Study and Evaluation of Publish/Subscribe Paradigm in Context-Aware Middleware

Ehsaneh Moslem Rezvani and Reza Ravanmehr

Department of Computer Engineering Islamic Azad University, Central Tehran Branch, Tehran, Iran

*Corresponding Author’s E-mail: Ehs.rezvani@gmail.com

Abstract

There are several important reasons for the development of a context-aware middleware. The most important factor is the use of smart environment and devices, such as sensors that provide interaction with the environment. Users who intend to have mobility according to the new environment and its dynamic changes can also get the required information by changing the current communication mechanism. Moreover, the context-aware middleware using acquired information, solves the user’s requirements and adapts to the environmental changes. This paper, first presents a survey on the context and the context-aware middlewares and then provides a comprehensive study and analysis of the context-aware publish subscribe middlewares in different applications and environments.

Keywords: Context, Publish/Subscribe, Context-Aware Systems, Middleware

1. Introduction

Middleware is a layer of software that is placed between the application and the operating system. It uses the power of the system, integrates itself with it and has the important ability to support the development of distributed systems. Context-awareness can also be considered as one of the most appropriate characteristics in order to make connections between the communication participants with the supporting of context from their environment. Context-awareness can also be considered as one of the most appropriate characteristics in order to make connections between the communication participants with the support of which, context-aware middlewares have empowered their communication mechanism. To better understand how to use context and context-aware application, first we need to know the meaning of context. First [4], the context is referred to as locations, identities nearby people and objects and changes to those objects. In a similar manner [20], context is defined as location, identities the people around the users, the time of day, season, and temperature. Third [18], context is defined as the user’s location, environment, identity, and time. The fourth definition of context is referred to as an emotional state, focus of attention, location, and orientation, date and time of objects and people in the surrounding environments [1]. Finally, complete definition of the context is any information that can be used to characterize the situation of an entity. An entity could be a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves [2]. Context awareness is really the use of context for the preparation of information and the services that users require to perform a task, or according to the context, it is to receive demands. The context awareness allows users that are in the same location to communicate with each other without getting to know each other. To realize such communication we need a middleware. That kind of middleware which is applied within the static context and the homogeneous environment is referred to as the traditional middleware. However, according to today’s communication requirements and the various changes occur in it, we need a context-aware middleware that adapts with the user’s requirement, as well as with context changes, such as changes in the available resources. Considering the need to apply context-aware middlewares, we explain four context-aware middlewares and some
usages of them. With reference to these needs, in this paper we would be referring to four different types of context-aware middlewares as well as their benefits from context-aware publish/subscribe on other middleware. In section 4 the above-mentioned benefits would be discussed.

One of the characteristics of the publish-subscribed paradigm is to provide supporting for communicational systems, such as message passing, remote procedure calls, notifications, shared spaces, and message querying. This also has the power of decoupling in time, space and synchronization, provided by greater flexibility to be used for different kinds of communications. Nowadays, communications are often mobile and short or long time disconnections may also occur. To facilitate communication, we need a middleware to support time decoupling, which we do not have at the time of communication where both participants are active. In the context-aware publish/subscribe middleware, it is possible that generated events get sorted and sent to them while subscribers are active. Other characteristics of publish/subscribe is the way that events are presented. This could be based on the topic, content or type. Context-aware publish/subscribe middleware is based on the context.

In this paper, we explain the context-aware middleware and context-aware publish/subscribe middleware, how they work and discuss the use of them. After introduction, in section 2, we briefly explain different aspects of context-aware middleware and then we described and compare four well-known context-aware middlewares. The integration of publish/subscribe paradigm to the context-aware middleware is presented in section 3 together with study and analysis of different types of context-aware publish/subscribe middlewares. In section 4, context-aware publish/subscribe solution are analyzed based on different quality factors, and finally we finish the paper by concluding remarks in section 5.

