Content-Based Dispatching in a Mobile Environment

Gianpaolo Cugola, Elisabetta Di Nitto, Gian Pietro Picco
Politecnico di Milano, Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{cugola, dinitto, picco}@elet.polimi.it

1. INTRODUCTION

The advances in the area of wireless networks and the availability of powerful mobile computing devices like laptops and personal digital assistants (PDAs) at a reasonable price have fostered the diffusion of *mobile computing* [7]. This term generically refers to a scenario where the hosts taking part in the computation are mobile. This leads to very different interpretations. At one extreme, mobility takes into account the needs of *nomadic users*, i.e., users who connect to the network from arbitrary and changing locations, and who are not permanently connected. Instead, *mobile users* retain connectivity during movement, typically by exploiting wireless communication links. These forms of mobile computing already found their way in the market, which offers services to read their e-mail or browse the Web while on the road.

At the other extreme, however, research in mobile computing is investigating even more radical scenarios where communication takes place completely through a wireless network, without any contact to a fixed network. Network functions (e.g., routing) are then provided completely by the mobile devices themselves, which operate both as intermediate and end nodes in the network, composing what is known as an *ad-hoc network* [7]. This allows people to communicate and collaborate even in environments where a conventional fixed network is not available, either because it has been destroyed (e.g., due to a natural disaster) or because it has never been developed (e.g., because economically or practically unfeasible).

In general, mobile computing fosters a new style of computation that is *context-aware*. The resources and services available to mobile users are not fixed, rather they change according to the availability of connectivity and to the current location. Thus, for instance, a mobile user moving through a building with a PDA could take advantage of services such as printing through the nearest printer available or browsing the list of internal phone numbers starting from the people that work in the department the user is walking through.

Despite the technological advances in wireless networking, the research community is still investigating how to support and exploit context-aware computing. In particular, context-aware computing is likely to have an impact on middleware technology. Middleware will provide the programmer with proper mechanisms to manage changes in connectivity and location-dependent access to resources and services. Based on our experience, we argue that publish/subscribe middleware is a good candidate for context-aware computing [2],[1][4]. The communication enabled by this middleware is inherently asynchronous, and thus better suited than synchronous communication to cope with the unannounced disconnection characterizing mobile networks. Moreover, the routing of a message is performed on the basis of its content and on a declaration of interest that is explicitly expressed by potential recipients, as opposed to conventional approaches where routing is determined by the address of the receiver as established by the publisher of the message. The aforementioned characteristics make this kind of middleware suitable for situations where the communicating components need to be decoupled and dynamic reconfiguration is a critical problem. For instance, a PDA could easily exploit a publish/subscribe approach to advertise its presence in a room and retrieve the services available in it.

Unfortunately, although several publish/subscribe middleware have been developed thus far, both in industry and in academia, they happen to be designed for fixed network environments. Thus, they do not take into account the problems introduced by wireless networks, related to the dynamic reconfiguration of the network topology.

The current goal of our research is to evaluate the impact of mobile computing on the design and implementation of a publish/subscribe message-oriented middleware. In the remainder of this paper we introduce publish/subscribe middleware, provide a preliminary discussion about how mobility affects this kind of middleware, and present our research agenda on the topic.

2. Publish/Subscribe Middleware

In a publish/subscribe middleware components communicate by generating and receiving *messages*. A component usually generates a message when it wants to let the “external world” know that some relevant event occurred either in its internal state or in the state of other components with which it interacts. The relevant aspect of this process is that the message does not contain the list of its recipients. The message is propagated to any component that has declared interest in receiving it by issuing a *subscription*, which is a predicate on the content of the event. A connector called *dispatcher* or *bus* is in

charge of receiving messages, evaluating the subscription, and propagating the messages to their listeners. The propagation of messages is completely hidden to the component that generated the message. Thus, the dispatcher implements a multicasting mechanism that fully decouples message generators from receivers. This provides two important effects. First, a component can operate in the system without being aware of the existence of other components. The only knowledge necessary is the one about the structure of the messages that are of interest, in order for the component to issue the related subscriptions. Second, it is always possible to plug a component in and out of the architecture without affecting the other components directly. These two effects guarantee a high compositionality and reconfigurability of the resulting software architecture.

