Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficient and Dynamic Event-Based Middleware For Location-Aware Mobile Application

D. Ganesh[*]
Assistant Professor,
IT Dept., SVNE, Tirupati

Y. Karthik
M.Tech (SE) student of IT,
SVNE, Tirupati

I.KumaraSwamy
Assistant Professor
EEE Dept., SVNE, Tirupati

Abstract— **Now a day's mobile application is playing a prominent role. With the combination of middleware, these are developing widely. The main contribution is the event based programming model to develop mobile applications due to its inherent support for loose coupling between the components required by mobile application. Existing middleware that supports the event based programming for location aware mobile application for which highly mobile components come together dynamically to collaborate at some location for that we introduce some of the techniques including location- independent announcement and subscription coupled with location dependent filtering and event delivery that can be used by event- based middleware to support such collaboration. We are including quality of service (QOS) in terms of event delivery latency. We intend to exploit the concept of proximity introduced here as the basis for performing admission control to allocate the necessary communication resources for timely event delivery with a dynamically varying of population of mobile components.**

Keywords— **Distributed system, middleware, publish subscribe, event based communication, mobile computing, collaborative and location-aware applications, wireless ad hoc networks, Bayeux**

I. INTRODUCTION

The widespread development in mobile application to support application with guaranteed QOS requirements in terms of event delivery latency is generally recognized as being the next major advance in the information technology industry. Both mobility and middleware represents the key enabling technologies of underlying the vision of communication paradigms for mobile computing.

The event-based communication model represents an emerging paradigm for middleware that asynchronously inter connects the components that comprise an application in a potentially distributed and heterogeneous environment,[5] and has recently become widely used in application areas such as large-scale internet services and mobile programming environments . Event based communication is well suited to address the requirements of mobile computing domain. Mobile computing environments can utilize either the infrastructure or the ad hoc network model devices that interact through IEEE 802.11b-based [1] services. Several middleware services utilizing the event-based communication model have been developed thus for by both the industry and academia. Most of these assume that the application components comprising an application are stationary and that a fixed network models, related to the dynamic reconfiguration of the network topology.

Here we present STEAM (Scalable Timed Event and Mobility),[1] an event –based middleware service that has been designed for mobile computing domain specifically, it is intended for IEEE 802.11-based wireless local area network (WLAN) utilizing the ad hoc network model. We also realized several prototypical collaborative application scenarios derived from relevant areas, including search and rescue gaming,[7] and especially transportation to evaluate the proposed techniques. The evaluation demonstrates the feasibility of accommodating representative scenarios from the target category of application and illustrates the trade-off of this support by assessing the cost of location- based event dissemination as well as the latency imposed by location-dependent event delivery in STEAM.

We envisage steam being utilized by collaborative applications in various domains including indoor and outdoor smart environments, augmented reality, and traffic management. In a traffic management application scenario, application components may represent mobile objects including car, buses, fire engines, and ambulances as well as objects with a fixed location, such as traffic signals and lights. When with in close proximity, such components may interact using STEAM in order to exchange information on current traffic situation. As a simple example, an ambulance might disseminate its location to the vehicle travelling in front of it in order to have them yield the right of way. In general, inter-vehicle communication may contribute the better driver awareness of nearby hazards and is likely to lead to safer driving.

This paper extends a previous paper [1]. The paper is structured as follows: After a discussion of STEAM Architecture Overview II, Section III provides information about Related Work, Section IV present the LOCATION AWARE EVENT-BASED MIDDLEWARE, and section V Location-Independent Announcement and Subscription, and section VI Location-Based Event Filtering, and section VII proposed system is Latency, and section VIII Concludes this paper.

II. STEAM Architecture Overview

The STEAM event-based middleware has a number of important differences from other events services that support mobility

- STEAM assumes an ad hoc network model supporting very dynamic coupling between application components.
- The architecture of STEAM is inherently distributed. The middleware is exclusively collocated with the application components and does not rely on the presence of ant infrastructure.
- Application components are location aware .geographical location information is provided by location information is provided by a location service and used to deliver events at the specific location where they are relevant.

The STEAM middleware is fully distributed over some physical machines as the components that comprise a collaborative application. This implies that the middleware located on every machine that has identical capabilities allowing its components either to initiate or respond to communication STEAM middleware architecture contains neither centralized components, such as lookup and naming services, nor the kind of intermediate components that are used by other event services to propagate event notification from event producers to event consumers. Generally, dedicated machines that are part of the event service infrastructure are used to host such components in order to ensure that they are accessible to all application components in a system at any time. However this approach is imperatival in ad hoc environments due to lack of infrastructure and the possibility of network partition.