2. Context-Aware Middleware

Context-awareness includes acquisition of the contextual information and arguments about context and behavior based on that current context. Middleware provides context-awareness to support each of these tasks. It also explains a common model for contexts so that all type of agents can use these contexts, and ensures that the various agents in the environment have a common semantic understanding in the contextual information. Distributed applications also need middleware in order to be able to adapt to environmental changes.

Context-aware middleware acquires an understanding maintaining that the environment can change the behavior according to contextual information and also it can help users to make the best decision according to the data obtained. Context-aware middleware delivered subscriber’s inquiries for information according to the location or user contexts [14-15].

Requirements for the implementation of the context-aware middleware, such as support of heterogeneity, scalability, support for mobility, support for privacy, traceability and control, tolerance for component failure, ease of deployment and configuration, dynamic reconfigurable, adaptability and asynchronous paradigm [14-15] should also be considered.

The most important aspects of context-aware middleware can be noted as environment, storage, reflection, quality, composition, migration, and compatibility [13]. These should be specified before the implementation of the middleware. The middleware has been evaluated in accordance to these aspects shown in Table 1. The following are brief descriptions of each of these aspects.

- **Environment**

  Context-aware middleware environment is divided into two parts: the middleware based on the infrastructure and the self-contained middleware. Middleware based on the infrastructure offers services needed by the middleware and applications. In self-contained middleware, communication devices does not rely on external services.

- **Storage**

  A context-aware data store is provided by some systems which order data on the basis of context information, permitting it to be repossessed in terms of some particular context parameters. For instance, file systems are provided by some systems where data is merely context-orientated. Centralized storage facilities get provided by other systems for context information, allowing applications to recover it.
• Quality
In the context-aware middleware, quality is related to the quality of services and the way to provide an appropriate and efficient resource.

• Combination
Component compositions regarding contextual events are performed by some middleware. Entities, for example, might work in double harness with all other entities in their vicinity, or composition might be altered if some context events occur.

• Migration
Some systems run entities migration. Mechanisms for migrating running codes are the ones that are merely provided when the application makes this decision, probably on the grounds of the context when other systems migrate entities mechanically on the basis of the context. This is closely connected to adaption, where diverse fragments of the systems adapts to the context.

• Adaptation
In the context-information being accessible, systems can adapt to alterations in the context. Dissimilar portions might all apply context in various ways, but most middleware systems do not utilize context-information on all of the system parts. Adaption to changes might occur in the middleware or in the applications.

  The most important features that need to be implemented in context-aware middleware are architectural style, location transparency, aspect-oriented decomposition, service discovery, fault tolerance, adaptability, and interoperability.

  For a better understanding, we will first describe the features and review context-aware middleware and in the following, we will compare them and conclude from them.

  Architecture style in middleware architecture is very important. Architecture style represents that there are several ways to use the components. Extensibility and flexibility in middlewares are mostly dependent on the architecture style, besides, architecture style affects adaptability in the middleware [23].

  Adaptability is the feature which refers to one of the most crucial functionalities needed within a context-aware environment. The ability to alter the behavior with regards to different environments is depicted by Adaptability. It can be divided into static (happening at the start-up or compile-time) or dynamic (taking place at the run-time) [23].

  Service Discovery is a crucial prerequisite in ubiquitous computing environment. It regulates how applications find other entities and how they can be found out by other entities [24].

  The reliability and safety parameters of any middleware architecture are determined by the fault tolerance. Applications get enabled by the fault-tolerant middleware to continue operating in the presence of faults [17].

  Another main aim of middlewares is to provide interoperability. By the use of this feature, for example, two various systems can be capable of exchanging information and as a result, they can get to improve the high usage of the exchanged information. In context-aware environments, mobile devices need to communicate with each other, but some context-aware middlewares, which lack the support of interoperability, do not offer this feature [24].

  Cross-cutting concerns are separated into modules referred to as aspect-oriented decompositions. Separation of cross-cutting concerns at the development-time, compile-time or run-time is allowed by the aspect-oriented decomposition. Non-functional behavior gets encapsulated by these aspects. Where safety and security concerns are vital, in real-time applications, matching with some context-aware middleware constructions that lend support aspect-oriented decompositions is necessary [23].