In order to guarantee scalability of publish/subscribe middleware, some approaches for distributing the dispatcher have been proposed. In the middleware we developed, JEDI [2], the dispatcher is actually represented by a number of *dispatching servers* organized in a hierarchy. The distribution of dispatching servers is transparent to the other components. The system, in fact, guarantees that subscribers receive all matching messages regardless of the position of the corresponding publishers in the dispatching hierarchy. Dispatching servers coordinate among each other to minimize the network traffic. In particular, they implement a coordination protocol that is used to manage propagation of messages outside the boundaries of a single dispatching server.

A JEDI feature relevant to mobile computing is the possibility of supporting the disconnection and reconnection of components to the system. Components can invoke the `moveOut` operation to disconnect from the dispatcher, change location, and then invoke the `moveIn` operation to reconnect again through a different dispatching server. Dispatching servers manage temporary storage of messages for the duration of the disconnection and coordinate during the execution of `moveIn` to guarantee that messages are not duplicated and are received in a sequence that respects causal ordering. As we discuss in the next section, although this feature enables logical mobility of components, it must be extended to cope with the requirements posed by physical mobility.

3. Publish/Subscribe for Mobility: A Preliminary Analysis

As mentioned in the introduction, mobile computing actually encompasses a broad range of scenarios, which are likely to pose very different requirements to the middleware. A first, rough categorization distinguishes among two scenarios:

Base station. This scenario is characterized by the presence of a wired network, that is exploited to carry out all network functions. Routers on such a network constitute the access points for the mobile hosts, that communicate one with the other by exploiting the fixed infrastructure. The Mobile IP protocol [6] is based on this scenario. Other examples are the cellular phone network (GSM and the new UMTS) and the satellite network.

Ad-hoc network. In this case only wireless connectivity exists, and there are no predefined hosts on the fixed network acting as routers. In a typical configuration, each mobile host operates as a router, thus acting as intermediary between two or more hosts that otherwise could not be reciprocally visible in the wireless network. Whenever hosts loose contact, the topology of the network is dynamically changed accordingly. An example of network protocol developed to support this scenario is Bluetooth [3].

Within this scenarios, however, different elements can vary, bringing very different requirements on the middleware. Typically, the following elements are considered relevant in the literature:

- *Number of mobile nodes*. It may range between few units in the case of “impromptu” collaborative work meetings, to thousands of units in the case of an automotive navigation system.
- *Range of movement.* Imprompty meetings typically need a short range. Disaster recovery and military applications typically involve a medium range of operations. Finally, automotive navigation is typically wide range.
- *Frequency of movement.* Scenarios span from continuous mobility (e.g., automotive navigation) to burst mobility (e.g., meetings or some cases of ubiquitous computing).

All the aforementioned dimensions influence the design choices for a middleware and in some cases could even lead to conflicting solutions. In general, we can argue that to operate in a mobile environment, a publish/subscribe middleware (and in general any middleware for mobility) must offer mechanisms that cope with the unannounced disconnection of its clients. The trivial approach could be of managing unannounced disconnections as network faults that are not under the responsibility of the middleware. This approach, however, overlooks the fact that, in mobile environments, unannounced disconnection is more the rule than the exception.

Other main problems to be considered concern the deployment and reconfiguration of the dispatching system. Intuitively, these depend mostly on the networking scenarios being considered. In the case of a base station scenario, it is reasonable to assume that dispatching servers are installed on the wired network. Mobile hosts connect to the dispatching server that is installed closest to the router they are currently connected to. By moving from a location to another, mobile hosts could loose connection with the router. From the point of view of the

publish/subscribe middleware, this means that they disconnect from the system in an unannounced way. On the side of the dispatcher the disconnection could be simply detected the first time it needs to contact the mobile host to deliver a message. In this case, the dispatcher could simply execute the move-out procedure on behalf of the mobile host. On the side of the mobile host, proper buffering mechanisms should also be provided for outgoing messages. The publish/subscribe infrastructure should also manage special cases such as the mobile host connects to another dispatcher before its origin dispatcher has realized that it has disconnected.

