Volume 3, No. 10, October 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

IMPLEMENTATION OF EVENT BASED PROGRAMMING ON LOCATION AWARE MOBILE APPLICATIONS

Mannam Chandra Sekhar[*1] and Mudiganti Vijaya Bhaskar[2]
[*1]M Tech Student , Gokul Institute of Technology and Science, Bobbili, India
[2]Asst. Professor , Dept of CSE , Gokul Institute of Technology and Science, Bobbili, India
chsekhar48@gmail.com

Abstract: The programming paradigms and middleware architectures are designed to support the development of mobile applications as they are became more widespread and increasingly very essential. For the development of mobile applications, the event-based programming paradigm is a strong candidate as it gives congenital platform for the loose coupling between components required by mobile applications. Although existing middleware supports the event-based programming paradigm, it is not well accepted to give support for location-aware mobile applications in which intensely mobile components come together dynamically to collaborate at some location. In this paper we present a number of approaches which involves location-independent announcement and subscription coupled with location dependent filtering and event delivery which can be used by event-based middleware. This paper discusses how these approaches have been put into action in STEAM which is an event-based middleware and it is a fully decentralized architecture that is clearly suits to deploy in ad hoc network environments.

Keywords: Distributed systems, publish subscribe, event-based communication, mobile computing, collaborative and location-aware applications.

INTRODUCTION

Emerging pervasive and mobile computing applications comprise large numbers of interacting components distributed over large geographical areas. Examples include context aware intelligent transportation systems and city-wide information systems. Middleware to support such applications must deal with the increased complexity that arises from such scale, from the geographical dispersion of components, and from the spontaneously changing connections between components.

Such mobile applications can be characterized as collaborative in the sense that mobile entities use a wireless network to interact with other mobile entities that have come together at some common location. Examples might include tourists visiting the same site or vehicles traveling in the same direction. Having come together in some area, collaborative entities establish connections with other collaborative entities dynamically, temporarily forming a group that has a common goal. The members of such a group may even travel together for a period of time, as in the case of a group of tourists coming together and deciding to participate in a guided tour or a group of vehicles traveling in the same direction forming a convoy to improve driver safety and reduce fuel consumption. Although these collaborative applications may use infrastructure networks, they will often use ad hoc networks since these are immediately deployable in arbitrary environments and support communication without the need for a separate infrastructure.

For pervasive and mobile computing as well as sentient computing, this collaborative style of application allows loosely coupled, highly mobile components to communicate and collaborate in a spontaneous manner anywhere and at any time. In many cases, such applications will be deployed in situations where wireless network infrastructure might be available. For example, museums may deploy wireless local area networks and proposals exist for large-scale deployment of wireless access infrastructure along roads such as the Vehicle Infrastructure Integration initiative in the United States. However, we argue that such applications cannot rely on the presence of such infrastructure: Tourist attractions such as national parks or archaeological sites are unlikely to have such infrastructure and cost mitigates against ubiquitous wireless infrastructure being deployed on every road. Moreover, there are many collaborative mobile applications that will never be able to avail of such infrastructure such as coordination of Unmanned Aerial Vehicles (UAVs) or autonomous military vehicles deployed in hostile environments.

In principle, event-based communication is well suited to such mobile applications since it naturally accommodates a dynamically changing population of interacting entities and the dynamic reconfiguration of the connections between them. Event-based communication supports asynchronous interconnections between components and is particularly useful where communication relationships among components are dynamically and frequently reconfigured during the lifetimes of the entities. The event-based communication model supports a one-to many or many-to-many communication pattern that allows one or more entities to react to a change in the state of another entity. Event notifications, or simply events, contain the data representing the change to the state of the sending entity. They are propagated from the generating entities, called the producers, to the receiving entities, called the consumers. Events typically have a name and may have a set of typed attributes whose specific values describe the specific change to the producer's state. A particular consumer may only be interested in a subset of the events produced in the system. Event filters provide a means to control the propagation of events.

Ideally, filters enable a particular consumer to specify the exact set of events in which it is interested. Essentially, a consumer's event filters are matched against the events received by the middleware and only events for which the matching produces a positive result are subsequently delivered to the consumer application.

