Architectural Styles for Opportunistic Mobile Communication: Requirements and Design Patterns

Davy Preuveneers
Department of Computer Science
Katholieke Universteit Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium

Ansar-Ul-Haque Yasar
Department of Computer Science
Katholieke Universteit Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium

Yolande Berbers
Department of Computer Science
Katholieke Universteit Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium

{davy.preuveneers,ansarulhaque.yasar,yolande.berbers}@cs.kuleuven.be

ABSTRACT

Mobile computing has been in the research epicenter for several decades. We have seen drastic shifts with mobile computing systems and wireless ad hoc networks dynamically adapting to create co-operating nodes that provide the right services at the right time. One aspect of such systems that has been poorly addressed is the complexity of developing applications for mobile users that require group interactions in large scale networks, especially between people unfamiliar with one another but with similar goals. Such systems need to know the type of social interaction, the context of all participants and nodes in the network, and a proper architectural design to implement scalability up to city wide networks. In this paper, we analyze which architectural styles and design patterns are the best suited the implement the various interaction requirements. We address scalability for interactions in large-scale networks from a software architectural perspective and focus on event-driven service oriented architectures to make the design of context-aware systems for large-scale mobile interactions a less complex task.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, selection process.*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*data sharing.*; H.2.11 [**Software Engineering**]: Software Architectures —*patterns.*

General Terms

Performance, Design.

Keywords

Context-Awareness, Publish-Subscribe, Event-Driven Architecture, Scalability.

1. INTRODUCTION

The growing presence of WiFi and 3G wireless Internet access and sensor network technologies will give further rise to the mobile computing paradigm. In a mobile computing environment information and intelligent services will be embedded in the environment around us, and large amounts of data will circulate in order to create smart and proactive environments that will significantly enhance both the work and leisure experiences of people.

The problem that we want to address is the lack of proper architectural support for developing applications that initiate collaborations in social groups within mobile networks, where the members of the group are unfamiliar with one another but share the same goals. In this paper we propose a context-driven reference architecture based on a combination of several architectural styles that are effective for large-scale networks for mobile opportunistic communication. The issues related to the location awareness and privacy also arise while dealing with the large-scale networks, but have not been addressed in detail in this paper as a lot of research [6] has already been conducted in this domain. 'Large-scale interactions' is a vague term that needs some clarification. The concept is three-fold and assumes communication between at least two participants. The 'large-scale' attribute can refer to (1) the distance between the participants, (2) the number of participants, and (3) the number of messages transmitted between the participants. To further explain how these aspects of interaction can become a concern, we will considered them in terms two real life based scenarios related to mobile human computer interaction; (i) *Geo-caching in a urban-based environment* and (ii) *Mobile social networks at the airport.*

A first scenario deals with mobile gaming and intertainment, and explores the interactions within a geo-caching game:

Geo-caching can played in an intelligent environment by two to tens of thousands people. Geo-caching can be played within a small building or across the globe by quite a large number of people thus involving a large-scale network. It is an entertaining adventure game for GPS users. It is a good way to take advantage of the wonderful features and capability of a GPS device by participating in the cache hunt. The basic idea behind the cache is that different people or organizations setup caches all over the world and share the GPS coordi-

nates over the internet. GPS users can then use the location coordinates to find the caches. Once found, a cache may provide the visitor with a wide variety of rewards known as the traditional cache or coordinates to another cache known as the multi-cache or offset cache. In an intelligent environment the game can be made even more exciting by hiding online puzzles that need to be solved to obtain clues or hints.

In this scenario the 'large-scale' is mainly in terms of the number of participants and the distance between the participants as geo-caching can be played with any number of people located across the globe. Different types of interactions in this scenario either in the form of a query or a message are listed as under:

1. *Where are my team members?*
2. *Who can help me with this puzzle?*
3. *Send a clue to all players in this region.*
4. *How many people are participating in the game?*
5. *Which caches are currently active?*
6. *How far I am from a particular cache?*

The second scenario deals with mobile social networks which is another emerging field in the domain of mobile human computer interactions.