In the ad-hoc network case the architecture of the dispatching system has to be radically modified in order to adapt to the new characteristics of the network. We can envisage two possible approaches:

- Dispatchers are installed on a set of mobile hosts (e.g., the ones that have more computational resources can be selected) according to a certain topology. As in the base station scenario they properly manage unannounced disconnection of publishers/subscribers. Differently from the previous case, now they need to implement some mechanisms that enable dynamic reconfiguration of the dispatching system whenever a dispatcher is not anymore reachable through the wireless communication. While the problem of dynamic reconfiguration is currently being studied for wired networks [5], the results obtained in this context could not be directly applicable in the ad-hoc network case given the potential high frequency of such reconfiguration events.
- The middleware does not rely on a dispatching system having a specific topology. Each component acts both as a publisher/subscriber and as a dispatcher. In particular, it just acts as a repeater of all the messages it receives by exploiting the broadcast nature of the wireless communication. All components being in a certain range can be reached by a message and repeat it again. In this case, of course the same component could receive a message twice. Therefore, proper filtering mechanisms should be defined so that the component avoids reconsidering and repeating this message again.

All approaches we have mentioned appear to have important effects on the semantics of services offered by the middleware to the applications. For instance, providing guaranteed delivery of messages in the ad-hoc scenarios could be even impossible, given the unpredictability of system reconfigurations. Therefore, the behavior of a middleware suitable for mobility has to be carefully identified and defined.

In addition, new requirements are imposed on the underlying network protocols. Our current experiments with TCP/IP have shown several weaknesses of such protocol when applied to ad-hoc scenarios:

- While TCP declares to guarantee delivery, it does not manage the cases where the two parties involved in a communication loose connectivity and never regain it.
- There is insufficient information to determine when and why a disconnection has occurred.

4. Conclusions and future work

In this paper we argued that publish/subscribe middleware is particularly suited for mobile computing. However, several problems must be tackled before currently available publish/subscribe middleware can be used in the mobile environment. Here, we gave a very preliminary analysis of some of the main issues to be considered. These range from the need for redesigning the internal structure of the dispatching mechanism to the opportunity of defining a new semantics for publish/subscribe services, to the need for clearly identifying the requirements for the underlying communication infrastructure. We plan to go further in this analysis. To reduce the complexity of the work, we will start considering the base station scenario identified in Section 3. After analyzing it in details we will move to the other, most complex, scenarios. For each scenario, we will analyze the problems introduced by mobility; then we will formalize the semantics of a possible publish/subscribe middleware to support the given scenario (in particular with respect to the semantics of dispatching); finally we will design and implement a prototype.

References

[1] G. Cugola, “Tolerating Deviations in Process Support Systems Via Flexible Enactment of Process Models”. *IEEE Trans. on Software Engineering*, vol. 24, num. 11, November 1998.

[2] G. Cugola, E. Di Nitto, A. Fuggetta “The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS”. To appear in *IEEE Trans. on Software Engineering.*

[3] R. Mettala. “Bluetooth Protocol Architecture”, white paper, August 1999. http://www.bluetooth.com/developer/whitepaper/whitepaper.asp.

[4] G.P. Picco, A.L. Murphy, and G.-C. Roman. “Lime: Linda Meets Mobility”, In *Proceedings of the 21st International Conference on Software Engineering* (ICSE'99), Los Angeles (USA), D. Garlan and J. Kramer, eds., May 1999.

[5] P. Oreizy et al. "An Architecture-Based Approach to Self-Adaptive Software," IEEE Intelligent Systems, vol. 14, no. 3, pages 54-62. May/June 1999.

[6] C. Perkins. IP Mobility Support. RFC 2002.

[7] G.-C. Roman, G.P. Picco, and A.L. Murphy. "Software Engineering for Mobility: A Roadmap". Invited contribution to the book *Future of Software Engineering*. ACM Press 2000.