RECENT RESEARCH

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 move Out operation disconnects the entity from its current dispatching server and move In operation allowing it to move to another location to connect the dispatch server.

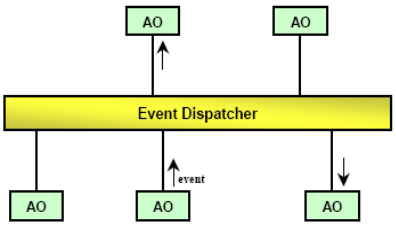

Figure 2.1: Overview of JEDI architecture

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.

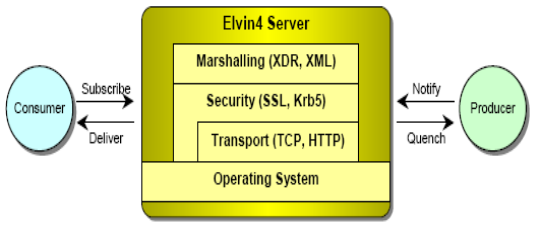

Figure 2.2: Overview of Elvin4 architecture

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, 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. 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.

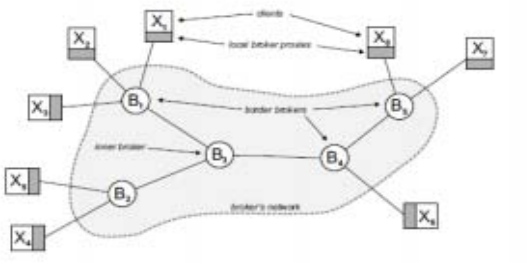

Figure 2.3: Overview of Rebeca architecture

MOTIVATIONS AND CONTRIBUTIONS

Existing event-based middleware for wireless networks has mainly focused on nomadic applications. These applications are characterized by the fact that mobile entities make use of the wireless network primarily to connect to a fixed network infrastructure but may suffer periods of disconnection while moving between points of connectivity.

Existing Computing devices are becoming more pervasive and the dependence on the information delivered through these devices is increasing. Due to these trends, users expect access to information on multiple devices at various geographic locations at any time, but Users may disconnect when network connectivity is absent or to conserve battery life. Therefore, support for disconnected operation is essential for information dissemination applications that support mobile users.

Existing problem of mobility from the viewpoint of the event-based paradigm and the two separate flavors of mobility identified. While physical mobility is tied to the notion of rebinding a client to different brokers and can be implemented transparently, logical mobility refers to a certain form of location awareness offering a client a fine-grained control over notification delivery in the form of location-dependent filters. But problems concerning the combination of mobility and pub/sub infrastructures remain. In existing system the disconnection while entities move from one access point to another, is handled whereas relatively little work has addressed the distinct requirements of collaborative mobile applications, especially those that use ad hoc networks. Existing system only identifies but does not classify the factors that affect the performance of

Publish/ Subscribe system that supports client mobility. Existing system have high cost associated with disconnection operation. There is lack in formalization of mobility algorithms for distributed Publish/ Subscribe system. No proper middleware to run distributed application on a variety of hardware and operating system. Problems concerning the combination of mobility and Pub/ Sub infrastructure not addressed.

For event systems to support collaborative mobile applications, they must enable collocated producers and consumers to be able to discover each other. Producers need to be able to advertise the events they intend to generate independently of their location and consumers must be able to subscribe to events persistently. Consumers must also be able to discover events of interest and to eventually deliver them at the relevant locations.

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 we use proximity i.e., location.

In this paper, we present a number of techniques that can be used by event-based middleware to support collaboration in location-aware mobile applications including location-independent announcement and subscription, location-based event filtering, and location-dependent delivery of events, Inherently distributed event service.

Location- independent announcement and subscription:

Location-independent announcement allows producers to advertise the (types of) events that they produce and have these advertisements persist while moving location. Likewise, consumers use location-independent subscription to subscribe to events allowing them to receive events of interest wherever they move. Such announcements and subscriptions are persistent in that they apply transparently to all locations independently of the specific location at which they have been issued and are vital to enabling the delivery of subsequently disseminated events at any location and while entities are moving.