Let us assume that there is a large Software Engineering conference in Leuven in which more than 500 people are participating from across the globe. Each of the participants uses a different mode of transportation but we will only consider the air route. The participants from several countries land at the Brussels International Airport which is quite a large and busy airport in Europe. It served approximately 48000 passengers everyday with different flights in 2007 [1]. Some of the participants are planning to take a taxi to from the airport to the conference centre, some of them want to meet the persons speaking the same language or taking the same flight to the same destination. The participants could share a taxi with someone else going to the same place and in order to help in reducing the cost. But on an airport with such a huge traffic of people it is nearly impossible to find the person with the same interest. It would be really nice if all the persons boarding the same flight get notified about each other or if everyone receives information about all the flights departing and arriving at the airport or if they get notified about all the facilities in their vicinity, etc.

Travellers can take part in different types of interactions, either with other travellers or with people offering services in the environment. Some of the interactions that the traveller may initiate at the airport in this scenario include:

1. *Where can I find a taxi? Is there one available?*
2. *Who is interested in sharing a taxi to this hotel?*
3. *Who is attending also the conference?*
4. *Who speaks my language?*
5. *Who is returning to the airport around that time?*
6. *When and at which gate will the boarding start?*

Context-awareness is an important factor required in many applications for ubiquitous and mobile computing and many of these applications will run on computing devices that vary from a traditional personal computer to smart handheld devices like mobile phones, PDAs and other smart devices with computing and communication capabilities. These computing devices are connected to various wired and wireless networks to assist us in our daily activities. The earlier work in the area of context-awareness has mainly focused on the location area networked (LAN) devices. However the networks are growing larger with time from a local area network resident in a small office and home environment to global wide area network (WAN) that cover whole cities or even larger regions. Developing context-aware applications that deal with such large networks is a complex undertaking. We need guidelines for developing applications dealing with large-scale interactions. This increases the demand for a context-driven software architecture that can serve as a reference to build context-aware applications for large-scale interactions. There are several issues identified earlier [8] while dealing with the large-scale networks, like scalability, automatic discovery, fault tolerance and the heterogeneity and dynamicity of the context sources. We will retreat these issues in the architectural styles.

The need for the proper architectural support for large-scale interactions arises while considering the scenarios mentioned earlier. A context-driven architecture for large-scale networked interactions can help in devising a solution that enables a minimal quality of service by adapting to the needs of the system and the users. For example, the need for an immediate event notification to each of the participants about a new clue in geo-caching or another conference participant interested in sharing a taxi illustrate the need for proper mobile data management in large scale ad hoc networks. The information should be filtered and only relevant information should be transmitted in both the scenarios for certain reasons later discussed in the paper. Evaluation of the architecture based on the scenarios is presented later in this paper. While reviewing the main architectural styles we will also describe how they relate to the scenarios and also to other examples in terms of large-scale interactions.

From above scenarios, we distill a list of requirements for context-aware applications and software architectures that target large-scale network scalability with a large number of participants in section 2. We describe a number of relevant architectural styles in section 3. Section 4 shows how various architectural styles have been integrated into a single software architecture that can serve as a reference design well suited for large-scale networks. Evaluation of the scenarios within the proposed suitable architectural style is presented in section 5. We describe related work in section 6. We conclude and propose future work in section 7.

2. REQUIREMENTS FOR LARGE-SCALE CONTEXT-AWARE INTERACTION

Fueled by ongoing developments in mobile and ubiquitous computing, context-awareness has attracted a lot of attention over the recent years to achieve non-intrusive behavior on personal and multi-user systems in highly dynamic environments. For example, a boarding warning notification service at the airport will inform the mobile phone that boarding will start soon. Although the network may not be that large-scale, the service needs to address from all the travellers at the airport those with a particular flight

