ITM Web of Conferences **6**, 03001 (2016)
DOI: 10.1051/itmconf/20160603001

Mobility support in publish/subscribe systems

Vladimir Antipov[1, a], Oleg Antipov[1] and Aleksander Pylkin[1]

[1] *Ryazan State Radioengineering University, Department of Computational and Applied Mathematics, 390005, Ryazan, Russia*

Abstract. Publish/subscribe (pub/sub) is considered as a valuable middleware architecture that proliferates loose coupling and leverages reconfigurability and evolution. Up to now, existing pub/sub middleware was optimized for static systems where users as well as the underlying system structure were rather fixed. We explore the question whether existing pub/sub middleware can be extended to support mobile and location-dependent applications. We first analyze the requirements of such applications and distinguish two orthogonal forms of mobility: the system-centric physical mobility and an application-centric logical mobility (where users are aware that they are changing location). We introduce location-dependent subscriptions as a suitable means to exploit the power of the event-based paradigm in mobile applications.

1 Introduction

Publish/subscribe (pub/sub) is considered as a valuable middleware architecture that proliferates loose coupling and leverages reconfigurability and evolution. Up to now, existing pub/sub middleware was optimized for static systems where users as well as the underlying system structure were rather fixed. We explore the question whether existing pub/sub middleware can be extended to support mobile and location-dependent applications.

2 Mobility issues in publish/subscribe middleware

Mobile clients have many characteristics, among them the need to disconnect from the network by different reasons. Be it by geographical, administrative, or power saving reasons, connection to the same broker all the time is no longer possible. Hence, we have to take into *account* that clients will disconnect from their border broker once in a while. The middleware has to deal with moving clients and the possibility that a disconnected client will reconnect to the same or to a different broker later. A first step towards mobility is to enhance existing pub/sub middleware to allow for roaming clients so that existing applications can be used in mobile environments.

This means that the interfaces for accessing the middleware and the top level applications are not required for changing. More importantly, the quality of service offered by the middleware must not degrade substantially. The resulting location transparency is necessary to make existing applications mobile, e.g., stock quote monitoring seamlessly transferred from PCs to PDAs. On the other hand, future applications do not require complete transparency, but rely on awareness of mobility. More specifically, mobility support should blend out unwanted phenomena, like disconnectedness, and enforce wanted behavior, like the location awareness in location-based services. Consequently, extending the interface of the pub/sub middleware to facilitate location awareness is a promising open issue, since most existing works are concentrated on the transparency only.

When roaming, clients change (at least some portion of) the context they are operating in, and they might want to react to these changes, e.g., to adapt their subscriptions. However, an appropriate infrastructure support has to relieve the application from having to react "manually" to all changes. The middleware should rather offer an automated adaptation to context changes, i.e., facilitating location dependency. This leads to a different notion of mobility and we distinguish:

Physical mobility: a client that is physically mobile disconnects for certain periods of time and has different border brokers along its itinerary through the infrastructure. The main concern of physical mobility is location transparency.

Logical mobility: a client that is logically mobile is aware of its location changes. In order to relieve the client from adapting manually to new locations, the main concern of logical mobility is automated location awareness within the pub/sub middleware.

Physical and logical mobility are two orthogonal aspects of mobility. Since the physical layout of a pub/sub system is usually fixed and its layout does not usually correspond to geographical realities, it seems reasonable to separate the two notions of mobility. In this paper, we assume logical mobility to be a refinement of physical mobility in that a client remains connected to the same broker when is roaming logically. The two notions

[a] Corresponding author : s_titov@mti.edu.ru

have different quality of service requirements and therefore different solutions are developed to match both.

3 Physical mobility

Physical mobility is similar to what in the area of mobile computing is called terminal mobility or roaming. A client accesses the system through a certain number of access points (GSM base stations, WLAN access points, or border brokers). When moving physically, the client may get out of reaching of one access point and move into the reaching of a second access point which are not necessarily overlapping. In general we cannot expect to have seamless access to the broker network but more a sequence of phases of connectedness, e.g., on the daily route between home and office. In this setting we analyze the quality of service requirements from the viewpoint of roaming clients:

Interface. Obviously, the interface to the pub/sub system must not be changed as the legacy applications have not an aware of mobility.

Completeness. Despite intermittent disconnects, the pub/sub middleware delivers all notifications for a client eventually. This is the core requirement for transparency.

Ordering. The sender orders notifications on first come (FIFO).