For example, an ambulance providing an emergency vehicle warning service while rushing to an accident site may use a location-independent announcement to persistently advertise this service while responding to an emergency call. Other vehicles may use a location-independent subscription to subscribe to this service, enabling them to receive emergency vehicle warning events every time their respective journeys intersect that of an ambulance.

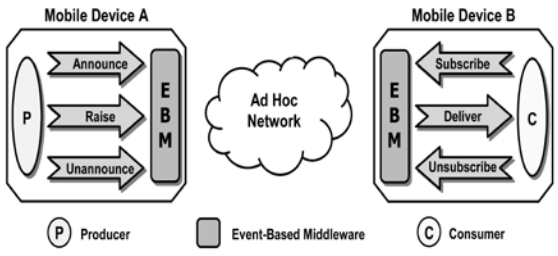

Figure 3.1: Location-independent announcement and subscription

Location-based event filtering:

Event filters define the specific subset of events in which a consumer is interested. Our approach to event filtering allows producers and consumers to define location-based filters that may use the actual entity location when applying a filter. Producers may define filters that describe a geographical area surrounding their location. These filters bound the geographical scope in which events are to be disseminated and move location with a migrating producer.

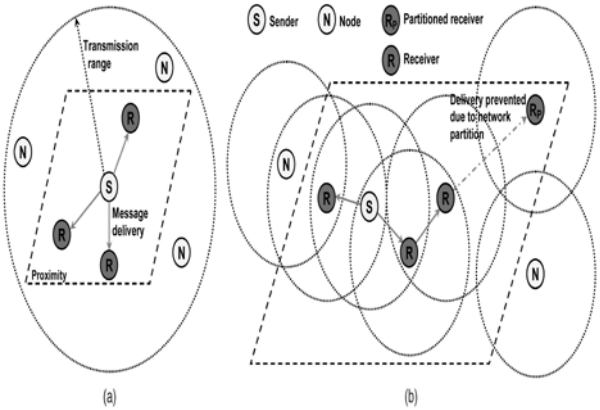

Figure: 3.2.Location-based event dissemination. (a) Single-hop event dissemination. (b) Multihop event dissemination

Location-dependent delivery of events:

Event propagation is location dependent in that events generated by a particular producer will only be delivered by consumers currently residing in a specified geographical area. Consumers may deliver a particular set of events at some location and subsequently deliver a different set of events at another location; they may deliver events generated by one producer and then deliver events generated by another producer at a different location.

Inherently distributed event service:

Event-based middleware for collaborative applications should enable entities that have come together at a certain location to communicate and collaborate through wireless connections even in the absence of any local network infrastructure, in particular by supporting ad hoc as well as infrastructure networks. Due to the characteristics of ad hoc networks, such an event service must be inherently distributed since it cannot rely on any service infrastructure. It cannot depend on logically centralized or intermediate components that are typically hosted by such an infrastructure. For example, it cannot rely on a well-known intermediate event broker to connect producers and consumers as has been proposed by SIENA to support nomadic applications. Moreover, the characteristics of collaborative applications, where entities come together to collaborate, move apart, and then come together with other entities at a different location, preclude dependency on broker nodes interconnecting such locations of collaboration across an ad hoc network. For example, it cannot rely on dynamically elected cooperating directories as have been proposed to support scalable service discovery for service oriented architectures based on ad hoc networks. Hence, event-based middleware for collaborative applications, employ concepts that can support such an inherent distribution, including event types and proximities, instead of centralized components.

THE STEAM EVENT SERVICE

The STEAM event-based middleware implements the techniques introduced in the previous section using group communication. STEAM provides location-aware event dissemination for collaborative pervasive and mobile applications running on mobile devices that interact through IEEE 802.11b-based ad hoc wireless local area networks .. Depending on the application areas in which they are used, such portable computing devices may range from handheld devices, such as personal digital assistants, to notebook computers. This section outlines the architecture and most important implementation techniques used by STEAM.

