling layer of software that resides between the application program and the networked layer of heterogeneous platforms and protocols. It decouples applications from any dependencies on the plumbing layer that consists of heterogeneous operating systems, hardware platforms and communication protocols”.

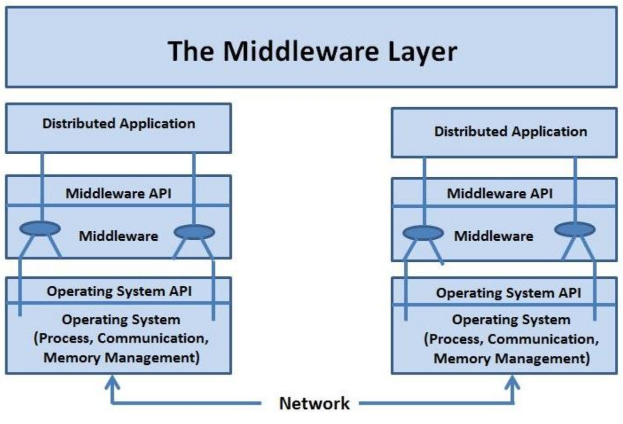


Fig 1.Middleware Layer

A. Common Features of Middleware

The following common features of wireless middleware products are needed to support mobile computing applications [2]:

Connection and message delivery: Middleware helps establish connections between mobile clients and servers over wireless networks and delivers messages over the connection. It also stores and forwards messages if the user is disconnected from the network.

Transformation: The middleware transforms data from one format to another (e.g., HTML to WML). The transformation may be intelligent enough to transform different types of data to different types of devices.

Detection and storage: Wireless middleware products can detect and store mobile device characteristics in a database. Upon detecting the type of mobile device or channel being used, the middleware can optimize the wireless data output according to device attributes.

Optimization: Middleware products can compress data to minimize the amount of data being sent over a slow cellular wireless link.

Security: Security features can be imbedded in wireless middleware to ensure end-to-end security. For example, digital certificates for handheld devices can be managed by a middleware service.

Operation support: Middleware can offer network and systems management utilities and tools to allow monitoring and troubleshooting wireless devices and networks.

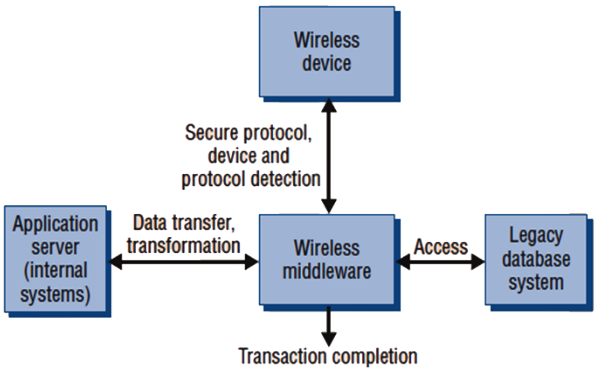

Wireless device
Secure protocol, device and protocol detection
Application server (internal systems)
Data transfer, transformation
Wireless middleware
Access
Legacy database system
Transaction completion

Fig 2. Functions of Wireless Middleware

The figure shows how middleware allows device, application server and database to work together as a mobile application. It also identifies the device and the communication protocol used which lets other parties know how to communicate with the device, it gives database and device a transparent route to each other’s data and it also provide API to the application server for working with the device.

III. CONTEXT-AWARE MIDDLEWARE

Mobile systems run in an extremely dynamic environment. The execution context changes frequently due to the user’s mobility. Mobile hosts often roam around different areas, and services that are available before disconnecting may not be available after reconnecting. Also, the bandwidth and connectivity quality may quickly alter based on the mobile host movements and their locations. The application developers cannot predict all the possible execution contexts that allow the application to know how to react in every scenario. The middleware has to expose the context information to the application to make it aware of the dynamic changes in execution environment. The application then instructs the middleware on how to adapt its own behavior in order to achieve the best quality of service. Many research groups gave special attention in particular to location awareness. For example, location information was exploited to provide travelers directional guidance, to discover neighboring services and to broadcast messages to users in a specific area. Most location-aware systems depend on the underlying network operating system to obtain location information and generate a suitable format to be used by the system. The heterogeneity of coordination information is not supported and hence different positioning systems are required to deal with different sensor technologies, such as the Global Positioning System (GPS) outdoors, and infrared and radio frequency indoors.
Context awareness involves performing data acquisition from sensors, context recognition and other tasks necessary to complete before the context can actually be used. Delegating the data acquisition and context processing tasks to applications makes them almost impossible to reuse. One solution to such a problem is to decouple the tasks from applications and move desired functionality to the lower layers. Such layers, which serve the needs of applications, usually form a special layer called middleware. This middleware layer hides the

