Middleware for Next Generation Distributed Systems: Main Challenges and Perspectives

Guy Bernard
GET/INT/CNRS - 9 rue Charles Fourier - 91011 EVRY Cedex -- France
Guy.Bernard@int-evry.fr

Abstract

In the recent past, the scope of distributed systems has drastically changed. Advances in device miniaturization and wireless networking have made distributed systems components evolve from workstations and servers linked by wired networks, to internet of things linked by a variety of networks. Middleware technology designed for legacy distributed systems is no longer suitable to these new kinds of distributed systems. In this paper we describe the main challenges raised by this evolution, with stress put on ad hoc, spontaneous and sensor systems. A survey of the current state of the art will be given, together with major pending issues and perspectives in the domain.

1. Introduction

The scope of distributed systems has drastically changed in the recent past. In the 1980's, the infrastructure supporting distributed applications typically consisted in workstations linked together and to external servers by wired local area networks and long haul networks. In the 1990's, the emergence of PDAs with wireless connection capabilities (typically GSM or WiFi) resulted in a first shift, where terminals have become parts of distributed systems with intermittent connectivity, mobility capability and limited resources. In the 2000's, computing have become more pervasive, with the integration of a variety of smaller and smaller devices (smartphones, smart cards, RFIDs, wireless sensors) into distributed infrastructures, resulting in new requirements in terms of heterogeneity of networks and systems, large scale, and increased mobility. This evolution will certainly continue in the coming years, resulting in "internet of things" where a myriad of (possibly mobile) objects will need to communicate and cooperate both between them and with medium scale or powerful servers. Heterogeneity, adaptability, large scale, spontaneity, will be the main characteristics of these new distributed system infrastructures, with a strong impact on middleware design.

The goal of this paper is to identify the new requirements for middleware design, the main challenges which have to be addressed, the current state of the art, and foreseeable perspectives for middleware research. The paper is structured as follows. The needs for revisiting middleware design are detailed in Section 2. A taxonomy of distributed systems according to physical characteristics and the corresponding requirements is presented in Section 3. Section 4 provides a state of the art of research and development projects dedicated to next generation distributed systems, for the middleware aspects. Finally, Section 5 summarizes the analysis and identifies the foreseeable perspectives in the area.

2. Needs for revisiting middleware design

There are four major reasons for revisiting middleware design [1]: changing environment, inappropriate programming model, software architecture issues, and dynamic reconfiguration requirement. We briefly review these reasons – see [1] for more detail.

As stated before, distributed infrastructure has been changing since the 1980's, when present middleware principles were designed. By that time, distributed systems were typically made of static and powerful terminals and servers, with a wired, permanent and high speed network connection, on a small scale, under the scope of a single management unit. This environment has dramatically changed. Nowadays, the need of enterprise application integration, on a large scale, with various interaction modes, requires loosely-coupled interaction between many application or data sources since a chain of consecutive remote procedure calls is too rigid. Second, the needs of commercial applications in terms of quality of service (response time, availability, security) over a best effort communication environment as Internet is today require the integration of quality of service management into the middleware in order to relieve application programmers of the burden of this management. Third, nomadic mobility of users, with the need of accessing and processing information anywhere and any time, in environments where available resources

Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)
0-7695-2641-1/06 $20.00 © 2006 IEEE

(error rate, network bandwidth, battery) may vary widely and unpredictably, lead to make middleware support the applications to explicitly accommodate these changes in environment. Last, future computing environments will comprise a variety of computing devices (from large servers to microscopic processing units), so that addressing and naming cannot be done with current technologies.

The traditional synchronous client-server programming model used in current middleware platforms is inappropriate to many interaction scenarios that will characterize future distributed systems. Other programming models, such as push/pull publish/subscribe, asynchronous events, shared memory, mobile code and peer-to-peer interaction, each in their way are more suited to support decoupled, flexible and scalable interactions. The challenge will be to make them cohabitate and cooperate in future middlewares.

Software architecture should be revisited too. Distribution transparency, which was the ultimate goal of distributed operating systems in the 1980's, cannot be the foremost goal in nomadic computing and context-aware application. There is a need for selective transparency features, because some changes in environment must be handled at application level, whereas others can be taken into account at middleware level, and yet others have to be managed at user level. Standard software layering (applications above the middleware above the operating system) is too rigid for achieving the required flexibility. Direct interaction between non-adjacent layers should be possible, so that middleware should offer appropriate programming interface elements to applications for passing information between them and the lower layers, and the same in the reverse direction (callbacks). Finally, for integrating small devices with limited resources into distributed environments, customized middleware platforms have to be designed. This can be done through frameworks, but the challenge is to identify the specific patterns that middleware frameworks require.