  An important characteristic, requiring client objects to precisely recognize a server object’s location while corresponding and seeking for services offered by that is location transparency [24].
Now, four context-aware middlewares are going to be discussed over. The reason for selecting these middlewares is their extensive use, and also the fact that each of these middlewares could be considered as a part of the flowing context, including computing environment, user environment, and physical environment [9]. Aura and Odin are included in the section of the computing environment. Middleware belongs to physical environment, and eventually publish/subscribe middleware is a part of the user environment.

2.1. Aura Middleware

One of the most important challenges of the ubiquitous computing is the adaptability of the architecture with resources that are dynamically changing. A framework which is made for ubiquitous computing is aura middleware. In fact, a task-oriented system for infrastructure environments which provides services for the management of tasks, applications, and contexts is Aura. Aura middleware has the following significant characteristics: firstly, user tasks become first-class entities being represented clearly and independently from a particular environment. Secondly, user tasks are characterized as alliances of abstract services. Thirdly, environments are prepared to self-monitor and renegotiate task maintenance in the presence of a wide range of capability and resource varieties [12].

Two of the Aura’s most vital capacities are shielding users from variations in resource availability and supporting users’ mobility. It means that when users are performing a task and due to the change of location, these tasks remain incomplete, Aura middleware by offering some special services from service providers makes this possible to continue the user’s working in the new environment. In the Figure1, the architecture of Aura is shown. By using the Aura’s architectural framework, two major problems in developing ubiquitous computing that supports the users’ mobility and adaptions with the dynamic resources in different environments are solved[12,7].

Aura can easily be used in situations where there is a need for mobility. For example, by watching movies on the internet, this middleware can be used in different environments. So that by moving users among different environments, available services can be detected and prepared to be applied by the users in order to, for example, watch movies online.

2.2. Middle Where: A Location-Aware Middleware

Capability, services and applications of ubiquitous computing are significantly elevated by location awareness. Besides, location awareness creates appropriate links between users and sources within the confinements of the environment. Technology and various ways of sensing technology provide extensive information regarding different locations in different formats and qualities. Context-aware middleware architecture Middle Where (shown in the Figure 2) makes combining different technologies possible in order to detect the position. By using these capabilities of the middleware and gathering comprehensive information regarding positions, positions should always be available [4].
Among the features that distinguish the Middle Where from other similar middleware, incorporation of multiple locations, sensing technology to obtain the real position of objects, handling the temporal nature of information about locations can be mentioned.

Hybrid location models show information in symbolic and coordinate, push and pull modes of interaction and also handle region-based and object-based locations that give the position of the certain element or elements in a particular area. Model of the world shows spatial relationships between mobile objects and their physical environments [4]. Middle Where represents a location in a hierarchical format, which is known as a GLOB. GLOBs can be represented in different formats of points, lines or polygons. For example, the table can be shown as Building1 / 3/338 / Desk1, meaning that Desk 1 is in Room 338, the 3rd floor is on Building 1.

![Figure 2. Middle Where middleware architecture [4]](image)

2.3. Context-Aware Odin Middleware

Odin is a middleware to provide mobile services and is provided to address the problems of the development of mobile services, service availability, scalability and availability, mobility and limited resources. Odin Context-aware middleware’s approach can be elaborated in two ways: First, context reasoning to prolong the time available and, thus, reducing the power of consumption, second, support context awareness that brings improved mobile services. Other characteristics of this middleware could be its three middleware-level adaptations that is distinct from other similar middlewares. In addition, Context acquisition gets support from Odin through both polling and subscription mechanisms and in this way, it offers a well-defined API to contribute to service-developers to plug in varied context-sources to middlewares simply and efficiently. Adaptation deals with the challenges of providing mobile services, including accessibility, service availability, and scalability i.e., the best use of available resources based on current operating conditions [3, 25]. Its architecture can be seen in Figure 3.
This middleware uses Bluetooth technology for nearby devices and Wi-Fi for mobile devices. It can also use 3G and LAN. This middleware can be used in telemedicine and also in the process of the development of the intelligent services, context information management in smart spaces and context-aware mobile applications.