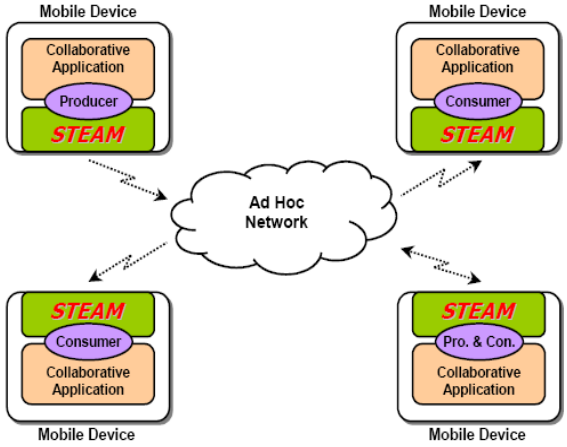

Figure 4.1: STEAM Event model

Inherently Distributed Service Architecture:

The STEAM event service is based on an inherently distributed architecture in which the middleware is exclusively collocated with the application components and does not depend on any separate centralized or intermediate components. As illustrated in Fig. 3, the architecture essentially consists of four key components that reflect the main features of the event service. The Event Service Nucleus (ESN) implements STEAM's application programming interface and therefore is explicitly exposed to applications. The event service nucleus can be regarded as STEAM's central component since it interconnects the remaining components and because it provides a filter engine that applies and maintains the various event filters that producers and consumers may define. The event service nucleus exploits a Proximity-based Group Communication Service (PGCS) to disseminate events depending on the locations of the relevant producers to consumers. The Proximity Discovery Service (PDS) provides the means for potentially mobile entities to persistentlyannounce and subscribe to events and eventually to discover events of interest. The discovery mechanism uses location information to map proximities to the subscriptions of its consumers and as a result, manages the proximity groups that are relevant at its current location. This implies that the proximity discovery service is responsible for maintaining a consistent notion of the relationship between proximity groups and subscriptions at any given time while considering the migration of entities and indeed of proximities.

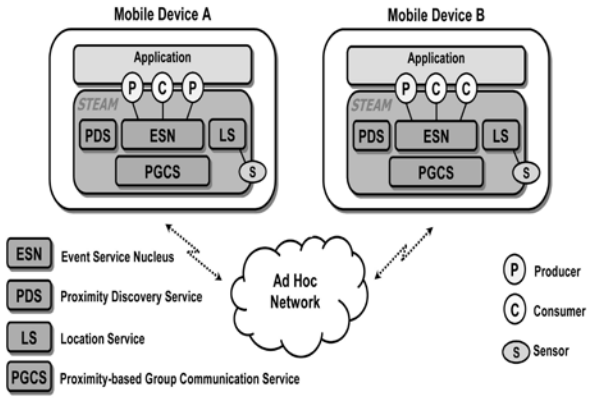

Figure 4.2: Mobile devices hosting the inherently distributed architecture of the STEAM middleware

The above figure illustrates the fact that every mobile device has identical STEAM capabilities. The middleware may support a variable number of consumers and producers on each mobile device, thereby allowing individual devices to both produce and consume events. The event type and proximity information announced by producers are exploited by the PDS to establish communication relationships between mobile entities rather than to optimize event routing as suggested by Carzaniga Subscriptions are used locally to map consumer interests to the sets of currently available events being disseminated within the discovered proximities. As shown below, the operations of the STEAM application programming interface reflect the fact that STEAM is based on an implicit event model as they refer neither to explicit entities nor to designated components of any kind. Instead, the operations for announcing and subscribing to events refer to event types, the former indicating the actual type and the latter using a subject filter to name the type.

announce(eventType et, proximityFilter pf)
unannounce(eventType et)
subscribe(subjectFilter sf,

deliveryHandler dh,
contentFilter cf)
unsubscribe(subjectFilter sf,
deliveryHandler dh,
contentFilter cf)
raise(event e)

STEAM depends on a Location Service (LS) to supply geographical location information. The current location service uses GPS-based sensor data to compute the current geographical location of the mobile device and provides this location information to the middleware and to producers and consumers hosted by the device.