heterogeneity and distributed nature of devices measuring the context information. A context-aware middleware serves the context needs of applications.

A context-aware middleware has to provide the applications with the following context-oriented functionality [3]:

- support of a variety of sensor devices
- support of the distributed nature of context information, because the data comes from different sources
- providing for transparent interpretation of applications and abstraction of context data
- maintenance of context storage
- control of the context data flow

Mobility introduces a number of constraints to the middleware [4]:

- the bandwidth is low and hosts can be unreachable due to network partitions or poor coverage
- the local resources like memory capacity and CPU power of the device are very limited
- the communication between the system components is asynchronous
- the execution environment is dynamic

Current Context Aware Middleware free developers from the implementation of low-level details related to the network, like Concurrency Control, Transaction management, Network Communication, so focus is on application requirements. Modern distributed applications need a middleware that is capable of adapting to environment changes.

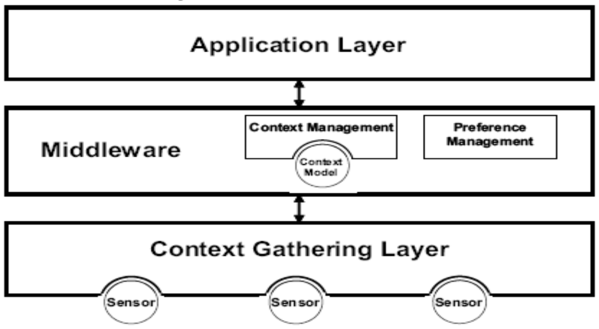


Fig 3. Architecture of a context-aware system

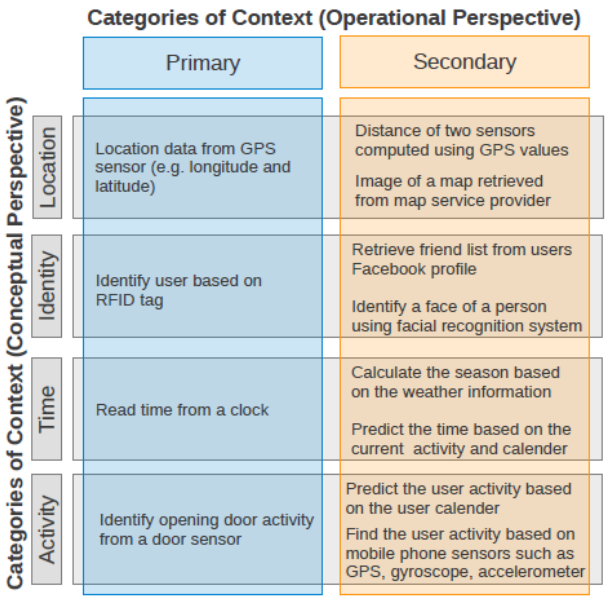
As per Marco B. and Leonardo B. [6], Context-awareness involves acquisition of contextual information, reasoning about context and modifying one's behavior based on the current context. A middleware for context awareness would provide support for each of these tasks. It would also define a common model of context, which all agents can use in dealing with context. It would also ensure that different agents in the environment have a common semantic understanding of contextual information.

"A context attribute is an element of the context model describing the context. A context attribute has an identifier, a type and a value, and optionally a collection of properties describing specific characteristics [7]."

As per Perera C. et al.[8], there are two main forms of context that are often used by the applications-

Primary context: Any information retrieved without using existing context and without performing any kind of sensor data fusion operations (e.g. GPS sensor readings as location information).