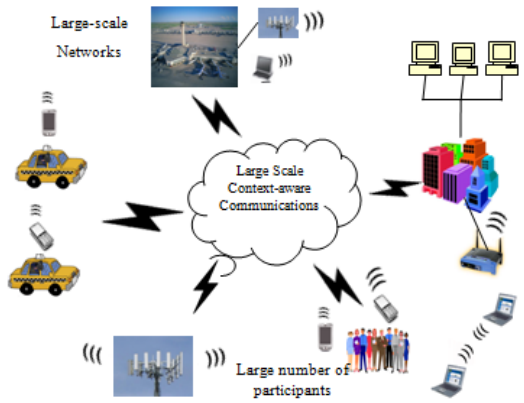

Figure 1: Event-driven coordination in city-wide network.

number and that are not already only located at their designated gate. It is clear that there are several requirements to implement large-scale interactions. In this section we will describe the most important requirements for scalability in terms of network size, number of participants, and number of messages.

2.1 Event-driven interactions

Event-driven coordination is required for notification by the devices / users working in the large-scale networks for scalability. If we take the scenario of sharing a taxi mentioned in the previous section, all the persons at the airport could get notified in case of an arriving taxi event. For example, travelers could get detailed information about the bus, train and the taxi going into the direction of the conference centre, or the person should get notified as soon as there is another person interested in sharing a taxi to the same place. The communication is mediated by the infrastructure. It typically involves asynchronous communication, there is no channel topology and the relaying of the information is based on the content and may address multiple receivers at the same time.

In event-driven coordination processes raise events and an event is delivered to the processes which subscribed their interest to its specific type of events. The events can be created whatever the current state of the system. An event plays a role in the system until it is implicitly removed from the system. An event has restricted visibility and suitable in the large-scale environment i.e. it is only and exactly available to the subscribed processes / users / devices.

2.2 Context-awareness

The term ”context” has been defined manifold by various researchers [2] according to the situation. As we will focus on large-scale interactions in terms of networks and number of participants, we have taken into consideration how the term context has been defined for these application areas. Context is defined by [6] to be the set of environmental states and interactions that either determines an application’s behavior or in which an application event occurs as that is of interest to the user. It refers to the idea that the computing devices can both sense and react based on their environment and situation.

To address the network scalability aspect, we rely on techniques that have already proven their usefulness for large distributed systems for many years. Two of these techniques include replication and caching. Replication is used for availability and caching for performance and scalability. A combination of both is indeed a requirement to support the large-scale networks to provide synchronization and availability of information.

The objective of replication is to improve the quality of service when people require continuous access to a particular service from various places in the large-scale network. Therefore, we need a group of processes that handle incoming events that trigger the replication of data or computation. In the case of data replication the processes maintain the stored service data, reply to synchronization requests. When we replicate computation (e.g. context-aware services) the only goal is to achieve a higher fault-tolerance to network disruptions by connecting to the nearest replicated service.

Caching of shared context data over multiple nodes can help in improving the performance of context-aware services. Not only is this one of the major goals in a context-aware large-scale network, but it also helps in fault-tolerance in case of a failure. Both caching and replication provides synchronization of information and services across multiple nodes in a large-scale network. This will ensure that access to services and information is available whenever required.

2.3 Scalability in mobile networks

In a large-scale network the number of nodes, users and information flow can be enormous. In this situation a user might only be interested in relevant piece of information according to its contextual needs and the requirements.

Small scale devices cannot handle large amounts of events or data. Context-aware filtering achieves considerable reductions in the transmission of data. In context-aware filtering, the system takes into account whether the data that is currently being received from sensors/nodes are relevant within the estimated range for the user’s current contextual position and if it can be used or processed by the user’s device. Thus context-aware filtering allows the flow of only relevant information for the users / devices.

For example, context-aware filtering can play a vital role to address the location-awareness of the nodes in a large-scale network by the help of the past information. For example, in the taxi scenario, we only would like to address nodes with a particular destination. In the case of the geo-caching scenario mentioned earlier, only members of a team need to be made aware of any solution or hint one of its members may have found. With multiple teams playing in multiple games, each of the teams involved in a particular game should only be notified with the information concerning to them. If we refer to Figure 1 all the information being available in a large-scale city-wide network should not be transmitted to every node in the city rather it should be filtered and sent accordingly to places, users or devices that consider the in-

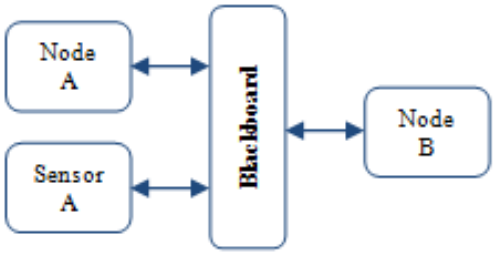

Figure 2: Blackboard Architecture.

formation relevant for their own purposes.

