On Introducing Location Awareness in Publish-Subscribe Middleware

Gianpaolo Cugola and Jose Enrique Munoz de Cote
Dip. di Elettronica e Informazione, Politecnico di Milan
P.za Leonardo da Vinci 32, 20133 Milano, Italy
{cugola, munoz}@elet.polimi.it

Abstract

Having the possibility of routing messages only toward specific areas or subscribing to messages originating in specific locations seems natural when a publish-subscribe model of communication is adopted. Unfortunately, very few work have investigated such kind of services and none of the most widely adopted publish-subscribe middleware implements them. In this paper we first classify possible location-based publish-subscribe services, then we describe an algorithm to efficiently implement them in a distributed publish-subscribe middleware.

1 Introduction

In the last few years, the publish-subscribe model of communication has been proposed in a large number of application domains as a smart solution to spread information (e.g., stock quotes or data acquired from sensors) among a large set of users.

In a large fraction of these application domains, location, either that of publishers or that of subscribers, could be a relevant information if available. Unfortunately, commonly available publish-subscribe middleware do not offer location-based services as part of their API.

In this paper we first (Section 2) study the problem of introducing location awareness into the publish-subscribe model of communication. The result of this effort is a classification of possible location-based publish-subscribe services and an initial analysis of the strategies that can be used to implement the different classes of services in a distributed publish-subscribe middleware[1]. In Section 3, we describe a routing protocol to efficiently implement location-based services in a distributed publish-subscribe middleware. The paper ends with a discussion about related works and a concluding section, which describes the work we plan for the future.

[1]As clarified in Section 2, distributed publish-subscribe middleware are those that use a set of cooperating brokers to route messages from publishers to subscribers.

2 Classification

Components of a publish-subscribe application communicate by *publishing* messages and by *subscribing* to the classes of messages they are interested in. A central component of the publish-subscribe middleware, the *dispatcher*, is in charge of recording information about subscriptions and routing messages from publishers to subscribers.

Within this model, location awareness can be introduced in several ways. In particular, we identified three orthogonal dimensions that allow a fine grain classification of possible location-based publish-subscribe services.

- First of all, location can be introduced at *subscription time* or at *publish time*. By adding location information to the `subscribe` primitive, application components may express their interest in messages published by components located in a specific area. Conversely, if location information is added to the `publish` primitive, it is possible to send messages only toward specific areas, i.e., for subscribed components located in those areas.

- Location can be expressed *physically*, by using geometrical units, e.g., providing latitude, longitude, and ray to indicate a circular area of interest; or *logically*, by using symbolic expressions, e.g., using the label `rome` to indicate the city of Rome.

- Finally, location can be expressed in a *relative* or *absolute* way. In the former case location is expressed relatively to the position of the component who called the service, while in the latter it is expressed according to an absolute coordinate system.

The three dimensions above are orthogonal and not mutually exclusive. As an example, the same middleware

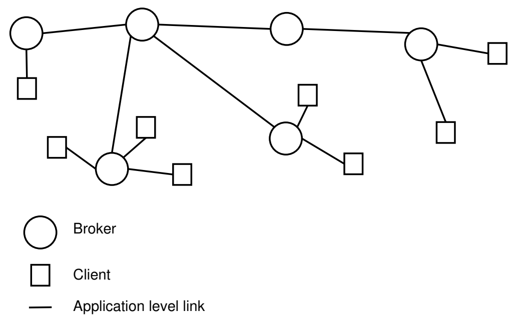

Figure 1. Distributed publish-subscribe middleware.

could offer both location-based subscriptions and notifications expressed physically and both relatively to the caller or absolutely.

The classification above is useful to investigate possible strategies to implement location-based services in a publish-subscribe middleware.

In the remainder of this paper we focus on *distributed*, *content-based* publish-subscribe middleware. To improve scalability, they adopt a distributed dispatcher composed of a set of *brokers* linked together in an *overlay dispatching network*, which collaborate to route messages from publishers to subscribers. Moreover, they offer a powerful subscription language, which allow components to specify the classes of messages they are interested in through *filters*, which enable sophisticated matching on the content of messages.

In Figure 1, an overlay dispatching network of brokers with some connected application components, which act as clients of the middleware, is shown.