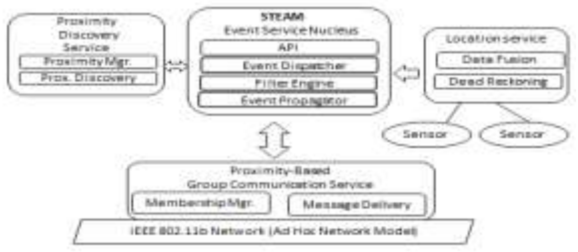

Fig. 1. STEAM architecture over view

The design of the STEAM architecture Fig.1 is motivated by the hypothesis that there are applications in which mobile components are more likely to interact once they are in close proximity. This means that the closer event consumers are located to a producer the more likely they are to be interested in the events that it produces. Significantly, this implies that events are relevant within a certain geographical area surrounding a producer. For example, in augmented reality games players are interested in the status of game objects or indeed other players, only when they are within close proximity. An example from the traffic management domain might be a crashed car disseminating an accident notification. Approaching vehicles are interested in receiving these events only when located within a certain range of the car.

We argue that the STEAM architecture and our approach to distributed event notification filtering helps to improve system scalability by omitting centralized components and by bounding the propagation of subscription information and event notification. This reduces the use of communication and computation resources which are typically scare in mobile environments. In general, distributed event filtering limits the number of filters being applied at a particular location and balances the computational load of filters matching between the physical machines in a system. The number of producer side filter is independent of the potentially large number of subscribers and the number of consumer side filters depends solely on the number of local subscribers. As a result, filter evaluation time can be bounded for the events disseminated in a particular scope.

III. Related Work

Recently some authors have begun to address distinct requirements of collaborative mobile application or of supporting event-based communication in ad hoc networks characterized by the absence of shared infrastructure. Eg. Application components using ad hoc networks cannot rely on the use of access point when discovering peers in order to establish connections to them .Event messages can neither be routed through access point nor rely on the presence of intermediate components that may reply on the presence of intermediate components that may apply event filters or enforce nonfunctional attributes such as ordering polices and delivery deadlines.

JEDI—It allows Nomadic application components to produce or consume events by connecting to a logically centralized event dispatcher that has global knowledge of all subscription requests and events. JEDI provides a distributed implementation of the events dispatcher consisting of a set of dispatching servers that is interconnected through a fixed network. Nomadic entities may move using the moveOut operation disconnects the entity from its current dispatching server and moveIn operation allowing it to move to another location to connect the dispatch server.

Elvin4—Represent event-based systems that support mobility through the use of a proxy server maintaining a permanent connection to the event servers on behalf of nomadic clients components. [3]The proxy server stores events while a client is temporarily disconnected and clients can specify a time to live for each subscription to prevent large numbers of events being stored indefinitely .Clients must explicitly connect to proxy server using a URL and must reconnect to the same proxy server each time they reconnect to the event system.

Rebeca—It allows nomadic clients to access a network of event routing brokers through local brokers. Local Brokers act as access points and allow clients to disconnect at the network broker to which they wish to relocate in a way similar to the approach described. Rebeca also promotes of location awareness. Eg: Describing the rooms in a house,[5] the places in a city, or the (coarse-grained) coordinates of GPS system.

Topss—Supports location awareness by extending its centralized filtering engine with a location matching engine. Location information can be expressed as latitude/longitude/altitude tuples and the location-matching engine receives periodic updates of the location of mobile entities. [7] Eg: Topss has been used for a friend-finder application in which mobile users specify a mobile friend about whom they wish to be notified when in closed proximity.[8]

IV. LOCATION-AWARE EVENT-BASED MIDDLEWARE

Event-based middleware to support pervasive and mobile applications in which collaboration between nearby entities is intrinsic must deal with the increased complexity that arises from a potentially large number of interacting entities, from their geographical dispersion, and from the spontaneously changing connections between them. Mobile entities that comprise this style of application characteristically move together and apart over time. Sets of such entities typically come together at a certain location to communicate and collaborate, move apart, and then come together with other entities at a different location to collaborate there. Hence, these entities are more likely to interact when they are in close proximity. For example, a vehicle is interested in receiving emergency vehicle warnings from an ambulance only when the ambulance is within close proximity.