Secondary context: Any information that can be computed using primary context. The secondary context can be computed by using sensor data fusion operations or data retrieval operations such as web service calls (e.g. identify the distance between two sensors by applying sensor data fusion operations on two raw GPS sensor values). Further, retrieved context such as phone numbers, addresses, email addresses, birthdays, list of friends from a contact information provider based on a personal identity as the primary context can also be identified as secondary context.


Categories of Context (Operational Perspective)

Categories of Context (Conceptual Perspective)

	Primary	Secondary
Location	Location data from GPS sensor (e.g. longitude and latitude)	Distance of two sensors computed using GPS values Image of a map retrieved from map service provider
Identity	Identify user based on RFID tag	Retrieve friend list from users Facebook profile Identify a face of a person using facial recognition system
Time	Read time from a clock	Calculate the season based on the weather information Predict the time based on the current activity and calender
Activity	Identify opening door activity from a door sensor	Predict the user activity based on the user calender Find the user activity based on mobile phone sensors such as GPS, gyroscope, accelerometer

Fig 4. Context categorization in two different perspectives: conceptual and operational

With advances in technology sensors are being deployed at a significant rate. It generate enormous amount of raw data, all the raw data gathered from sensors are understood and added value by their collection, modelling, distribution and reasoning about the context. Huebscher M. C., McCann J. A., [9] has given the layers involved in context provision and is as depicted in the figure below:

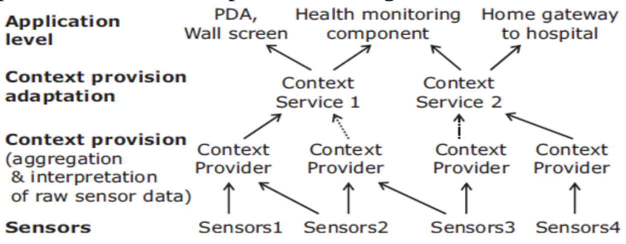


Fig 5. Layers in context provision

Sensors are at the bottom layer todeliver raw sensor data. These could be wireless sensor networks, ultrasonic badges for location, RFID tags for identification, video cameras for tracking, or others. These produce raw sensor data, often preprocessed for saving communication cost as much as the sensor devices allow. These data are passed up one level in to the *context providers*.

Context providers (CPs) are components (software or hardware) that aggregate and interpret the sensor data to produce some higher-level context, e.g. location, identity,

type of activity, health condition of a person. As illustrated in the figure, more than one CP may access the same group of sensors. Similarly, it is possible for one CP to use sensor data from a group of sensors, as data redundancy often improves the total reliability of the derived context.
Context services (CSs) connects below to different context providers that provide the same type of context, e.g. location, but implemented using different underlying sensors, and above connect to the applications. They allow an application to use a type of context while abstracting from the actual instance of a context provider. An application can poll for context information or subscribe for notification when a certain condition on the context information is met, e.g. when the temperature exceeds 26ºC or when Alice enters the kitchen.
Certain CPs and applications may use context information from a CS to provide higher-level context. For instance, a CS may provide location in the form of 2D or 3D coordinates in the home. A CP may then use this information to determine the room in the home this location maps to. Again, this higher-level context would be delivered through a context service.

A. *Context Life Cycle*

Dey provided a definition for context as follows [5]:
"Context is 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."

This *section* shows the movement of context in context-aware systems. In addition to the life cycles, [10] identified three phases in a typical context management system: context acquisition, information processing, and reasoning and decision. Perera C.et al. [8] derived an appropriate (i.e. minimum number of phases but includes all essential) context life cycle as depicted in Figure 6. This is the simplest form of a context life cycle. These four steps are essential in context management systems and middleware solutions.