Responsiveness. The delay of relocating a roaming client should be minimal to maximize the responsiveness of the system. This has to be taken into account when designing a relocation protocol.

4 Possible solutions

One solution would be to rely on Mobile IP [1] for connecting clients to border brokers, hiding physically mobility in the network layer. The drawback, however, is that the communication is also hidden from the pub/sub middleware, which is then not able to draw from any notification delivery localities or routing optimizations, thereby possibly violating the requirement of responsiveness. Such an approach might only be feasible if the physical and logical layouts of a given system are completely orthogonal.

A different, solution to implement physical mobility would be to use sequences of sub-unsub-sub calls to register a client at a new broker. When a client moves from border broker B1 to B2, it simply unsubscribes at B1 and (re-) subscribes at B2, without any support in the middleware. But a client may not detect leaving the range of a broker and in this case is not able to unsubscribe at its old location. Even more severely, during its time of disconnectedness, the client might miss several notifications or get duplicates, even if notifications are flooded in the network and the location change is instantaneous. This problem is depicted in Figure 1. Hence, this solution is not complete [2].

While physical mobility is a rather technical issue invisible to the application, logical mobility involves location awareness. An example for logical mobility is when clients move around a house or building that is served by only one border broker. In this case, the user might be interested to receive just those notifications that refer to the room he is currently located in. Note that a client can be both logically and physically mobile at the same time.

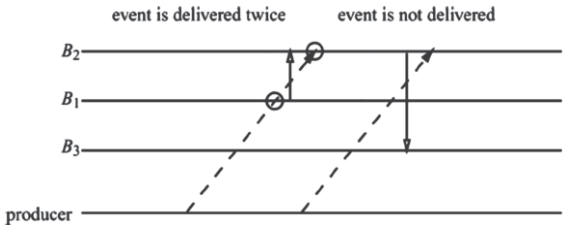

Figure 1. Missing notifications in a flooding scenario

A logically mobile client moving from one location to another, e.g., from one room to the other in a company building, will expect a frictionless change of location explicitly without a notable setup time after having changed from its own office to the conference room next door. The adaptation of some location dependent subscription should take place "instantaneously". Intuitively, we would like to experience the notion of being subscribed to "everything, everywhere, all the time" and increase the reactivity of the system to moving clients.

5 Location-dependent filters

A pub/sub system that offers location-dependent filters has the same interface as a regular pub/sub system (i.e., it offers the pub, sub, unsub, notify primitives). However, in specifying subscription filters for name/value pairs referring to "location" it supports a new primitive to specify things like "all notifications where the attribute location equals my current location". More precisely, we postulate a specific marker myloc that can be used in a subscription. The marker consists of a specific set of locations that depend on the current location of the client. For example, a client could issue a subscription for all free parking spaces in the vicinity of his current location as follows: (service = "parking"), (location $\in$ myloc), (car-type $\geq$ "compact").

The set of locations associated with the marker is taken from a particular range L of locations. This set is application dependent and can, for instance, contain all the different rooms of a building, all the streets of a town, or all the geographical coordinates given by a GPS system up to a certain granularity.

Given a notification with the attribute location, the subscription (location $\in$ myloc) will evaluate to true for a particular client at location y if and only if $x \in$ myloc(y) where myloc(y) is the specific set of locations associated with y. In this case we say that the notification matches the location-dependent filter. The simplest form of myloc(y) is simply the set$\{y\}$. In this case a notification matches the subscription if $x = y$. But in the car example, the car driver looking for a parking space might want to specify: (location = "at most two blocks away from myloc"). In this case, myloc corresponds to all elements of L that satisfy this requirement.

A Tentative but Incomplete Solution for Logical Mobility. While location-dependent filters are not directly supported by current pub/sub middleware, one might argue that it is not very difficult to emulate them on top of currently available systems in this case. The idea would be to build a wrapper around an existing system that follows the location changes of the users and transparently unsubscribes to the old location and subscribes to the new one when the user moves. However, depending on the internal routing strategy of the event system, it may lead to unexpected results. The routing strategies deployed in many existing content-based event systems such as Siena [3], Elvin [4], and Rebeca [5] lead to blackout periods where no notifications are delivered. The problem is that it usually takes an unnegligible time delay to process a new subscription. After subscribing to a filter, it takes some time td until the subscription is propagated to a potential source. Then it takes at least another td time until a notification reaches the subscriber. This phenomenon is depicted in Figure 2a. (Note that the delay t_d may be different for different notification sources and may change over time.) If the client remains at any new location less than $2t_d$ time, then the subscriber will "starve", i.e., it will receive only few or no notifications.