Event Types and Proximities-- An implicit event-based programming model, i.e., one that does not make the present of other entities or event brokers explicit to application programmers [2], is naturally suited for applications in ad hoc environments. It allows producers to publish events of specific event types and consumers to subscribe to events of a particular type rather than having to subscribe at another entity or at an intermediate, as is required by peer-based and mediator-based event models [2]. Producers may publish events of several event types and consumers may subscribe to one or more event types. To accommodate collaborative applications, we propose an implicit event model that supports geographical scopes allowing producers to explicitly disseminate events to nearby consumers. Producers associate the type of event they intend to generate, or raise, with a geographical area, called the proximity, within which events of this type are to be disseminated.

Consumers can receive events of some type if (and only if) they are located inside a proximity in which events of this type are being raised. For example, an ambulance may define a proximity, whose size may depend on its speed and prevailing road conditions, for its emergency vehicle warning events. Other vehicles will only receive these events when located within this proximity relative to the ambulance.

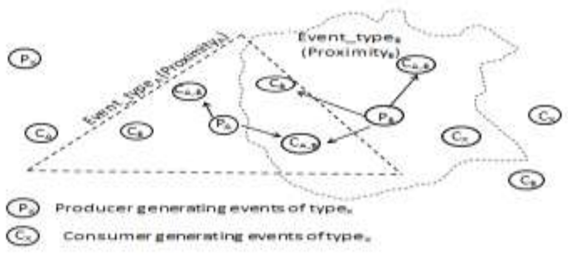
Event_type$_A$ (Proximity$_A$)
Event_type$_B$
(Proximity$_B$)
Producer generating events of type$_x$
Consumer generating events of type$_x$

Fig. 2. Supporting collaboration using event types and proximities

Proximities may be of arbitrary shape and may be defined as nested and overlapping areas. Nesting allows a large proximity to contain a smaller proximity subdividing the large area. Fig. 2 depicts two event types associated with overlapping proximities of different shape and illustrates that multiple consumers may reside inside a proximity. Proximity$_A$ and Proximity$_B$ have been defined for Event¬_type$_A$ and Event_type$_B$, respectively. Consumers that have subscribed will be delivered these events if they reside inside the appropriate proximity. Note that consumers located inside these areas but which are only interested in other event types will not be delivered events of either type.

V. Location-Independent Announcement and Subscription

Location-independent announcement allows producers to advertise their events and have these advertisements persist while moving location. As summarized in Fig. 3, a producer using location-independent announcement specifies an event type/proximity pair to associate a specific event type with certain proximity. That producer may then generate events of the announced type until it is unannounced and have them delivered to interested consumers within the proximity. Likewise, consumers use location-independent subscription to subscribe to events allowing them to receive events of interest whenever they move into a proximity in which such events are being generated

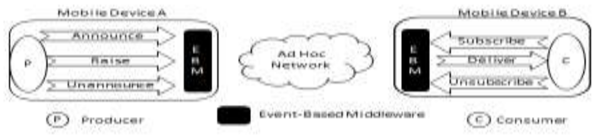

Fig. 3. Location-independent announcement and subscription.

Announcements specify both the functional and non-functional attributes of the events to be generated by some producer. The definition below shows that an event type consists of a subject and content representing its functional attributes, as well as of a self-describing attribute list representing its non-functional attributes.

Event Type = {Subject, Content, Attribute_List}

Location-Dependent Event Delivery— Event propagation is location dependent in that events generated by a particular producer are only delivered by consumers currently residing in the appropriate geographical area. Producers announce proximities to specify the locations at which their events are relevant. Consumers discover these areas of interest and subsequently deliver events at the locations where they are relevant. Mobile (and stationary) consumers transparently discover the proximities and ultimately the events of interest that are available at their current location regardless of the dynamics of the producers, i.e., whenever they enter a proximity or a proximity (attached to some mobile producer) arrives at their location. Consumers then deliver these events for as long as they reside inside the proximity.

VI. Location-Based Event Filtering

An event system consists of a potentially large number of producers [3], [4], [5], all of which produce events containing different information. As a result, the number of events propagated in an event-based system may be very large. However, a particular consumer may only be interested in a subset of the events propagated in the system or even within its current locality. Event filters provide a means to control the propagation of events. Ideally, filters enable a particular consumer to receive only the exact set of events in which it is interested. Events are matched against the filters and are only delivered to consumers that are interested in them, i.e., for which the matching produced a positive result.

Distributing Location-Based Filters--Location-based event filtering allows an application to specify multiple event filters, each of which may apply to a different attribute of a specific event. Such filters may be combined and a particular event is only delivered to a consumer if all filters match. For example, considering that applications often consist of more consumers than producers [6], [5], applying the filters defined by many consumers on a single (producer) node, may result in significant computational load for that node.