The STEAM application programming interface also illustrates how producers and consumers specify the irrespective event filters. Producers specify their proximity filters and announce them, together with their event types, there by grouping them into associated pairs, while consumers specify both their subject filters and their content filters together with their delivery handlers. Consumers may omit content filters, allowing them to express their interest in events solely using their types, thereby employing a classic, topic-based subscription mechanism.

RESULTS

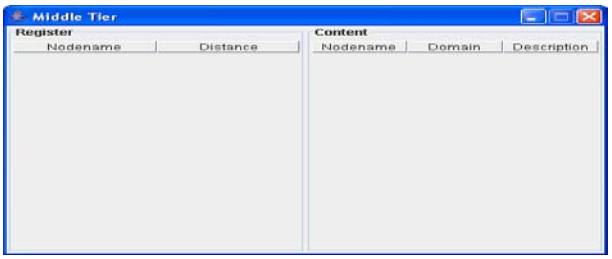

Figure 6.1: MiddleWare(Server)

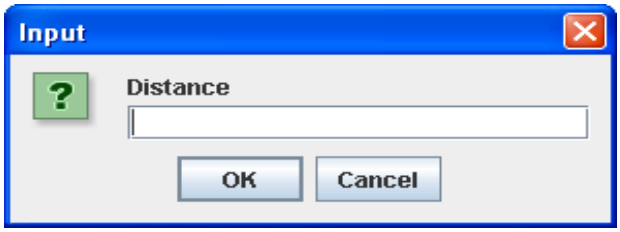

Figure 6.2: Producer Initial Screen

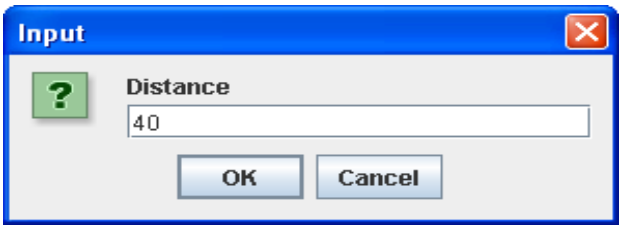

Figure 6.3: Producer Distance Entered

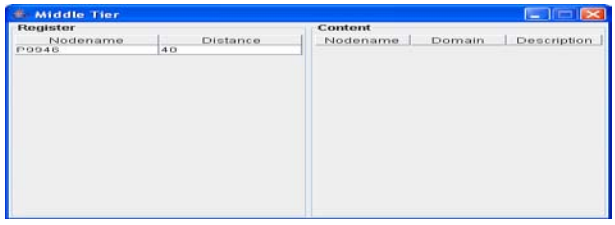

Figure 6. 4: After entering Producer distance, the distance is registered with Middleware.