We assume that both brokers and application components that require the location-based services know their own absolute position (e.g., through a GPS). Also, we do not make any assumption about the underlying physical network. As we will briefly outline at the end of this section, some specific scenarios, like the use of wireless links, could suggest specific routing mechanisms but we are looking for a general (even if less optimized) solution.

Given these premises, we may first observe that a location-based `subscribe` primitive can be easily (even if not efficiently) implemented "on top" of an existing content-based publish-subscribe middleware. This can be obtained by encoding the position of the publisher onto each published message. At subscription time, the content-based subscription language can be used not only to express the required conditions about the content of messages of interest but also the location they are expected to come from.

This "layered" approach works for both physical and logical locations, and independently from the fact that location is expressed absolutely or relatively, with the overhead of adding location information to every message.

Unfortunately, this approach shows all its limitations in presence of a distributed dispatcher. In fact, if brokers are unaware about location they cannot use it to improve routing. Depending from the strategy adopted for routing subscriptions and messages, either the location-based subscriptions could be routed in areas where they are not relevant or the same could happen to messages.

Things becomes even more complex if we want to implement publish-time location services on top of an existing publish-subscribe middleware. Consider, in fact, a service in the form:
`publishTo(Message msg, Location loc)`
where `loc` identifies the location where message `msg` must be delivered. To implement such service using a "layered" strategy similar to the one above, we should add the position of the subscriber to each subscription, then encode the destination `loc` as part of the message `msg`. Unfortunately, `loc` is conceptually a filter. It should match the positions of the subscribers that have to receive the message, and in most of existing publish-subscribe middleware only filters (issued at subscription time) can match messages, not vice-versa.

As a last remark, we may briefly consider how the specific networking scenario, and particularly the topology of the overlay dispatching network, impacts possible implementations. As an example, it is possible to imagine situations where the topology of the overlay network of brokers can be profitably used to easily implement relative location-based publish-subscribe services. This is the case when the set of brokers is connected through wireless links in a MANET [17]. In such a scenario, the number of hops along the network of brokers is an approximate measure of the distance travelled by messages and subscriptions. This fact could be used to improve routing in presence of location-based publish-subscribe services where location is expressed relatively (with respect to the publisher or subscriber's current position).

3 The location forwarding routing protocol

From the considerations above, it is clear that a new mechanism is required to efficiently implement location-based publish-subscribe services. The general approach we describe in this section moves from an initial consideration: to efficiently route messages and subscriptions, each broker of a distributed location-based publish-subscribe middleware must have some knowledge about the location of the other brokers and clients. We obtain this by defining a *location forwarding* algorithm to distribute location information among brokers.

Additionally, we realize that such an approach may lead to network overload in presence of physical mobility of clients or brokers. We try to alleviate this by introducing a *zone merging* policy. It can be tailored to the specific scenario of mobility to balance between accuracy in routing messages and subscriptions and efficiency in propagating location information.

3.1 Distributing location information

The routing strategy we propose can be described as a refinement of the *subscription forwarding* strategy [4] adopted by most of the currently available distributed publish-subscribe middleware. In such middleware brokers are connected to form an acyclic overlay network. Filters issued by clients at subscription time propagate from broker to broker, flooding the overlay network to define the routes followed back by matching messages. This scheme is usually optimized by avoiding subscription forwarding of the same filter in the same direction[2].

We refine this schema by introducing location information and distributing them among brokers. In our new routing schema, each broker keeps a *location table*, which stores information about the location of all of its neighbors (being them application components which act as clients of the dispatching network or other brokers). In particular, the location table of a broker B stores the current position of the clients directly connected to B, plus information about the position of the clients reachable from each of the neighboring brokers. When a client connects, disconnects or moves, it informs the broker it is attached to of its actual position. This information is used by the broker to update its *location table* and to propagate such update to its neighboring brokers according to a *merging policy* whose details are given in next section.

We call the above process *location forwarding* for its evident analogies with the subscription forwarding schema. It provides each broker with the necessary geographical information to efficiently route both location-based subscriptions and messages. In particular:

At subscription time. When a client subscribes, it provides both a content-based filter and a location filter. The latter is used to define the originating zone of the messages it is interested in. Given this information, each broker, starting from the one the client is attached to, uses its location table to choose the neighboring brokers that must receive the subscription. In practice, this process follows the usual subscription forwarding schema enriched by the information provided by the location tables, which are used to limit the propagation of subscriptions.