2.4. Context-Aware Publish-Subscribe Middleware

Adapting well to the loosely coupled environment of allocated collaboration in large-scaled uses, the publish/subscribe interaction model is one of the best solutions. It owes this popularity to its simplicity and the quality of preparing its users’ desired information in the appropriate times [21].

Context-aware publish/subscribe middleware has three components [21]: publisher, broker, and subscriber. Publisher is the one which publishes information to the event services and subscribers who subscribe for that information can simply receive it from the event services. Its architecture can be seen in Figure 4. Publish/subscribe model allows the subscribers to connect to the network at any time they wish and also, can receive and produce information or become aware of new generated information and use them. If they do not desire to consume the information, generated information is not lost, instead it is stored in the space, known as shared space (or database) and the subscribers can make use of the generated information, after getting active.

The advantage of this middleware being compared to similar middlewares is in its decoupling, that is considered to make communication between publishers and subscribers more convenient. This feature facilitates access to the maximum use of the information, so that the subscribers, independent of time, space, and synchronization can use the already-mentioned information. According to this decoupling feature between publishers and subscribers and its advantages, this middleware has the ability to store information and this is the unique characteristic that separates this middleware from others.

Figure 3. High-level overview of Odin’s architecture [3]
Another advantage of this middleware is in the expression of the events of interest. In the way that subscribers always do not want to receive all the events; instead, they prefer to receive their selected ones. Thus, they can express an interest in events based on types, contents, and topics [21]. Content-based publish/subscribe approach, particularly in situations where several participants interact with each other, lead to inefficiencies in the routing messages from publishers to subscribers. To avoid this, it is better to use the context-aware publish/subscribe communication model. In this case, the useful content of the messages is not the only aspect that is relevant to the process of determining the information flow, but also the context in which this body of information is produced as well as its relation to the context of subscribers is relevant to determining the information flow.

As a good solution, using the advantages of content-based schema in context-aware interactions can be suggested. In this mode, both content-based and context-based communication pattern models are used in the process of routing protocols and the distributed context-aware publish/subscribe system [8] is created in this way. Therefore, all events get filtered based on the content, but sent according to the context (the current context of the user).

The publish/subscribe middleware along with its many advantages, by adding context, gets more powerful and thus, it can be used in multi-hop wireless networks. As other applications, the provision of health services, distributed environments, e-commerce applications, wireless sensors and so on.

2.5. Architectures Assessment

In the previous section, four context-aware middlewares and their uses were described. Each middleware is designed for a specific purpose and before designing, its properties, such as environment, storage, quality, composition, migration and adaption needs to be determined [23]. In Table 1, an overview of the described middleware is presented.

<table>
<thead>
<tr>
<th>Middleware</th>
<th>Environment</th>
<th>Storage</th>
<th>Quality</th>
<th>Composition</th>
<th>Migration</th>
<th>Adaption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Infrastructure</td>
<td>Self-Contained</td>
<td>Data</td>
<td>Context Resources</td>
<td>Context</td>
<td>Middleware Application</td>
</tr>
<tr>
<td>Aura</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Middle Where</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Odin</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pub/sub</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
In Table 2, the designed properties of the middleware is shown and in Table 3, a comparison between among all middlewares based on their non-functional features is presented.