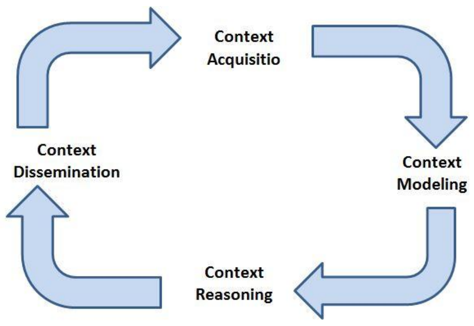


Fig 6. Context life cycle

This context life cycle consists of four phases. First, context needs to be acquired from various sources. The sources could be physical sensors or virtual sensors (context acquisition). Second, the collected data needs to be modeled and represent according to a meaningful manner (context modeling). Third, modeled data needs to be processed to derive high-level context information from low-level raw sensor data (context reasoning). Finally, both high-level and low-level context needs to be distributed to the consumers who are interested in context (context dissemination).

1) *Context Acquisition:*

There are three ways to acquire context: sense, derive, and manually provided.
Sense: The data is sensed through sensors, including the sensed data stored in databases (e.g. retrieve temperature from a sensor, retrieve appointments details from a calendar).
Derive: The information is generated by performing computational operations on sensor data. These operations could beas simple as web service calls or as complex as mathematical functions run over sensed data (e.g. calculate distance between two sensors using GPS coordinates). The necessary data should be available to apply any numerical or logical reasoning technique.
Manually provided: Users provide context information manually via predefined settings options such as preferences. This method can be used to retrieve any type of information.

2) *Context Modelling:*

"A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time [7]." Context modelling is also widely referred to as context representation. Typically, there are two steps in representing context according to a model:
Context modelling process: In the first step, new context information needs to be defined in terms of attributes, characteristics, relationships with previously specified context, quality-of context attributes and the queries for synchronous context requests.
Organize context according to the model: In the second step, the result of the context modelling step needs to be validated. Then the new context information needs to be merged and added to the existing context information repository. Finally, the new context information is made available to be used when required.

3) *Context Reasoning:*

Context reasoning can be defined as a method of deducing new knowledge, and understanding better, based on the available context [11]. It can also be explained as a process of giving high-level context deductions from a set of contexts [12]. The requirement of reasoning also emerged due to two characteristics of raw context: imperfection (i.e. unknown, ambiguous, imprecise, or erroneous) and uncertainty. Context reasoning is broadly divided into three phases.[13]
Context pre-processing: This phase cleans the collected sensor data. Due to inefficiencies in sensor hardware and network communication, collected data may be not accurate or missing. Therefore, data needs to be cleaned by filling missing values, removing outliers, validating context via multiple sources, and many more. These tasks have been extensively researched by database, data mining, and sensor network research communities over many years.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

Sensor data fusion: It is a method of combining sensor data from multiple sensors to produce more accurate, more complete, and more dependable information that could not be achieve through a single sensor [14]. In the IoT, fusion is extremely important, because there will be billions of sensors available. As a result, a large number of alternative sources will exist to provide the same information.

Context inference: Generation of high-level context information using lower-level context. The inferencing can be done in a single interaction or in multiple interactions. For example, in the first iteration, longitude and latitude values of a GPS sensor may be inferred as PurplePickle cafe in canberra. In the next iteration PurplePickle café in canberra may be inferred as John's favourite cafe. Each iteration gives more accurate and meaningful information.

4) *Context Distribution:*

Context distribution is a straight forward task. It provides methods to deliver context to the consumers. From the consumer perspective this task can be called context acquisition. There are two methods to that are often used in context distribution:

Query: Context consumer makes a request in terms of a query, so the context management system can use that query to produce results.

Subscription (also called publish / subscribe): Context consumer can be allowed to subscribe with a context management system by describing the requirements. The system will then return the results periodically or when an event occurs (threshold violation). In other terms, consumers can subscribe for a specific sensor or to an event.

B. *Characteristics of Context Information*

Several requirements have to be taken into account when modeling context information [15]:

*Heterogeneity and mobility:*Context data obtained from databases or digital libraries like geographic map data are often static. Many context-aware applications are also mobile or depend on mobile context information sources. This adds to the problem of heterogeneity as the context information provisioning must be adaptable to the changing environment. Also, location and spatial layout of the context information play important roles due to this requirement.

Relationships and dependencies: There exist various relationships between types of context information that have to be captured to ensure correct behavior of the applications. One such relationship is dependency whereby context information entities/facts may depend on other context information entities.