[2]Other optimizations are possible, e.g., by defining a notion of "coverage" among filters.

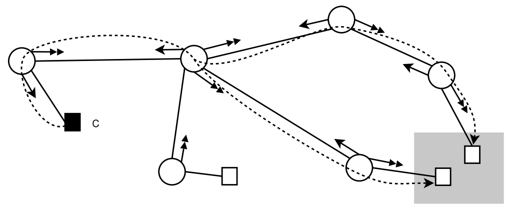

Figure 2. An example of subscription routing.

As an example, Figure 2 shows the path followed by a subscription issued by client C for a "black" filter and for the gray zone. Double headed arrows outgoing from a broker model the content of its location table with respect to the gray zone, i.e., the "directions" along which one or more clients located into that zone can be reached. Single headed, black arrows model the content of the subscription table of each broker as populated by the "black" subscription issued by C.

When location tables change. Since location tables are used to limit the propagation of subscriptions along the overlay network of brokers, when they change it can be necessary to forward subscriptions that were not forwarded before or to revoke previously forwarded subscriptions. In particular, the subscription table of each broker must record not only the content-based filter associated with the subscription, but also the related location filter. When the location table changes, this information can be used to decide if the subscription must be forwarded to new brokers or if some previously forwarded subscription must be revoked.

At publish time. When a client publishes a message it provides a location filter, which defines the zone the message must reach. This information is matched by each broker (starting from the one the client is attached to) against its location table, while the message content is matched against the subscription table as usual. Both these information are used to decide the neighbors that must receive the message.

As an example, Figure 3 shows the path followed by a message sent by C, which matches the "black" filter and is directed toward the gray zone. Again, single headed arrows model the content of subscription tables of each broker for a "black" and a "gray" filter that where previously issued by clients $C1$, $C2$ (the black one), and $C3$ (the gray one) toward the zone where client C currently stands. Double headed arrows model location tables of brokers with respect to the gray zone. From the figure you may notice that client $C1$ did not receive the message even if it was

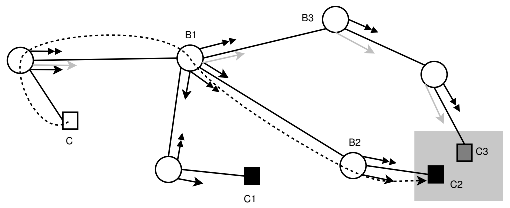

Figure 3. An example of message routing.

subscribed to the black filter because it is outside from the gray zone, i.e., the location table of broker $B1$ suggested to route the message only toward $B2$. Similarly, client $C3$ is inside the gray zone but it is not subscribed to the black filter, consequently it does not receive the message,i.e., this time it was the subscription table of broker $B1$ to suggest routing the message toward $B2$. Only client $C2$ satisfies both the requirements of being inside the destination (gray) zone and being subscribed to the (black) filter matching the message.

3.2 Merging policy

In describing the routing schema above, we glossed over the details about how the location information is propagated from broker to broker. At first, we may suppose that the exact location of each client, received by the broker the client is attached to, is forwarded to the other brokers. The effect of this policy is that the location table of each broker is populated with information about the exact position of each client reachable from each neighboring broker. Moreover, each time a new client connects to the system or existing clients disconnect or move, location information must be updated all over the network of brokers. Clearly, this approach provides maximum precision, but it cannot be applied to large, dynamic scenarios.

Inspired on Carzaniga's work [5], which describes a strategy to merge filters to reduce the size of subscription tables and the communication overhead required to update them, our idea is to use an appropriate *merging policy* for location information. According to this policy, each broker merges information about position of the clients directly connected to it and location information coming from neighboring brokers to a larger zone, which would be forwarded to the other neighboring brokers. If new clients whose position is within the merged zone already propagated connect, or if updates comes from neighbors that fall within the merged zone, no further propagation is required.

Clearly, the merging policy adopted must be carefully chosen to find the best balance between the goal of reducing the overhead involved in propagating location information, with the precision required to correctly limit the propagation of location-based subscriptions and messages. This balance must take into consideration a precise characterization of the application scenario in which the middleware is going to be used in terms of size, dynamic, and expected traffic. Currently (see Section 5) we are implementing the location forwarding protocol and we plan to experiment with it in different scenarios to acquire the necessary experience on how the best merging policy could be chosen.

