Preprints of the 30th IFAC Workshop on Real-Time Programming and
4th International Workshop on Real-Time Software pp. 7–12

Improving Predictability in Service Oriented Architectures ⋆

Alejandro Alonso * **Emilio Salazar** ** **Jorge López** ***

* *Depto. Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain. (e-mail: aalonso@dit.upm.es).*
** *Depto. de Ingeniería Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain. (e-mail: esalazar@dit.upm.es).*
*** *Depto. Arquitectura y Tecnología de Sistemas Informáticos, Universidad Politécnica de Madrid, Spain. (e-mail: jlopez@datsi.fi.upm.es)*

Abstract: facilitate communication and distributed intelligence across groups of users using different wireless standards. It efficiently supports the communication between humans and embedded systems. The target applications in the MORE project have some time requirements. However, it has not been possible to provide full support to these type of requirements. In this paper, CPU budgets are used to provide some support for meeting this type of requirements and to improve system predictability, when services with time requirements are executed.

Keywords: Real-Time Systems, Service Oriented Architectures, Resource Management.

1. INTRODUCTION

MORE (Middleware for grOup communication and Resource sharing across heterogeneous Embedded systems) MORE (2009) is a project of the 6th Programme Framework of the European Union, which main goals are:

- Facilitate communication and distributed intelligence between groups and users.
- Provide a middleware appropriate for embedded systems that facilitates the relation with persons.
- Provide an API that hides the underlying complexity in the communication between embedded devices to the developer.

The application domains for MORE are heterogeneous systems that must collaborate in order to provide a given functionality to the end user. Two use cases have been used. The first one is a system to monitor physical forest parameters. In this scenario a set of sensors measure a number of field parameters, then send them to a data logger, which delivers them to a central server, where data is analysed to evaluate the current forest state. The other use case is a monitoring system for chronic patients. This system takes measurements of parameters, such as the blood glucose level in diabetes treatment, and sends them to a central server in a hospital. If anomalous values are detected, it is possible to take immediate actions for proper treatment.

The MORE middleware is based on a Service Oriented Architecture (SOA) Dan (2006) Liang (2006). This approach provides means for the automatic discovery of available services in the system. In addition to the basic software in the middleware, the MORE project provides a number of services that facilitates applications development by offering a set of advanced functions such as security, resource management, communication groups management, measurement, remote storage, etc. MORE software is implemented in Java, in order to take advantage of features of the language as portability and available libraries.

System requirements have been based on the mentioned use cases. Some of them are related to safety or real-time. However, given the main goals of the project, it was not advisable to use techniques and or language subsets specially oriented to these type of systems. In order to cope with this lack of support, CPU budgets have been used. They allow for guaranteeing a certain usage time, with a given priority to those threads that executes service operations with time requirements

The goal of this work has been to modify the core of the MORE middleware, using CPU budgets for improving the predictability of applications with the given requirements. The results show an improvement in system behaviour, although further work is needed for getting even better results.

2. MORE MIDDLEWARE

The goal of the MORE project is to develop a middleware and a set of services to facilitate the development of network-centric and embedded applications. The API provided hides the details of the communication protocols

⋆ The work described in this paper has been partially funded by the Spanish Ministry of Science and Innovation, project no.TIC2005-08665-C03-01 (THREAD) and by the European Union within the 6th Framework Programme, project no. 32929.

used in a particular system. Mobile devices, such as smart phones, or PDAs, are normally used for managing the agenda, the business contacts, or for making phone calls. These devices have some common features with traditional embedded devices, such as their small size, and a limited computational capabilities.

MORE allows for the interaction of devices ranging from small embedded devices, that perform functions such as data logging or control, to powerful servers, where data is received and analysed, or mobile devices, that allow accessing this information or receiving alarm notifications. The MORE API facilitates the development of applications requiring this type of interactions.

MORE is based on a Service Oriented Architecture (SOA) for facilitating the communication and distributed intelligence between different groups of users, using different communication standards. MORE relies on *Devices Profile for Web Services* (DPWS) Hyung (2007) Sleman (2008), which defines a subset of the web-services specification (messaging, discovery, service description, and events) oriented towards platforms with limited computational capabilities, as described above. DPWS allows for implementing web-services in small devices. DPWS specification defines an architecture with services that offer a particular functionality, and services that includes other services, that are directly related with the hardware and allows for the dynamic discovery of other services.