2.4 Relevance feedback

The previous requirements are needed, but they are not foolproof. For example, caching is irrelevant at places where nobody uses the information. In highly dynamic environments, context-aware filtering and event-driven notification may not be sufficient.

Implicit or explicit relevance feedback is a requirement that helps to decide whether information that is being distributed is actually being used. Also, feedback may also help to see which requests could not being addressed due to lack of information. It intends to improve a user's query and assist retrieval of information relevant to a user's request. The main idea is to take the returned results of the current request into account and to decide that whether the obtained information is relevant or not. If not, the filtering may need to be adapted or the relevant information should be replicated at other places. As such, relevance feedback is another very important aspect that helps to improve the quality of service for a user in a large-scale network where the information is in enormous quantity and only relevant information is required.

3. ARCHITECTURAL STYLES

There are several architectural styles / patterns suitable for specific large scalability scenarios. We investigated several architectural styles because there is no single architectural style that can handle all the issues mentioned earlier in section 1, so we propose a combination of architectural styles for large-scale interactions.

3.1 Blackboard architecture

Blackboard models have been widely used as a particular kind of problem solving model [4]. The blackboard model allows multiple independent nodes to share information in a central store known as the blackboard. Figure 2 represents the blackboard model.

For example, the blackboard architecture could be used as a shared message board in the taxi scenario. A person could use it to inform other travelers about his whereabouts and travel destinations. It could also be used as a traffic information system to inform any other related information.

However, blackboard models are difficult to test, no good solution is guaranteed, difficulty of establishing good control

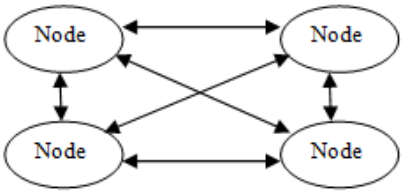

Figure 3: Peer-to-Peer Architecture.

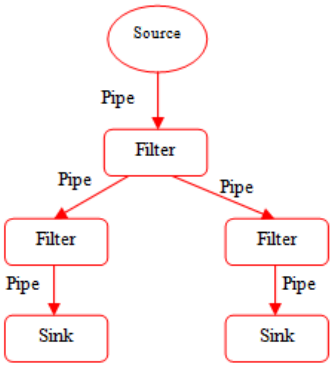

Figure 4: Pipes and filters Architecture.

strategy and no support for parallelism [4]. There is a fair chance of a bottle neck as well in the case of enormous number of nodes. Thus, the blackboard architectural pattern alone is not suited for supporting a large-scale network.

3.2 Peer-to-peer architecture

Another architectural pattern that is used to share information is known as the peer-to-peer architecture as shown in Figure 3.

In a peer-to-peer architectural style each of the participating nodes collaborates in such a manner that they have a direct communication channel in between. This type of architectural style is not very well suited for large-scale networks as the performance degrades with the increase in the participating nodes.

3.3 Pipes and filters architecture

Pipes and filters is another type of the architectural style as shown in Figure 4. This is a well known architectural style for the data stream oriented processing [6]. Each processing step is implemented as a filter. In pipes and filters each node has a set of inputs and outputs. A component reads a stream of data as input produces a stream of data as its output [6]. The stream of information is passed through the pipes in between the filters and the sink. The data after passing through a series of pipes and filters reaches the sink.

This architectural style can be used in the geo-caching scenario. For example, a member of team A in Belgium has solved a puzzle and gained a hint that is relevant for his colleague(s) in France. This means that the information flow should be filtered on place and addressees.

Figure 5: Service-oriented Architecture.