4 Related Work

Most publish-subscribe middleware target local area networks and adopt a centralized dispatcher. In the last few years, the problem of wide-area event notification attracted the attention of researchers and systems exploiting a distributed dispatcher became available, e.g., [9, 4, 12, 13, 20, 1, 19].

In [10] authors describe (1) an extension to the REBECA [12] publish-subscribe middleware to support roaming of clients between brokers and (2) location-based subscriptions. As for this second issue, which is related with our work, authors describe an extension of the REBECA's routing mechanism to provide relative, location-based subscriptions. The approach adopted is similar to the one described in Section 2: location-based subscriptions permeates the entire network of brokers, potentially reaching also areas that are not relevant, since they are outside the location specified in the subscription. An interesting issue investigated in [10] is that of providing mobile clients with correct and precise information while they move, in presence of location-based subscriptions. An approach is presented to guarantee the correct behavior to the routing protocol is such situations. A similar approach is described in [8] to limit the bootstrapping latency required to setup routes when a new client arrives or when an existing client moves, roaming from broker to broker. Currently, we did not address these issues, which impacts the way information about new location is propagated. We plan to investigate them in the future, possibly adapting the ideas described in the works above.

[6] presents a mechanism to efficiently filter a stream of events that represent current location of clients, against a set of spatial predicates. The goal is to determine the set of clients that could be interested in receiving some message based on their position. The authors propose a middleware based on a centralized spatial matching engine, which collects subscriptions and delivers them to clients. Clients are in charge of matching those subscriptions against their current position. The results of this process are given back to the engine. This approach tries to reduce the effort for the matching engine and is similar to the generic approach described in Section 2 for location-based subscriptions, if we consider subscription-forwarding up to clients as the main

routing mechanism.

Scalable Timed Events And Mobility (STEAM) [16] developed at Trinity College of Dublin, is an event based middleware that has been designed to be deployed over MANETs. STEAM targets application scenarios that include a large number of application components that communicate using wireless technology in an ad-hoc scenario. In STEAM events are valid in a certain geographical area surrounding the publisher. As a consequence STEAM provides what we called a relative, publish-time location-based publish-subscribe service. The STEAM implementation is specifically tailored to MANETs and takes benefit of a proximity-based group communication service [15].

[7] presents CAMEL, a middleware to support intelligent location-based services in a component-based distributed architecture that supports relative-physical location matching.

L-ToPSS [3] is an extended model for a publish-subscribe middleware (ToPSS) that offers location-based-services which supports relative, physical location matching. They process location information independently from the subscription-publication matching process. In their work authors focused on the filtering engine, consequently their work can be seen complementary with respect to the one presented in this paper, which focused on routing.

In our work we distinguished between relative and absolute modeling of location and between logical and absolute. In [2], Bauer et al. discuss explicit location modeling for infrastructure based and ad hoc networks in the NEXUS and CANU systems, the outcome of their work are the requirements for a general location modeling language in ubiquitous computing.

Mobile devices of today can offer location-based services (LBS) that allow end users to obtain domain dependant information based on their present location. Hinze and Voisard inquire on combining LBS and event-based systems to their Tourism Information Provider (TIP) [14] system. While moving, clients trigger location events that produces a central system reaction, ending in the dissemination of tailored (location dependant) information.

In [11] the notion of *scope* is introduced to structure event availability and notification by restricting the visibility of published events to a subset of subscribers in the system. This general idea can be adapted to the problem of providing location-based publish-subscribe services by using location and proximity as the mechanism of scoping.

Advertisements, as first introduced by Siena [4], can be used to reduce the overhead involved in the "layered" strategy described in Section 1 to implement location-based subscriptions, in case of distributed publish-subscribe middleware. Unfortunately, they do not help in implementing a location-based `publish` service.

5 Conclusions and future work

Location awareness seems to be relevant in most of the application domains where publish-subscribe middleware are in use. In this paper we proposed a classification of location-based publish-subscribe services and described a "location forwarding" protocol to implement the different kind of services in a distributed publish-subscribe middleware. A "merging policy" separate from the general routing mechanism provides a way to adapt the general approach to specific scenarios.

Currently, we are working on implementing this algorithm in REDS [18], a distributed publish-subscribe middleware that was implemented at Politecnico di Milano based on the experience gained in working with JEDI [9].