### Table 2. Design features of Context-Aware Middlewares

<table>
<thead>
<tr>
<th>Middleware</th>
<th>Location Transparency</th>
<th>Aspect Oriented Demonstration</th>
<th>Architecture Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aura</td>
<td>No</td>
<td>No</td>
<td>Module (task manager, Environment Manager and Context Observer)</td>
</tr>
<tr>
<td>Middle Where</td>
<td>No</td>
<td>No</td>
<td>Layered (Provider Interface, Location and Service inference engine)</td>
</tr>
<tr>
<td>Odin</td>
<td>Yes</td>
<td>Yes</td>
<td>Layered (transport layer, message layer and service layer)</td>
</tr>
<tr>
<td>Pub/Sub</td>
<td>Yes</td>
<td>Yes</td>
<td>Distributed or centralized architecture module (publisher, broker, context manager and subscriber)</td>
</tr>
</tbody>
</table>

### Table 3. Non-functional requirements of the context-aware middlewares

<table>
<thead>
<tr>
<th>Middleware</th>
<th>Fault Tolerance</th>
<th>Interoperability</th>
<th>Service Discovery</th>
<th>Adaptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aura</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Middle Where</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Odin</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Pub/Sub</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3. **Analyze of the Context-Aware Publish/Subscribe Solution**

In section 2-4, we became aware of the nature of the context-aware publish/subscribe middleware and the power of this interactive schema in efficient communications between publishers and subscribers. It was also shown that context-aware publish/subscribe middleware has proved to be more influential comparing with other middlewares at hand. Besides, they happened to possess different communication patterns, such as the one based on the topic, type, and content. Also, in terms of having decoupling in time, space and synchronization, it mostly adapts with the needs of different users and applications.

Efficient communication has been established by context-aware publish/subscribe middleware by the use of content-based scheme and taking advantage of it, and also by adding context to the communication. This feature brings many advantages with the most important being as follows: preventing from publishing the useless information, easing of scalability, reducing communication requirements, decreasing communication costs, lowering traffic bandwidth, enhancing system performances and supporting many users. Due to the advantages of context-aware publish/subscribe middleware over others described, this middleware has been studied in different environments.

As we got familiar with the context, we recognized that this context could be location, services, and environmental information, such as the amount of ambient light or noise as well as some similar cases. Context in context-aware publish/subscribe middleware can be defined similarly. In this section, the middleware that had use for the context-aware publish/subscribe scheme and also the one of which the context is the location of participant parts will be discussed.
3.1. Context-Aware Publish-Subscribe in MANET

The network without infrastructure is mobile ad hoc network that is capable of self-configuring. It consists of some mobile devices that are connected through some wireless links. It is made up of mobile devices that are connected through wireless links. Within a MANET, each device as this freed omto move towards different directions. However, the network changes constantly due to the mobility of the nodes and thus, the final result is the creation of dissimilar directions among the nodes. In the case of the occurrence of a changeable communication, not being focused enough, lack of efficient network which can be implemented or even lack of reliability, the perfect solution is MANET. Moreover, when there is no communications infrastructure or the existing infrastructure are too expensive and not easy to be used, MANET can be of great value [11].

By the use of context-aware publish/subscribe model, publishers can restrict publish events by determining each event that belongs to a particular context. Similarly, the subscriber can also subscribe to receive the events that are relevant in a specified context or publications, events belonging to a particular context [6].

Points needed to be considered regarding mobile nodes in MANET include: Bringing Context into Publish-Subscribe, Context Specifications, Bringing Context into Events and Subscriptions, Persistent Events and Subscriptions, Contextual Relevance and Interest, Constraining sets of publishers and subscribers, Events, Subscriptions, and Matching [20].

In MANET, publish/subscribe model has been developed in a way that contextual information could be used in events and subscriptions. This improves the communication between publishers and subscribers, so that it significantly increases the capability of the system by necessary communications. By the contextual information, publishers and subscribers can control the diffusion of information and restriction of the receivers. This has reduced the unnecessary communication and prevented from forwarding useful messages.

3.2. Context-Aware Publish-Subscribe in VANET

VANET network is a special type of MANET networks in which nodes are in vehicle. In fact in this network, any vehicle can detect surrounding vehicles at any time and connect them to organize networks and thus, make necessary communications. Vehicles can communicate with one another autonomously. This leads to the creation of an unstructured wireless network.