Two use cases have been used for defining the requirements of the MORE middleware and for validating the framework:

- Mitigation Management: The goal of this scenario is monitoring forests, in order to determine their state and improve its condition. The goal is to distribute a number of sensors and to remotely get the measurements taken, instead of having to go physically and periodically for getting this data in the field. Sensors are connected to a data logger, that is where the MORE middleware is run, and that sends this data to a central server. An additional advantage is the immediate availability of this information. The main scenarios are the continuous and discontinuous data sampling, remote control of devices, and real-time alarms. The developed software has been validated in laboratory and in-field tests. Final users can access immediately the the measures taken, and getting alarm events, if the obtained values are outside of given thresholds or in emergency situations, such as fires.
- Chronic Care - Health monitoring: The goal is to facilitate the monitoring of patients with diabetes. A sensor takes measures of the blood glucose automatically and send them to a central computer in the hospital, where this data is analysed. In this way, it is possible to monitor de physiological condition of the patient, assess possible risk situations, and perform counselling or tele-education activities. A real experiment is currently being driven. A number of devices have been provided to patients, which are being monitored by means of a MORE application.

The Core Management Service (CMS) is the only mandatory service of the MORE middleware framework. It provides the basic means for managing the services in an application, as well as the basic communication abstractions needed. The CMS is responsible for the services running on a node. It is in charge of starting the initial services in a node and to dynamically manage those that are created during application execution. The CMS relies on OSGi OSGI (2003) Yiqin (2008) for the implementation of the main part of its functionality.

The Group Management Services (GMS) is one of the relevant innovations in MORE. It is responsible for the administration of groups and their members. It provides a decentralised point of contact for MORE services to create, delete, modify, query a specific group, and to send messages to a group. A policy engine is used in conjunction to the GMS. It can make decisions based on data sent from the GMS. In this way, it can create new groups to handle a given situation (set of doctors for managing a patient) or distributed messages to a group for keeping members aware of the evolution of a problematic situation.

Applications based on the MORE middleware may involve exchange and transmission of personal and sensitive data across several communication protocols. It is therefore critical that MORE provides facilities to protect the privacy rights of individuals and their personal information space, and also has the capability to prevent unauthorised access and use of restricted data and services. The analysis of the MORE requirements shows that two principles should be followed in the conception of the security in MORE:

- The required security level is not the same for every MORE application. Hence, the security support to be provided should be configurable.
- In a MORE system there will be devices with a computational capabilities ranging from powerful servers to tiny embedded devices. The overheads of the security services should be reasonable and not consume excessive system resources.

The proposed approach relies on a set of security services whose cooperation helps to fulfil the security requirements. Applications may only use those security functions needed. The security services are:

- Cryptographic Service (CS): provides signing, enciphering and deciphering algorithms to other services or applications.
- Identity Provider Service (IPS): is responsible for verifying the identity of any user who demands it. Upon successful identity verification, an authentication ticket is produced for future secure interaction.
- Permission Management Service (PMS): is responsible for storing information about users' and groups' privileges for accessing services.
- Key Distribution Service (KDS): is a trusted service, which helps distribute keys in a secure manner to any user/application/service.

The CS is the only mandatory service for any device that requires any security function. The rest can be executed in more powerful servers and accessed remotely.

The MORE project offers a set of additional services for making it easier applications development. The Resource Management Service is responsible for observing

and controlling a device or system state in order to make adjustments for improving power consumption or resource usage. Measurement services allows for requesting or sending data from remote devices. Some of the implemented services include a generic measurement framework for easy attachment of new proprietary devices and sensors, and specific services for gas measurement or blood glucose value taking. Other implemented services include a local or remote storage service, a compression service, a trigger and a configuration service.

In summary, the MORE framework makes it possible to develop advance applications where different devices interact in a seamless way, intelligence is distributed over a set of embedded devices, and where users can access information remotely and immediately.

3. REAL-TIME SUPPORT IN MORE

Safety systems Lee (2008) are usually defined as those which failure may cause important damages to the environment and to persons. There are some requirements in MORE, that can be catalogued as such. However, the development of safety systems implies a set of very strict design and development rules. Among them, it can be required the use of specific language subsets that allows for performing a set of exhaustive tests to ensure proper system operations. There are language features that prevent some of these tests, and, hence, they must be avoided. There is currently an activity to define a safety Java language profile. However, it will not be suited for most of the MORE target applications. This is one of the main reasons why it has not been possible to include support for developing safety systems in the context of the MORE middleware.