Dynamic reconfiguration will be a global need for future distributed systems. Mobile terminals can be switched on or off, or out of reach because of lack of wireless covering. This should not be considered as a fault and handled by conventional fault-tolerance mechanisms. Rather, middleware should support disconnected operation (asynchronous communication will be helpful to this end), but a number of issues (consistency, state management, request-response pairing) are still to be addressed. Even if connectivity is maintained, mobile applications encounter dynamic variations in resource supply, such as bandwidth, and must cope with these variations. As stated before, the underlying middleware cannot completely mask these fluctuations – rather, it must monitor the resource supply and demand, compute adaptation decisions, and notify applications if they require adaptation. Strategies for making adaptation still require exploration. Moreover, generalized mobility can result in ad hoc organization, where devices automatically detect others and spontaneously form ad hoc agglomerations. However, existing middleware platforms do not scale to device diversity, population size and runtime dynamics. Finally, low-end devices may be incapable of hosting the complete middleware software, so that the support of an intermediary on a more powerful device is required – again a dynamic configuration need arises, e.g. for manipulating information streams.

3. Taxonomy and requirements of new distributed systems

In this section we characterize the various kinds of next generation distributed systems according to their composition in terms of computing elements and communication relationships. This is because each category has different requirements for middleware design, independently of the conditions of use, purpose or scope of the target distributed applications – for instance, a pervasive home environment may include sensors, ad hoc systems and one or several fixed servers, linked by Bluetooth, WiFi or wired Ethernet. Thus, a taxonomy based on the sole purpose of distributed systems (e.g., pervasive computing, spontaneous networks, sensor networks) would not be discriminating from middleware design point of view. The classification is taken from [2] – please refer to this document for more detail.

In *mobile systems*, terminals offer (relatively) large resources in terms of computing power and memory capacity (typically, a processor at 400 MHz and a 64-128 Mbytes RAM), and are connected to the outside world by wireless links (typically, WiFi or GPRS). This category can be further subdivided into *nomadic systems* and *ad hoc systems* according to whether some nodes in the distributed environment are fixed and some part of the network is wired, or not (no wired network, all nodes are mobile). The scale can range from small (e.g., an ad hoc system made of a few cars approaching a crossroad) to very large (e.g., video transmission of a football match on thousands of PDAs). The requirements of mobile systems in terms of expected middleware functional properties are event notification, mobility and location awareness, addressing, service discovery, and code updating (dynamic changing of protocol or application/middleware code in order to adapt to a new context).

Embedded systems are made of processing components embedded in a "device" (car, plane) with a wired connection to one or more server(s) inside the device. The nodes are not mobile (between themselves) and the scale in generally small. The main requirement of embedded systems in terms of expected middleware functional properties is real-time capability. Because of very specific issues, embedded systems will not be considered any longer in this paper.

Sensor systems consist of sensors per se (capable of providing physical data, such as temperature or pressure) and computing and communicating elements to which each sensor is attached, making a sensor node. Sensor nodes have generally very limited resources, are not mobile, and communicate to a server by a wireless connection. The scale can be very large. The main requirements of sensor systems in terms of expected middleware functional properties are wakeup coordination, crosslayer communication, multi-hop routing, clustering, and distributed task scheduler.

In supplement of functional properties, these three categories of distributed systems require non-functional properties from the middleware layer: heterogeneity (with respect to language, operating system and hardware), openness (capability to extend/modify the system), scalability, disconnection tolerance, fault tolerance, security, adaptability (capacity to dynamically adapt to changes in application needs, user behavior or network parameters), and performance.

4. State of the art

In early 2005, the authors of [2] made a detailed survey of middleware platforms for mobile, embedded and sensor systems, designed by research teams. They have shown that most of the effort has been put on the requirement of adaptability – but the adaptation mechanism remains static during the lifetime of the middleware instance, and that no single middleware platform addresses all the non-functional requirements.

We came to the same conclusion by analyzing middleware platforms designed jointly by partners in European research projects in response to 6th framework program call in late 2003 in the information society technologies area [3]. Most of these projects, started in 2004, are still running so definite conclusions cannot be drawn. However, a detailed look at their work program and the available deliverables gives an insight into their focus. It can be seen that the main target is pervasive computing within the scope of nomadic systems, with the stress on adaptation, and that a few projects address middleware issues for ad hoc or sensor systems.