No station or central node exists in VANET network to govern or control the network and it consists of mobile vehicles, having no fixed location and therefore, they do not play as the role of a router or an access point. Due to the mobility of the nodes in VANET, node context is one of the characteristics to be considered where a communication is taking place. Another characteristic of VANET is high partitioning possibility that causes frequent disconnection. In this situation, critical events would not be received [16]. To prevent this situation, context-aware publish/subscribe middleware makes it possible for the events to remain persistent in the network.

Publish/subscribe solutions for VANET network is divided into three solution categories, namely: geographic routing based solutions, proximity routing based solutions and solutions based on overlay network. In geographic routing based solutions, the network uses geographic routing to publish events from the publishers to the subscribers. In order for these operations to work correctly, two conditions must be met: nodes in the network need to equip a positioning system, such as GPS, and all events need to be relevant to a certain geographical area. Context-aware publish/subscribe solutions are of this category [16].

VANET is one of the main parts in intelligent transportation systems (ITS), and in recent years this has been given a lot of attention due to the direct relationship between its networks, vehicles, and traffic safety. VANET’s goal is to enable publishing the traffic and road information conditions for independent moving vehicles. Published event in VANET are used to improve the quality of driving in terms of time, distance, and safety. VANET network is an efficient network to disseminate warning messages through
vehicles on the network about the factors blocking the front of the road. Published events will be used to notify the vehicle for changing traffic conditions on the road so we could enjoy an efficient transportation system. The VANET can be used for safety, convenience, and commercial applications in intelligent transportsations.

3.3. Helferlein Middleware

New methodologies to improve distributed applications need to be studied with the advent of ubiquitous environments. In this environment, most devices are mobile and often change their locations. However, various components need to communicate and these communications should be content-driven and indirect. Meaning that, each event should not be routed directly and each component must identify its favorite event. This issue becomes more complex when most applications and components are context-dependent, and do not need to receive all the events and those events are important that relate to the context.

Context-aware publish/subscribe system provides suitable facilitates for efficient communications among various components in a distributed environment. Helferlein is a type of context-aware publish/subscribe middleware for intelligent distributed environments that attach the context properties to the events. By using the context properties, communications among the distributed events in the system include the least number of connections [22].

This middleware made scalability easier. It also made it possible to use components of middlewares, as in temporary series, without modifying the component.

In the Helferlein context-aware publish/subscribe middleware, development of distributed applications undergoes two phases: (a) the development of components (encapsulated pure functionality in a number of software components, which ensures reusability of components) and (b) the development of a set of distributed ensembles (communications among components through middlewares). One of the main characteristics of this middleware is its ability to track the state of all the necessary parts and to display any change in the state of the components. Meaning that the events are used to indicate state changes (If some changes happen to the value of a property, an event will be created showing this state change) [22].

By the use of Helferlein middleware, the two following problems have been solved: (1) how to enable event-based context-aware interactions among different parts within a dynamic joint and, (2) how to minimize the level of necessary communication needs [22]. The Helferlein middleware is used in environments, such as smart classrooms and intelligent and dynamic heterogeneous environments.

3.4. Pervaho Middleware

Concept of the awareness provided many advantages for mobile applications. Providing developers with sufficient tools to overcome the developments burden is an essential challenge to facilitate the proliferation of such applications. In order to be developed in an optimal way, mobile context-aware applications need suitable middleware services. Among the available middlewares, Pervaho[19], is an integrated middleware, provided specifically to develop and test context-aware applications. Pervaho offers location-based publish/subscribe services. This service has extended content-based publish/subscribe services through (1) adding location filters and 2) spreading introduction of the concept of the persistent publish. In the Pervaho platform, published events could be stable or unstable and also they might be related to the context. Furthermore, publications are associated with the context of interest.