*Timeliness:*Context-aware applications may need access to past states and future states (prognosis). Therefore, timeliness (context histories) is another feature of context information that needs to be captured by context models.

*Imperfection:*Due to its dynamic and heterogeneous nature, context information may be of variable quality. In fact, it may even be incorrect. Most sensors feature an inherent inaccuracy and the sensed values age if the physical world changes, so that this inaccuracy increases over time. Thus, a good context modeling approach must take these problems into account to enable proper reasoning about context information changes to achieve appropriate adaptations for the application, and thus provide an experience for the user that is consistent with the physical world.

*Reasoning:*Context-aware applications use context information to evaluate whether there is a change to the user and/or to the environment situation; taking a decision whether any adaptation to that change is necessary often requires reasoning capabilities. Reasoning techniques can also be adopted to derive higher level context information.

IV. ISSUES WITH CONTEXT-AWARE MIDDLEWARE

With the birth of context-aware middleware many issues are came out, like security, balance of user control, threat to privacy of the user. In the following sections we examine the issues with the new middleware approach.

A. *Balance of user control*

In order to increase software autonomy, applications depend on context information to dynamically adapt their behavior to match the environment and user requirements [6]. Therefore, context-aware applications not only require middleware for distribution transparency of components, but also to support personalization and adaptation based on context-awareness. However, context-aware applications may not always adapt as the user expects, and may cause users to feel loss of control over the behavior of their application.

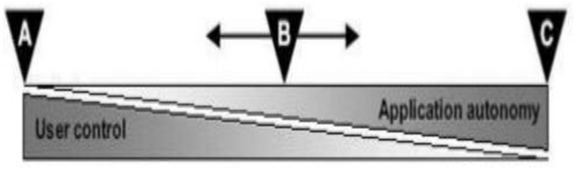


Fig 7. Continuum of user control versus software autonomy

However we can see in figure, at the leftmost end (A), users is given full control over application behavior, and applications have very little autonomy. Applications designed in this way are the most interactive. Conversely, at the other end (C), applications only require a small amount of user control. Applications can also occupy any intermediate position on the continuum (position B). The appropriate position along the continuum will be dictated by the user's needs, situation and expertise.

But, context-aware applications may not always adapt as the user expects, and may cause users to feel loss of control over the behavior of their applications. Thus, an autonomous context-aware system must provide mechanisms to strike a suitable balance between user control and software autonomy.

In traditional applications, the trade-off between user control and software autonomy has been fixed at design-time. In contrast, context-aware applications may need to adjust the balance of user control and software autonomy at run-time by adjusting the level of feedback to users and the amount of user input.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

B. *Privacy*

Privacy is the right of the individual to control his personal information by posing specific obligations on the subjects that process his data. Privacy sensitive information is available to pervasive service providers continuously making it difficult to protect it. Stealing sensitive information through analysis of traffic might be constrained with use of cryptography but it is still possible to glean some information like identity, location and activities. Under any situation it is important to ensure that users don't feel they are being spied on and using the services exposes them to unexpected threats and misuse of context and private information.

New technologies like RFID (Radio Frequency Identification) have on one hand made pervasive computing a reality but on the other hand have caused concerns about privacy as users are tracked when they move around. Laws such as the Patriot Act are also causing concerns that not only can hackers but also law enforcement agencies can intrude on the privacy of individuals. Protection of private and sensitive information is mandated under laws such as HIPAA (Health Insurance Portability and Accountability Act) and the European Union's Data Protection Directive, but their implementation in the Pervasive Computing realm is complicated ([16], [17]).

Privacy of location information deals with controlling access to the huge amount of sensitive information that might be generated when location systems continuously track users [18]. The user would not want to stop all access because some applications can use this information to provide critical services like public safety, transportation, emergency response, and disaster management [19], but there is a need for the user to be in control [18].

C. *Security*

When it comes to security, one size does not fit all [20]. Hence, the security architecture deployed should be able to provide different levels of security services based on system policy, context information, environmental situations, temporal circumstances, available resources, etc. Scenarios which require a higher-level of assurance or greater security may require users to interact with the security subsystem explicitly by, say, authenticating themselves using a variety of means to boost system's confidence.