Some of the advantages of the pipes and filters architecture are [4]; (i) filters can be exchanged flexibly, (ii) filters can be reused, (iii) parallel processing support and (iv) filters are isolated from each other. Some of the disadvantages are [4]; (i) sharing state information is expensive, (ii) parallel processing in real sense cannot be achieved as some of the filters may need all the inputs before producing an output and (iii) there is an overhead in the data transformation. Thus, making the architecture not well suited to support the large-scale networks.

3.4 Service oriented architecture

Service-oriented architecture is another popular type of architectural style as shown in Figure 5. In the service-oriented architecture the functionality is divided into smaller units known as services and distributed over a network. The services communicate with each other by passing data from one service to another.

Service-oriented architecture has loosely coupled interactions i.e. the services are invoked independent of their technology and location, has one-to-one communication where one specific service is invoked by one consumer at a time and is synchronous.

3.5 Event driven architecture

Event driven architecture is another important type of architectural styles. It deals with the production, detection, consumption and reaction of events and it complements the service-oriented architecture. An event can be a significant change in the state. It is shown in Figure 6.

Event driven architecture consists of event producers and event consumers. Event producers and consumers are subscribed to an event manager. Whenever an event manager receives an event from the producer the manager forwards the event to consumer. Event driven architecture uses messaging also known as store-and-forward, to communicate among two or more processes. A major advantage of event driven architecture over service-oriented architecture that it has decoupled interactions in which event publishers are not aware of the existence of the event subscribers. Moreover, it provides many-to-many communication in an asynchronous way which makes it very well suited for notification applications.

4. ARCHITECTURE FOR LARGE-SCALE INTERACTIONS

A combination of event driven, service-oriented, pipes and filters and blackboard architectural styles can result in an architectural pattern which is well suited for the large-scale interactions. These three architectural styles have already been explained in the previous section.

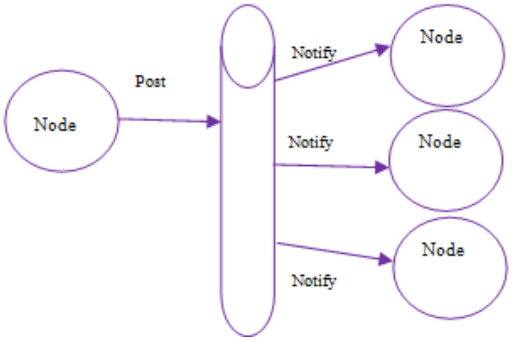

Figure 6: Event Driven Architecture.

The combination of the architectural styles will enable the asynchronous and synchronous communication, interactions between one-to-one, one-to-many and many-to-many communication, location-awareness and privacy in large-scale networks. The architectural style is shown in Figure 7. The peer-to-peer architectural style is not explicitly shown in this figure as a way for nodes to communication with one another but it could serve the purpose of relaying events to other infrastructures.

In this architectural style the Location-aware node is serving as a location based server to which all the other nodes publish their location information and the same information can be used by the other nodes. This concept is related to the blackboard architectural style. There is a privacy management node which is also acting as a privacy server. This server will not reveal the identity of a particular node to another node interested in communication until both the parties accept the communication. Platform for Privacy Preferences (P3P) is the suggested [7] solution to be used for handling the privacy issue. The communication is bidirectional where a node can request a service from a particular node through message passing and one service can be called by many subscribers.

5. EVALUATION

As the main topic of this paper is about large-scale interactions we evaluate the proposed architecture in terms of scalability and flexibility. Therefore, we have chosen not to evaluate the proposed architecture in terms of performance or network throughput but rather in terms of design and simplicity. The evaluation will be carried out from a qualitative perspective rather than from a quantitative perspective and this with the scenarios in mind.

5.1 Evaluating the Urban-based gaming scenario

Urban based gaming: Geo-caching can be played world wide as explained earlier in the previous section 2.1. Let's assume that 500 people in groups of 5 from all over the globe are playing for a multi-cache finally located in La Plagne, France. All of them are equipped with GPS enabled PDA's / cell phones. Traditionally the participants are unaware of the total participants in a particular cache and the cache is always some physical entity.