Presenting an integrated open source platform with the goal of developing and testing mobile context-aware applications and running on small devices, such as smart phones or PDAs is the purpose of Pervaho middleware Context-aware applications are able to react within two sort of contexts, namely egocentric contexts and social contexts. Egocentric contexts indicate locally obtainable information, such as its location or its battery power using a device. Social contexts indicate information extracted from other devices situated in the network, such as their locations, their battery powers, or details in terms of the provided facilities [19].
Pervaho platform consists of both location-based publish/subscribe services and phone motion simulators. A communication service for mobile context-aware applications which is based on locations called LPSS. The existence of a context (location) in the matching of published events or subscriptions is the main difference between location-based publish/subscribe and classic types, such as topic-based or content-based ones is in.

Location-based communication is one of the most prominent parts in the mobile applications. In these applications, environment is dynamic and vast and nodes are often mobile. In this type of communication, participants have to be separated, for example, they are only allowed to communicate with each other anonymously. By the use of Location-aware Publish/subscribe Service (LPSS), this need can be met. LPSS can be known as a combination of topic and content-based publish/subscribe services. It can also be used in wired communications [19].

The location-aware applications could be used in voting services (to gather the votes of those participants who are in a specific area), they also could be used in searching services (uses by the participants to find out who they are in their range), news services (giving news to participants) and in similar cases.

3.5. LBS Middleware

Location-based services (LBS) has gained more attention recently. With advances in Internet technology and GPS, LBS system provides efficient variety of services related to the location of the users.

In [26], the LBS middleware based on publish/subscribe mechanism was present. This middleware enters granularity-refined LBS information due to the benefits of publish/subscribe system, such as an asynchronous, loosely-coupled, and multiplex communication mechanisms. Also, users can easily focus on their points of interest through combining position information and publish/subscribe paradigms. The positioning module enables the system to handle several scenarios in the vast environment and thus, system can easily add new positioning servers and dispatch its load of work to them by using load balancing mechanisms. System performance does not decrease under massive requests and this middleware can offer an efficient service to the users. This middleware can offer refined location-based services to several users in both indoor and outdoor environments. This middleware has three functions, positioning, subscribing and receiving messages, and finally displaying those messages that are considered as foundations for the developers to develop various types of LBS applications.

Among those technologies that can be used in this middleware are the user's fingerprint, Wi-Fi, GPS, and A-GPS. This middleware application is used for publishing location-based information by using publish/subscribe mechanisms and balancing the load of positioning servers. This middleware has the ability to manage a large number of sequenced requests and can perform them well. Therefore, in cases where there is possibility of sequenced requests, this middleware is very good and can offer refined location-based services to several users in both indoor and outdoor environments.

3.6. A Location-Based Publish/Subscribe Framework

With the advancement of mobile and sensing technology, users are able to obtain ubiquitous sensing data relevant to their locations and activities at any time and any place.

Users can subscribe to events of their interests in the sensing field and receive relevant data when unusual events occur. Providing efficient and reliable data delivery and subscription is very important, even though users may be disconnected or relocated.

In [10], a novel location-based publish/subscribe framework for mobile users (wireless sensors and mobile phones) is present for the users to subscribe for sensing data by simple ways through specifying the event types and target locations of their interests. The framework which is developed on a location-based Distributed Hash Table (DHT), being formed by a network of brokers is called a Location-Based Publish/Subscribe Framework. In addition to the above features, supporting users’ mobility, presenting
a reliable protocol for managing, handling users’ registration customer, and locating users’ relocations can be mentioned.

It can provide reliable data delivery without causing any data loss even when users are disconnected or moved to a new place. Whenever an event is detected by a wireless sensor, notification will be published along with the associated data to the corresponding subscribers.

The framework includes the subsequent characteristics: (1) distributed system architecture that is scalable and also is able to support heterogeneous devices, (2) a routing strategy that is efficient in terms of message delay and communication overheads, and (3) supporting the reliable data delivery to avoid data loss in disconnections or relocations due to the user’s mobility [10].

In this framework, mobile users can subscribe for sensing events through their associated brokers in specified target areas. Then, received events are forwarded to the corresponding brokers and finally, they get dispatched to the subscribers.