Often, traditional security is somewhat static and context insensitive. Pervasive computing integrates context and situational information, transforming the computing environment into a sentient space. The security aspects of it are no exceptions. Security services should make extensive use of context information available. For example, access control decisions may depend on time or special circumstances. Context data can provide valuable information for intrusion detection mechanisms. The principal of "need to know" should be applied on temporal and situational basis. For instance, security policies should be able to change dynamically to limit the permissions to the times or situations when they are needed. However, viewing what the security policy might become in a particular time or under a particular situation should not be possible. In addition, there is a need to verify the authenticity and integrity of the context information acquired. This is sometimes necessary in order to thwart false context information obtained from rogue or malfunctioning sensors.

Pervasive computing environments can host hundreds or thousands of diverse devices. The security services should be able to scale to the "dust" of mobile and embedded devices available at some particular instance of time. In addition, the security services need to be able to support huge numbers of users with different roles and privileges, under different situational information.

V. Requirements of Current Middleware System

The middleware must address many of the requirements of traditional distributed systems, such as heterogeneity, mobility, scalability and tolerance for component failures. In addition, it must protect user's privacy. The large number of components that are present in context-aware systems introduces a requirement for straight- forward techniques for deploying, configuring and managing networks of sensors [21][22].

Support for heterogeneity: hardware components ranging from resource-poor sensors, actuators and mobile client devices to high-performance servers must be supported, as must a variety of networking interfaces and programming language.

Support for mobility: all components can be mobile and the communication protocols must therefore support appropriately flexible forms of routing. Context information may need to migrate with context-aware components.

Scalability: context processing components and communication protocol must perform adequately in very changing domains.

Support for privacy: flows of context information between the distributed components of a context-aware system must be controlling according to user's privacy needs and expectations.

Tolerance for component failures: sensors are likely to fail in the ordinary operation of a context-aware system; disconnection may also occur.

Ease of deployment and configuration: it must be easily deployed and configured to meet user and environmental requirements.

Dynamic reconfiguration: detecting changes in available resources and reallocating them or notify the application to change its behavior.

Adaptivity: the ability of a system to recognize unmet needs within its execution context and to adapt itself to meet those needs.

Asynchronous paradigm: decoupling the client and server components and delivering multicast messages.

VI. Conclusion

The current state-of-the-art of context-aware middleware explores quite different approaches to support pervasive and mobile computing based on context information. With the rapid development of information

technology, it is inevitable that the distributed mobile computing will evolve to pervasive computing. The shift to the pervasive computing paradigm brings forth new challenges of security and privacy.
Based on our survey we conclude that context-aware middleware is the only middleware that can meet the new requirements imposed by mobility as it can sense the user activity continuously. In our future work we propose to provide dynamic reconfiguration to the context-aware middleware so that it can adapt and reconfigure itself to enhance the behavior of the application. This will help designers building mobile applications to better utilize the scarce resources of mobile device like CPU power, battery life and memory.