We are also working on generalizing the approach presented in this paper to the broader problem of *context-based routing*. Indeed, location is only one of the contextual information that can be useful to filter messages and subscriptions in a publish-subscribe model of communication. Other information could be used, e.g., information about the hardware running application components or about the users using them. As an example, one could be interested in routing messages carrying high resolution images only toward interested users that have a screen large enough to show them.

Our plan is to provide a precise characterization of the kind of contextual information that could fit this framework and to generalize the location forwarding algorithm presented in this paper to these new services.

References

[1] G. Banavar et al. An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. In *Proc. of the 19th Int. Conf. on Distributed Computing Systems*, 1999.

[2] M. Bauer, C. Becker, and K. Rothermel. Location Models From the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks. In *Proc. of the Workshop On Location Modeling For Ubiquitous Computing, (UBICOMP)*, 2001.

[3] I. Burcea and H.A. Jacobsen. L-ToPSS: Push-Oriented Location-Based Services. In *Proc. Technologies for E-Services*, LNCS, pages 131–142, 2003.

[4] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Trans. on Computer Systems*, 19(3):332–383, August 2001.

[5] A. Carzaniga, M.J. Rutherford, and A.L. Wolf. A Routing Scheme for Content-Based Networking. In

Proc. of IEEE INFOCOM 2004, Hong Kong, China, March 2004.

[6] X. Chen, Y. Chen, and F. Rao. An Efficient Spatial Publish/Subscribe System for Intelligent Location-Based Services. In *Proc. of the Workshop on Distributed Event-Based Systems*, June 2003.

[7] Y. Chen, F. Rao, X. Yu, and D. Liu. CAMEL: A Moving Object Database Approach for Intelligent Location Aware Services. In *Proc. of MDM 2003*, LNCS 2574, pages 331–334, 2003.

[8] M. Cilia, L. Fiege, C. Haul, A. Zeidler, and A.P. Buchmann. Looking Into the Past: Enhancing Mobile Publish/Subscribe Middleware. In *Proc. of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, San Diego, CA, June 2003. `http://www.eecg.utoronto.ca/debs03/dp.html`.

[9] G. Cugola, E. Di Nitto, and A. Fuggetta. The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS. *IEEE Trans. on Software Engineering*, 27(9):827–850, September 2001.

[10] L. Fiege, F.C. Gartner, O. Kasten, and A. Zeidler. Supporting Mobility in Content-Based Publish/Subscribe Middleware. In *Proc. of Middleware 2003*, LNCS 2672, 2003.

[11] L. Fiege, M. Mezini, G. Mühl, and A.P. Buchmann. Engineering Event-based Systems with Scopes. In B. Magnusson, editor, *Proc. of the 16th European Conference on Object-Oriented Programming (ECOOP02)*, volume 2374 of *LNCS*, pages 309–333, Malaga, Spain, June 2002. Springer-Verlag.

[12] L. Fiege, G. Mühl, and F.C. Gärtner. Modular Event-Based Systems. *Knowledge Engineering Review*, 17(4):359–388, 2002.

[13] R. Gruber, B. Krishnamurthy, and E. Panagos. The Architecture of the READY Event Notification Service. In *Proc. of the 19th IEEE Int. Conf. on Distributed Computing Systems—Middleware Workshop*, 1999.

[14] Annika Hinze and Agnes Voisard. Combining even notification services and location-based services in tourism. Technical report, March 2003.

[15] M. Killijian, R. Cunningham, R. Meier, L. Mazare, and V. Cahill. Towards Group Communication for Mobile Participants. In *Proc. of ACM Workshop on Principles of Mobile Computing (POMC'2001)*, pages 75–82, Newport, Rhode Island, USA, 2001.

[16] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks. In *Proc. of the International Workshop On Distributed Event-Based Systems (DEBS'02)*, Vienna, Austria, 2002.

[17] C.E. Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2000.

[18] REDS Web Page. `http://zeus.elet.polimi.it/reds/`.

[19] B. Segall et al. Content Based Routing with Elvin4. In *Proc. of AUUG2K*, Canberra, Australia, June 2000.

[20] M. Wray and R. Hawkes. Distributed Virtual Environments and VRML: an Event-based Architecture. In *Proc. of the 7th Int. WWW Conf.*, Brisbane, Australia, 1998.