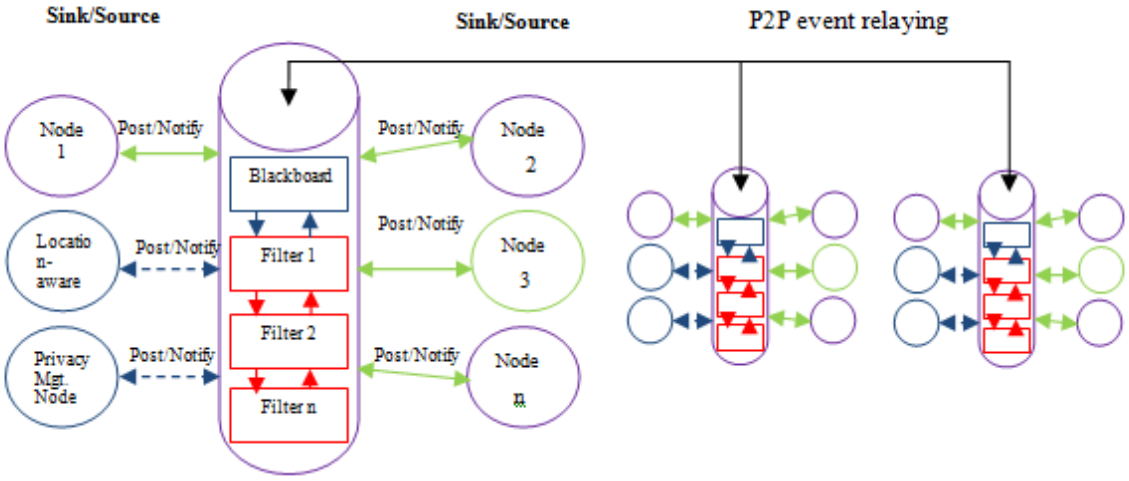

Figure 7: Context-driven Architecture for Large-scale Interaction - a combination of Blackboard/SOA/EDA/Pipes and filters.

But using our context-driven architecture for large-scale networks all the participants are notified about total number of participants looking for a particular cache, when a new participant joins or an old participant leaves a particular cache. The cache can also be either a physical or virtual entity. For example the participants have to find such a spot near some ski resort where they get notified of either the next coordinates or the virtual cache.

Only the participants playing as a team gets notifications about the location information of all the team members. If one of the team members get a cache / hint it is automatically being transmitted to other team members for information or action. Our context-driven architecture supports multi cache geo-caching played among a large group of people in new fashion.

5.2 Evaluating the Mobile social networks scenario

The second scenario is also an interesting scenario for information sharing at Brussels International Airport. The participants for the conference in Leuven arrive at the airport. All of them have scheduled their conference plans. The ones planned to take a taxi from the airport to the conference venue; a message gets posted automatically according to the context anonymously over the network to the other participants with the same interest. All the participants going to the same conference and planning to take a taxi at the airport gets notified. If one or more participants are interested in sharing a taxi then as they accepts the message all the interested participants gets notified about each other's identity information. Information about a nearest meeting point is also sent automatically for all the interested participants. The information about the facilities available at the airport is being sent to all the interested persons waiting at the airport or the flight information is being transmitted over to everyone at the airport. If someone wants to meet the person speaking the same language or taking the same flight he/she must be notified about it upon request.

Our context-driven architecture for large-scale networks supports the scenario for taxi sharing. It can easily handle the issues of context-driven messaging between the people with common interests at a particular place. It also caters the privacy and location-awareness issues.

5.3 Applying other architectural styles

There are many other architectural styles that have not been discussed in this paper. The reason for this is that they may have not matured yet to be used in real life scenarios. For example, two of these styles are search-oriented architecture and shared nothing architecture.

Search-oriented architecture is a new architecture for the search engines. In it the data tier may be replaced by another tier which contains a search engine and search engine index which is queries in-place of the database management system. The search engine itself crawls the relational database management systems along with conventional data sources such as web pages or conventional file systems. This architecture increases the response time of the system like a search engine.