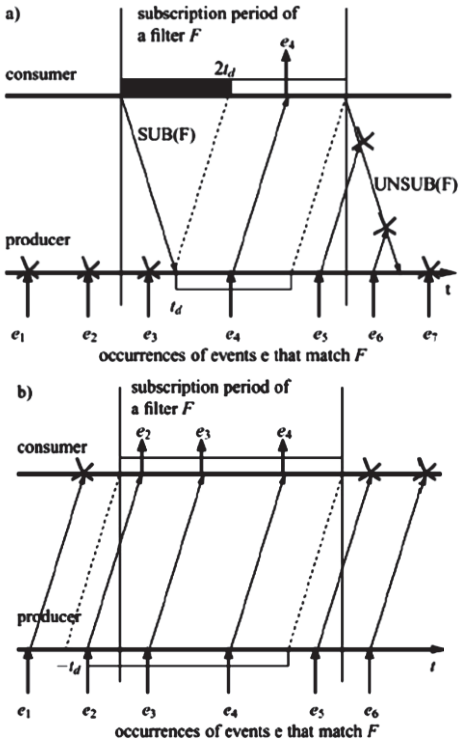

Figure 2. Blackout period after subscribing with simple routing a) and flooding with client-side filtering b).

An Intuitive but Inefficient Solution. Another basic solution that can be immediately built using existing technology is again based on flooding. The local broker can then decide to deliver a notification to a client depending on the client's current location (Figure 2b). Obviously, flooding prevents the blackout periods, which were present in the previous solution, but it should be equally clear that flooding is a very expensive routing strategy especially for large pub/sub systems [6].

Quality of Service of Logical Mobility. Interestingly, while flooding is very expensive and therefore not desirable, it comes very close to the quality of service that we would like to achieve for logical mobility, namely to the notion of being subscribed to "everything, everywhere, all the time". The problem is that it is hard to precisely define the behavior of flooding without reverting to some unpleasantly theoretical constructions of operational semantics.

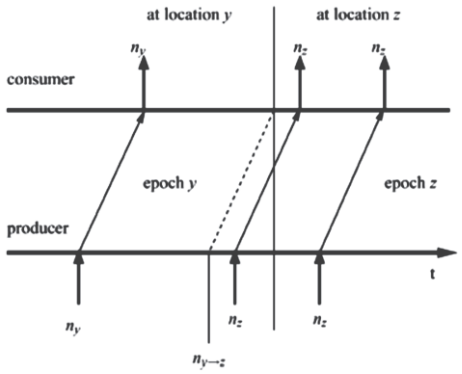

Figure 3. Defining the quality of service for logical mobility using virtual notifications $ny \rightarrow z$ that arrives at the consumer just at the time of the location change from y to z.

With logical mobility there is, however, no danger of receiving a notification twice because the consumer remains attached to the same "delivery path". The quality of service we require for logical mobility therefore is simply stated as follows: On change of location from x to y, all notifications should be delivered to the consumer "as if" flooding were used as underlying routing strategy. This statement is made a little more concrete in Figure 3 where the sequence of notifications generated by any consumer is divided into epochs that correspond to when the notification actually arrives at the consumer (the epoch borders between location y and z are drawn as a virtual notification $ny \rightarrow z$). We require that all notifications matching the current location-dependent subscription from every such epoch must be delivered. Intuitively, the epochs define the semantics of flooding.

6 Conclusion

This paper has presented an approach to support mobility in existing publish/subscribe middleware. We have analyzed the problem of mobility from the viewpoint of the event-based paradigm and have identified two separate types of mobility. 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.

References

1. D. Johnson. Wireless Networks, **1**, 311 (1995).
2. Zeidler and L. Fiege, *Proceedings of the 23rd International Conference on Distributed Computing*

Systems Workshop on Mobile Computing Middleware, (2003).
3. W. Segall and D. Arnold, *Proceedings of the 1997 Australian UNIX Users Group*, (Brisbane, 1997).
4. A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. ACM Transactions on Computer Systems, **19**, 332 (2001)
5. L. Fiege and G. Mühl, *Rebeca Event-Based Electronic Commerce Architecture* (2000)
6. G. Mühl, L. Fiege, F.C. Gärtner, and A.P. Buchmann. *Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002)* (2002).