Defining Location-Based Filters---Location-based event filtering supports three classes of event filters: subject filters, content filters, and proximity filters Subject filters match the subject of events and allow a consumer to specify the event type in which it is interested.

Subject Filter= {Subject}

Filter Term= {Content Parameter Name,
Operator, Value}
Content Filter=
{(Conjunctive | Disjunctive),
Filter Term,[Filter Term], ...}

Proximity Filter= {(Stationary | Mobile),
Area (Shape, Dimensions,
Reference Point), Naval}

VII. LATENCY

Latency is mostly implemented in Subscription coupled with Location-dependent Event Delivery and the Location based Event Filtering. Better latency is provided through **Bayeux** is a protocol for transporting asynchronous messages (primarily over HTTP), with low latency between a web server and web clients.

Purpose (Bayeux):

The primary purpose of Bayeux is to support responsive bidirectional interactions between web clients, for example using AJAX, and the web server.

Bayeux is a protocol for transporting asynchronous messages (primarily over HTTP), with low latency between a web server and a web client. The messages are routed via named channels and can be delivered:

- server to client
- client to server
- client to client (via the server)

By default, publish subscribe routing semantics are applied to the channels. Delivery of asynchronous messages from the server to a web client is often described as server-push.

The combination of server push techniques with an Ajax web application has been called Comet.

CometD is a project by the Dojo Foundation to provide multiple implementation of the Bayeux protocol in several programming languages.

Bayeux seeks to reduce the complexity of developing Comet web applications by allowing implementors to more easily interoperate, to solve common message distribution and routing problems, and to provide mechanisms for incremental improvements and extensions.

VIII. Conclusion

The techniques provide the basis for supporting a wide range of mobile applications, proposed work remains for this to support applications with guaranteed quality of service requirements (QOS) in terms of event-delivery latency. The main intend to exploit the concept of proximity is here as the basis for performing admission control to allocate the necessary communication resources for timely event delivery within a dynamically varying population of mobile components

REFERENCES

[1] R. Meier, "Event-Based Middleware for Collaborative Ad Hoc Applications," PhD thesis, Dept. of Computer Science, Trinity College, Univ. of Dublin, Sept. 2003.

[2] B.P. Crow, I. Widjaja, J.G. Kim, and P.T. Sakai, "IEEE 802.11 Wireless Local Area Networks," IEEE Comm. Magazine, vol. 35, no. 9, pp. 116-126, Sept. 1997

[3] P. Sutton, R. Arkins, and B. Segall, "Supporting Disconnectedness—Transparent Information Delivery for Mobile and Invisible Computing," Proc. IEEE Int'l Symp. Cluster Computing and the Grid, pp. 277-285, 2001

[4] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," ACM Trans. Computer Systems, vol. 19, pp. 283-331, 2001.

[5] G. Mu¨ hl, L. Fiege, and P.R. Pietzuch, Distributed Event-Based Systems. Springer-Verlag, 2006

[6] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri, "Generic Support for Distributed Applications," Computer, vol. 33, no. 3, pp. 68-76, Mar. 2000

[7] G. Cugola, E.D. Nitto, and A. Fuggetta, "The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS," IEEE Trans. Software Eng., vol. 27, no. 9, pp. 827-850, Sept. 2001.

[8] I. Burcea, H.-A. Jacobsen, E.d. Lara, V. Muthusamy, and M.Petrovic, "Disconnected Operation in Publish/Subscribe Middleware," Proc. IEEE Int'l Conf. Mobile Data Management, pp. 39-50, 2004.

[9] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," Proc. Second ACM Int'l Workshop Data Eng. Wireless and Mobile Access, pp. 27-34, 2001.

[10] R. Meier, "Communication Paradigms for Mobile Computing," ACM SIGMOBILE Mobile Computing and Comm. Rev., vol. 6, pp. 56-58, 2002.S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

Authors Bibilography

Mr. D. Ganesh received his B.Tech degree in Information Technology from JNT University, Hyderabad in 2006 and M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University in 2010.During the period 2006-07 he worked as Assistant Professor in Information Technology department at AITS, Rajampet, India. Since 2007, he is working as Assistant Professor in IT Department at Sree Vidyanikethan Engineering College, Tirupati, India. He has Published 8 papers in national and International conferences. His current research interests are computer networks, wireless networks,Object oriented design and unified modeling. He is a member of ISTE, CSI.

Mr. Y.Karthik received his B.Tech degree in Information Technology from JNT University, Anantapur in 2010 and Persuing M.Tech in Software Engineering from Sree Vidyanikethan Engineering College (Autonomous) Tirupati in 2011-2012.