REFERENCES

[1] Linthicum D., B2B Application Integration: e-Business-Enable Your Enterprise, 2001
[2] Umar A., e-Business and Distributed Systems Handbook: Middleware Module. 2nd ed. NGE Solutions, 2004.
[3] Salber D., Dey A., Abowd G., "The Context Toolkit: Aiding the Development of Context-Enabled Applications*", in Proceedings of ACM SIGHI Conference on Human Factors in Computing Systems (CHI-99), Pittsburgh, Pennsylvania, USA, 1999.
[4] Carpa L. et al.: Middleware for Mobile Computing, *InProceedings of the 8th Workshop on Hot Topics in Operating Systems*, Elmau, Germany, 2001.
[5] Abowd G. D., Dey A. K., Brown P. J., Davies N., Smith M., and Steggles P., "Towards a better understanding of context and context-awareness," in Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, ser. HUC '99.London, UK: Springer-Verlag, 1999, pp. 304–307.
[6] Marco B. and Leonardo B., "A survey about context-aware middleware", Retrieved from http://www.slideshare.net/LeoBruni/a-survey-about-contextaware-middleware, June 16, 2009.
[7] Henricksen K., "A framework for context-aware pervasive computing applications," Computer Science, School of Information Technology and Electrical Engineering, The University of Queensland, September 2003, http://henricksen.id.au/publications/phd-thesis.pdf [Accessed on:2012-01-05].
[8] Perera C., Zaslavsky A., Christen P. and Georgakopoulos D., "Context Aware Computing for The Internet of Things: A Survey", published in Communications Surveys & Tutorials, IEEE (Volume:16 , Issue: 1), 03 May 2013.
[9] Huebscher M. C., McCann J. A., "Adaptive middleware for contextaware applications in smarthomes", Published in Proceedings MPAC '04 Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, Pages 111 – 116.
[10] Bernardos A., Tarrio P., and Casar J., "A data fusion framework for context-aware mobile services," in Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on, aug. 2008, pp. 606 –613.
[11] Bikakis A., Patkos T., Antoniou G. and Plexousaki D., "A survey of semantics-based approaches for context reasoning in ambient intelligence," in Ambient Intelligence 2007 Workshops, M. M, F. A, and A. E, Eds., vol. 11. Springer-verlag berlin, 2008.
[12] Guan D., Yuan W., Lee S., and Lee Y.K., "Context selection and reasoning in ubiquitous computing," in Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on, oct. 2007, pp. 184 –187. [Online].
[13] Nurmi P. and Floréen P., "Reasoning in Context-Aware Systems." Position paper Department of Computer Science, University of Helsinki., Dec 2004.
[14] Hall D. and Llinas J., "An introduction to multisensor data fusion," Proceedings of the IEEE, vol. 85, no. 1, pp. 6 –23,jan 1997.
[15] Shehzad A., Ngo H.Q., Pham K.A., Lee S. Y., "Formal Modeling in Context Aware Systems", In Proceedings of the 1 st International Workshop on Modeling and Retrieval of Context (MRC'2004).
[16] Stanford V., "Pervasive health care applications face tough security challenges", *Pervasive Computing*, IEEE, Volume 1, Issue 2, April-June 2002, pp. 8-12.
[17] Jacobs A. R., Abowd G. D., "A Framework for comparing perspectives on privacy and pervasive technologies", *Pervasive Computing*, IEEE, Volume 2, Issue 4, Oct-Dec 2003, pp. 78-84
[18] Beresford A. R., Stajano F., "Location Privacy in Pervasive Computing", *Pervasive Computing*, IEEE, Volume 2, Issue 1, Jan-Mar2003,pp.46-55.
[19] Lee C.K., Lee W.C., Leung H.V., "Nearest Surrounder Search", IEEE International Conference on Data Engineering, April 2006.
[20] Campbell R., Al-Muhtadi J., Naldurg P., Sampemane G., and Dennis Mickunas M.,"Towards Security and Privacy for Pervasive Computing" ISSS'02 Proceedings of the 2002 Mext-NSF-JSPS international conference on Software security: theories and systems, pages 1-15.
[21] Hadim S., Al-Jaroodi J. and Mohamed N., "Trends in middleware for mobile ad hoc networks", *JCM*,1(4):11–21, 2006.
[22] Henricksen K., Indulska J., McFadden T. and Balasubramaniam S., "Middleware for distributed context-aware systems", In *International Symposiumon Distributed Objects and Applications (DOA*, pages 846–863. Springer, 2005

BIOGRAPHIES

Miss. Chitra Sahuis PG Scholar. She holds B.E in Computer Engineering and pursuing her M.Tech from ShriRamdeobaba College of Engineering and Management, Nagpur. Her areas of interests are Computer Networks, Context Aware Middleware, Pervasiveand Ubiquitous Computing.

Dr.D. S. Adane is currently working as Professor and Head, Information Technology Department, Shri Ramdeobaba College of Engineering and Management, Nagpur. He is PhD in Computer Science and Engineering and has over 22 years of experience in teaching and research. His areas of interest are Distributed and Mobile Computing, Wireless Sensor Networks, Mobile Agents and Network Security. He is also Life member of ISTE and Institution of Engineers India (MIE).