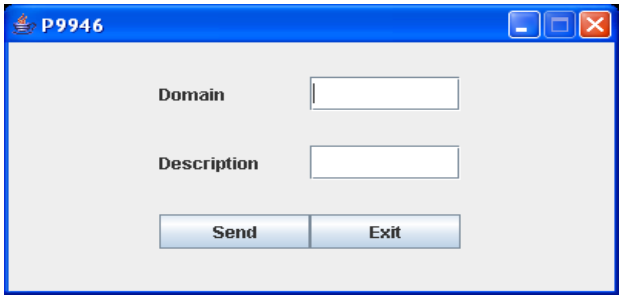

Figure 6.5: Domain Information or Event popup for Producer P9946

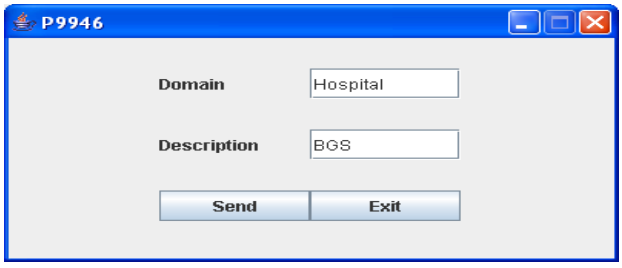

Figure 6.6: Event Information with Domain name and Description for P9946

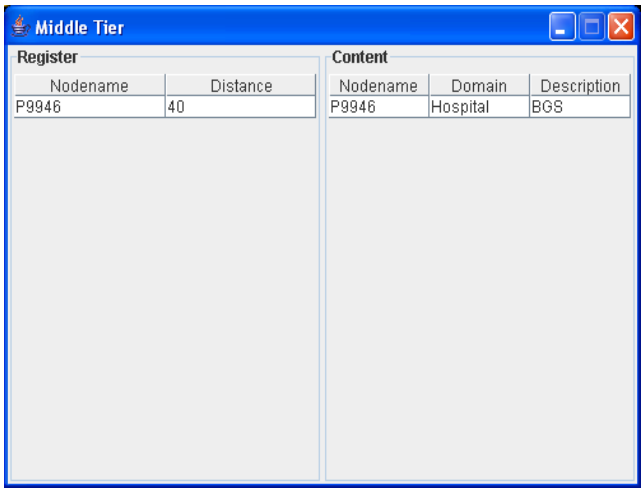

Figure 6. 7: Middleware with Event and Producer range Information

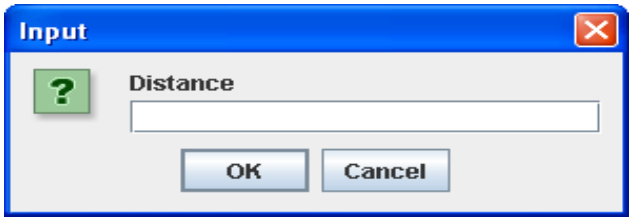

Figure 6.8: Consumer Initial Screen

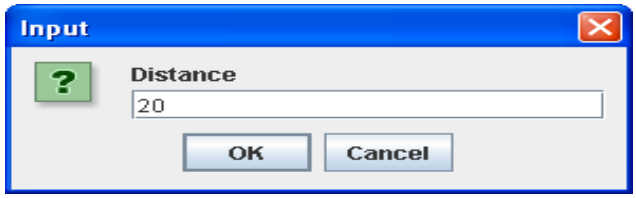

Figure 6.9: Consumer Range Entered

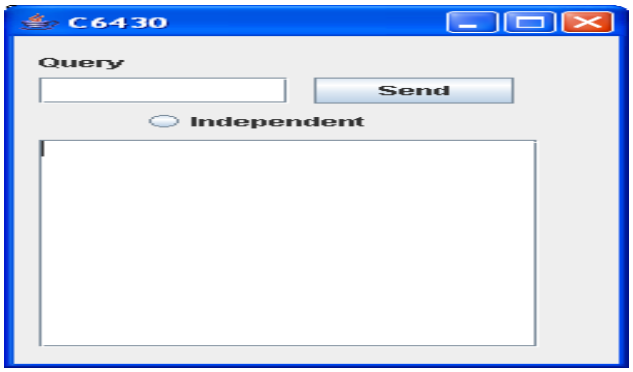

Figure 6.10: Consumer with C6430

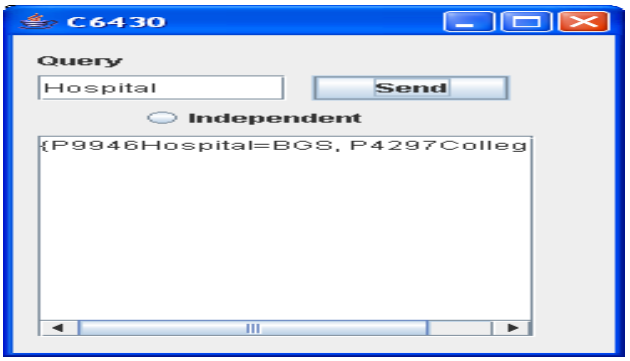

Figure 6.11: Consumer without filter has Hospital (Dependent) Showing all the Events