Shared nothing architecture is a distributed computing architecture where each node is independent and self-sufficient and there is no single point of conflict across the system. Shared nothing architecture is currently popular in web development due to its scalability which is also an important factor in the domain of large-scale networks.

6. RELATED WORK

In [8] the authors discuss the concern of having a large-scale multitude of context sources that continuously publish that contextual information. Another concern that they investigated for context-awareness in wide area environments is the presence of even more clients that search for and consume

this context information. Many protocols do not address the unique challenges of such environments. The authors summarize scalability, fault tolerance, heterogeneity, dynamicity and automated discovery as key challenges for large scale interactions.

The pipe-and-filter architectural style is also discussed in [6] to design and implement a large-scale context fusion network. They have demonstrated the effectiveness of this architectural pattern with various applications for smart spaces and emergency response.

In [7] the authors discuss location management system to gather process and manage location information from a variety of physical and virtual location sensors. They discuss scalability in terms of a large numbers of sensors and a large number of clients. They also address distribution of location information through the network.

In [10] the authors discuss large-scale networks and the use of data replication as a solution for greater scalability in wide area networks. They base their research on top of content addressable networks to achieve the centralized object locating and routing. The authors also propose analytical approach to examine their system techniques in the absence of actual large-scale deployments.

The service-oriented architectural style is also discussed in [9] to facilitate service discovery composition and verification by taking into consideration the requesters context to enhance the precision of request service match making.

The use of event driven middleware for context-awareness in mobile computing has been discussed in [5]. The authors propose an event model that provides a highly composable event notification framework that uses multiple levels of environments monitors to provide a complex setup of composite events.

The importance of caching context information has also been addressed in [3]. As disconnections between nodes in large-scale networks may occur due to nodes mobility, the authors discuss smart caching algorithms that improve the traditional methods for distributed systems by using various kinds of meta-data.

7. CONCLUSION AND FUTURE WORK

In this paper we have proposed an architectural design for context-aware systems that target large-scale interactions. We have discussed how large-scale interactions can reflect itself in terms of number of participants, distance between participants and size of the network. In order to cover these challenging concerns we have studied and analyzed various architectural styles with respect to these scalability aspects. We finally proposed an architectural style that integrates four different architectural styles with a proven track record in the software engineering domain. Combined they offer the required flexibility to implement context-aware applications in four various mobile human computer interaction scenarios.

For further research into this domain we are planning to look into further upcoming architectures like search-oriented and shared nothing architecture. The distinction between the architectures are not clear so we will further investigate that how these architectures can map onto our reference architecture proposed earlier. After a detailed study we might enhance our reference architecture with some interesting features of the studied architectures.

We are also planning to work on a modeling tool support for high design without dealing with the implementation issues. This will enable us to design before we can implement. Later on we can see how these modeling tools simplify development for the large-scale networks.

8. REFERENCES

[1] Brussels airport, 2008.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[3] M. Anandarajah, J. Indulska, and R. Robinson. Caching context information in pervasive systems. In *MDS '06: Proceedings of the 3rd international Middleware doctoral symposium*, page 1, New York, NY, USA, 2006. ACM.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[5] . A. T. S. Chan, S.-N. Chuang, J. Cao, and H.-V. Leong. An event-driven middleware for mobile context awareness. In *The Computer Journal*, 2004.

[6] G. Chen, M. Li, and D. Kotz. Design and implementation of a largescale context fusion network. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, 2004.

[7] J. Indulska, T. McFadden, M. Kind, and K. Henricksen. Scalable location management for context-aware systems, 2003.

[8] Y. Liu and K. Connelly. Towards wide area context-aware environments. In *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, page 616, Washington, DC, USA, 2006. IEEE Computer Society.

[9] S. J. H. Yang, B. C. W. Lan, and J.-Y. Chung. A new approach for context aware soa. In *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 438–443, Washington, DC, USA, 2005. IEEE Computer Society.

[10] B. ZHAO, A. JOSEPH, and J. KUBIATOWICZ. Localityaware mechanisms for large-scale networks, 2002.