Among technologies that can be used in this framework are Wi-Fi, Bluetooth, ZigBee, and GPRS. The application of this middleware is in phone mobile networks and wireless sensor networks. This middleware ensures that events are delivered to the subscribers even if they have mobility or short-time or long-time disconnections without failure or any data loss. In cases where we have many users and also there is possibility of short-time or long-time disconnections, this middleware can store the data and forward the stored data to the users when they get active.

4. Evaluation and Comparison of Middlewares

In previous section, we got familiar with a few well-known context-aware publish/subscribe middleware in different environments as well as some advantages of using this middleware. By using this type middlewares, some limitations and difficulties of the traditional middlewares will be reduced and also these middlewares have been introduced as a perfect solution for the networks, having mobile nodes to communicate with each other as well as in cases where participants of the communications do not need to identify others.

The context-aware publish/subscribe middleware can be used in mobile networks, such as Mamet and VANET, so that by adding context to the events and subscription, publishing and receiving events get limited. Helferlein middleware provides the ability to trace the components, using this feature. Other components can behave and make decisions according to the changes in the situations of other components. Pervaho middleware can be used in context-aware application, for example, to connect users within a specific area. LBS middleware by the use of positioning modules has the ability to answer to the user positioning requests and by using load balancing mechanisms, positioning servers has the ability to respond to the users’ high-sequenced requests. By using the location-aware publish/subscribe paradigms, information of users’ interest is delivered without any data loss even if in the case of short or long-time disconnections.

Each reviewed middleware has specific applications in specific domains. The common feature among these middlewares is their context. Every six middlewares are location-aware and have mobile nodes. In Table 4, we review general aspects of these middlewares.

| Table 4. Overview of Context-Aware Publish/Subscribe Middlewares |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Middlewares     | Context         | Node Type       | Node Mobility   | Participant   | Identification  | Application      |
| Context-aware P/S in MANET | Location of node | Laptop – Smart phone - Tablet | Yes | No | Communication among mobile nodes |
| Context-aware P/S in VANET | Location of node | Any type of vehicles | Yes | No | Communication among mobile nodes, specifically car |

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication among mobile nodes</td>
</tr>
<tr>
<td>Communication among mobile nodes, specifically car</td>
</tr>
</tbody>
</table>
In Table 5, the investigated middlewares with suitable features for a context-aware middleware will be evaluated. The investigated parameters are features that are considered important for designing and evaluating suitable context-aware middlewares.

<table>
<thead>
<tr>
<th>Appropriate Middleware Feature</th>
<th>Support for Heterogeneous</th>
<th>Support for Mobility</th>
<th>Scalability</th>
<th>Support for Privacy</th>
<th>Traceability and Control</th>
<th>Tolerance for Component Failure</th>
<th>Ease of Deployment and Configuration</th>
<th>Dynamic Reconfiguration</th>
<th>Adaptability</th>
<th>Asynchronous Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context-Aware P/S in MANET</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Context-Aware P/S in VANET</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Helferlein</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Pervaho</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBS P/S</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LB P/S</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Conclusion**

The absolute need for exchanging information among distributed and heterogeneous nodes, the necessity to develop faster, reduction of efforts, coordination problems, software reusing and key patterns are key factors leading to the emergence of the middleware architectures and more advanced capabilities in this area. Although the existing conventional middlewares are useful, they are not suitable due to such reasons as the need for environmental information to communicate. For this reason, there is a need for context-aware middleware. In addition to the ability to provide the previous needs, it could provide new communication’s needs as well; needs such as adapting to the users’ mobility as well as discovering users’ positions. For this reason, in this paper different context-aware middlewares have been discussed and evaluated. According to the survey on middlewares, we found that the context-aware publish/subscribe middlewares provide more appropriate communications and are more efficient than the other ones. In addition to providing a context-based mechanism along with all its advantages, a context-aware publish/subscribe middleware uses contexts to filter the delivered information. This leads to improved functionality and increased efficiency. Thus, those middlewares utilizing this communicational paradigm have been discussed in section 3 and have been evaluated in section 4.

**References**