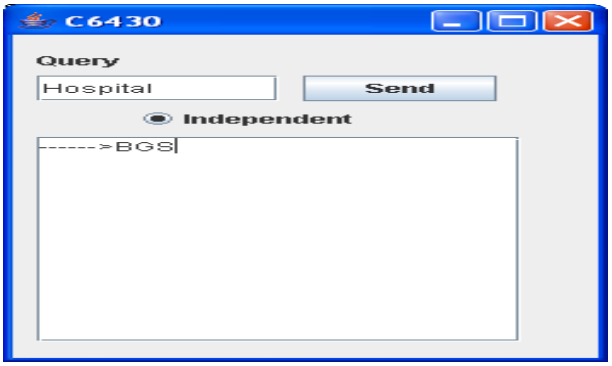

Figure 6.12: Consumer with filter has Hospital (Independent selected) Showing event filter based on query and range of consumer

CONCLUSIONS

For engineering distributed applications, Mobile computing has become increasingly common and the event-based communication paradigm is increasingly adopted. The focus of our work in this paper is on techniques that support collaborative mobile applications which use event-based communication, and we found that a number of other paradigms have also been extended to support such applications as exemplified by various extensions to the tuple space paradigm. In this paper, we presented techniques that can be used by event-based middleware to support collaboration in location-aware mobile applications including 1)location- independent announcement and subscription that allows producers to advertise their events and have these advertisements persist while moving location,2) location-based event filtering, is a distributed approach to filtering that allows an application to define multiple event filters, which may use the actual location of a producer or a consumer, and to apply them at both the producer side and the consumer side. And3) location-dependent delivery of events allows an application to specify multiple event filters, each of which may apply to a different attribute of a specific event. Our techniques provide the basis for supporting a wide range of mobile applications, to support applications with guaranteed quality of service requirements in terms of event-delivery latency is the topic of our future work .We want to exploit the concept of proximity which is introduced 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]. P. Sutton, R. Arkins, and B. Segall, "Supporting Disconnectedness – Transparent Information Delivery for Mobile and Invisible Computing," in Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001). Brisbane, Australia: IEEE CS Press, 2001, pp. 277-285.

[2]. J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri, "Generic Support for Distributed Applications," IEEE Computer, vol. 33, pp. 68-76, 2000.

[3]. G. Cugola, E. D. Nitto, and A. Fuggetta, "The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS," IEEE Transactions on Software Engineering (TSE), vol. 27, pp. 827-850, 2001.

[4]. Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," in Proceedings of the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access(MobiDe'01). Santa Barbara, CA, USA, 2001, pp. 27-34.

[5]. M. O. Killijian, R. Cunningham, R. Meier, L. Mazare, and V. Cahill, "Towards Group Communication for Mobile Participants," in Proceedings of Principles of Mobile Computing (POMC'2001). Newport, Rhode Island, USA, 2001, pp. 75-82.

[6]. Object Management Group, CORBAservices: Common Object Services Specification - Notification Service Specification, Version 1.0: Object Management Group, 2000.

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

[8]. G. Banavar, T. Chandra, R. Strom, and D. Sturman, "A Case for Message Oriented Middleware," presented at Proceedings of the 13th International Symposium on DIStributed Computing (DISC'99), Bratislava, Slovak Republic, 1999.

[9]. R. Meier, "Communication Paradigms for Mobile Computing," ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), vol. 6, pp. 56-58, 2002.

[10]. B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks," IEEE Communications Magazine, pp. 116-126, 1997.

[11]. I. Podnar, M. Hauswirth, and M. Jazayeri, "Mobile Push:Delivering Content to Mobile Users," in Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02). Vienna, Austria, 2002, pp. 563-570.

[12]. http://ulir.ul.ie/bitstream/handle/10344/642/2010-Meier-On-Event-Based.pdf?sequence=2

[13]. http://www.tara.tcd.ie/bitstream/2262/56442/1/On%20Event.pdf

Short Bio Data for the Author

Mannam Chandra Sekhar Completed B.tech in cse and presently pursing M.tech in SE from Gokul Institute of Technology and Science.

Mudiganti Vijaya Bhaskar, Completed his B.Tech ,Computer Science (A.U) 1997 and M.tech,Computer Science (A.U) 2000. Presently Pursuing Ph.D, Soft computing (A.U) .His area of Intrest are Image processing , Artificial Intelligence,Robotics and Soft computing