The analysis of two workshops organized recently by the "Information Society and Media Directorate-General" of the European Commission also gives a highlight about concerns of industrials, academics and European institutions in the domain of future distributed systems. The first one, "Co-operating Objects Workshop" [4], was organized in six sessions according to the contributions received. One session was devoted to wireless sensor networks, another one to middleware for co-operating objects. Among the 43 presentations, only 7 mention the word "middleware". This shows that the role of middleware for designing and supporting (mobile) co-operating objects systems is not commonly perceived. Eight months later, the second workshop ("Pervasive Networked Systems" [5]) shows a slow evolution: 10 presentations among 33 mention the word "middleware". Whereas it appears clearly to academics that middleware will play a key role in software for future distributed systems and that industrials do need middleware for the development of ad hoc systems and sensor systems, there is still a long way to go before this idea is shared by everybody in Europe.

Outside Europe, the emergence of new distributed environments seems to be more clearly perceived and some significant national initiatives have been launched. In Japan, the "U-Japan Strategy" [6] targets "universal communications, next generation networks, new security and safety for the ubiquitous networked society". Following the "e-Japan Strategy" 2001-2005, this initiative puts the stress on sensors, RFIDs, ad hoc networks, and personal area networks. The Korean government follows the same path: the "U-IT839 Strategy" [7], launched in February 2006 is an extension to the "IT839 Strategy", which makes a large place to sensor networks deployment. In US, the NSF GENI ("Global Environment for Networking Investigations") [8] also targets research in pervasive computing, mobile, wireless and sensor networks. We do hope that these national projects will make the right place to middleware research.

As for standardization activities for mobile computing, "the standardization landscape is quite a mesh" [9]. Middleware services are standardized here and there, including tens of forums.

5. Synthesis and perspectives

Middleware design for next generation distributed systems raises a number of issues. These have been identified, but only some of them have begun to receive an answer. The maturity level of research in this area is not the same for the different categories which have been presented. For nomadic systems, the pervasive computing domain has been already heavily investigated; a number of research projects focus on this domain, because it is the simplest one (devices have relatively large resources, mobility is restricted, and the availability of dependable powerful servers simplifies the software design). For ad hoc systems, only partial solutions exist; this domain is more complex (lack of resources, full mobility, full decentralization is needed). Resource limitation (e.g., energy) is not always taken into account, and security deserves more attention. A lot of research work in the area of protocols for ad hoc systems has been achieved, so that this area is now mature, but this is not the same for the middleware layer, in which research effort is only beginning. Finally, sensor systems is a very complex domain, since it spans from electronics to applications, and middleware designers have to imagine completely new solutions to address the issues in this area.

For ad hoc systems, the main challenge is the necessity to design fully decentralized solutions in all areas (e.g., discovery, location, addressing, security, fault tolerance) – the potential contribution of peer-to-peer techniques has to be further investigated. For sensor systems, the major issue is how to distribute computing and storage between sensor nodes and intermediary servers (the right tradeoff between energy consumption for computing and energy consumption for communication remains to be found). For next generation distributed systems in general, the global issue is to which extend can a generic middleware support application design and execution, with a tradeoff between development time (minimized with a generic middleware) end code efficiency (maximized with a specialized middleware).

10. References

[1] Geibs, K., "Middleware challenges ahead", *IEEE Computer*, June 2001, pp. 24-31.
[2] RUNES Consortium, "Survey of Middleware for Networked Embedded Systems", *Deliverable D5.1*, http://www.ist-runes.org/docs/deliverables/D5_01.pdf, January 7, 2005.
[3] Sixth Framework Programme Projects. http://www.cordis.lu/ist/projects/projects.htm
[4] European Commission, "Co-operating Objects Workshop", Brussels, June 23-24, 2005. http://cordis.europa.eu/ist/embedded/workshop_230605.htm
[5] European Commission, "Pervasive Networked Systems – From RFID to the Internet of Things", Brussels, March 6-7, 2006. http://cordis.europa.eu/ist/audiovisual/neweve/e/conf6-70306/conf6-70306_a.htm
[6] Murakami, T., "Japan's National IT Strategy and the Ubiquitous Network", *NRI paper No. 97, Nomura Research Institute*, November 1, 2005.
[7] KBS World, "U-IT839", February 9, 2006. http://english.kbs.co.kr/news/zoom/1381799_11781.html
[8] National Science Foundation, "The GENI Initiative". http://www.nsf.gov/cise/geni/
[9] Raatikainen, K., "Recent Developments in Middleware Standardization for Mobile Computing", *Second Workshop on Context Awareness for Proactive Systems (CAPS 2006),* Kassel, Germany, June 12-13, 2006.