Time requirements are also common in systems with some safety implications. Real-time systems are those which correctness depends not only on their outputs, but on the time when they are produced as well. A good result that is obtained late can be a failure in this type of systems. It is important to note that the main issue is not about performance, but on predictability of the timeliness of the results. The MORE requirements mentioned above also implies time requirements.

The development of real-time systems highly predictable implies the use of specific techniques and protocols: languages with appropriate programming constructs, operating systems with real-time scheduling algorithms, communication protocols, careful designs avoiding unbounded delays, etc. The use of these approaches is out of the scope of MORE, which major requirements are not of this type. In addition, the use of these techniques would have implications such as the lack of proper libraries, low general use of communication protocols, or lack of standards for the intended application domains.

However, it has been found that it is possible to provide some support for improving system predictability within the MORE middleware. One facility that is helpful is the use of CPU budgets. A budget is characterised by a period, a usage time and a priority. A thread is allowed to execute during the usage time every period with the given priority. If the budget is exhausted, some actions must be taken in order to ensure that other threads can use its budget. In MORE the priority is changed to that of a standard thread. The usage time is replenished periodically. Real-Time Java Specification provides real-time threads with a behaviour similar to that described. The main innovation of this work is using this facility on a standard Java Virtual Machine (JVM).

The origin of this contribution was twofold. First, there is an implementation of a couple of modules (Budget Accountant, BACC, and Run-Time Control, RTC) that provides CPU budgets to threads Alonso (2004). There is an implementation on Linux running on a single core PC. This implementation provides a C interface and relies on some real-time facilities available in the Linux kernel.

Second, JamVM is open source Java virtual machine and, after a deep study of the source code, it was noted that there is a one-to-one relation between a Java thread and a Linux (POSIX) thread. Hence, the approach would be to assign the budget to Java thread. Some successful experiments have also been done with the Sun JVM. It appears that also follows the one-to-one model, although it has not been possible to check with absolute confidence this fact.

As a consequence, the implementation performs calls to the RTC functions from the Java code. The following additional activities have been performed for providing this functionality:

- Porting of the BACC and RTC modules to version 2.6.21 of Linux. The Linux kernel has also been slightly modified to call the BACC when a context switch occurs.
- Development of classes in JamVm for providing developers with access to RTC functions from Java.
- Analysis and modification of the MORE core, for assigning budgets to the threads involved in the handling of an invocation to a service with predictability requirements. In order to ensure a proper response, it is not only necessary to assign proper budgets to application threads, but to the middleware ones as well. The lack of detailed documentation of the DPWS stack has posed some problems of that activity. Finally, the threads (usually one) involved in the handling of a service invocation has been changed accordingly.

4. CPU BUDGETS

The motivation for this work is to develop a software component able to add resource usage accounting to an operating system that provides basic real-time features. The required extensions should provide the following services to support the above described model:

- *Budget management*: It includes assignment of budgets to tasks and accounting for the resources usage of the tasks.
- Detection of *Budget overruns* and execution of the appropriate handling actions.
- Keep *Consistent information* about the relations between, clusters, tasks, resources, etc. This is specially important when clusters are used.

- Acquisition of *Statistical information* about the resources used by each task or cluster.

The accounting and enforcement functions are based on clocks associated to resource time usage. Tasks' budget is refilled to its original value following a given pattern, that usually is periodic. The following main events drive the operation of the target software component:

- Interrupts from a timer, to indicate that a budget has expired and hence, an overrun has occur.
- Notification that the task using a resource has changed.
- Request from the user, to set the system characteristics, to ask for information or to set the behaviour of the component.

The BACC module provides the basic functions for accounting for threads resource usage. It detects overrun situations and notifies this event to upper layers. It provides functions for configuring its behaviour: activation of monitoring functions, activation of accounting for each individual thread, or configuration of the action to take when a budget is exhausted (overrun). The operating system kernel has to call a special function in the BACC whenever a context switch occurs. In order to implement this behaviour, it has been needed to modify the kernel. It has also been required to implement timers with higher precision than the standard one provided by Linux.

The RTC module executes operations with higher abstraction level. The most important ones are the periodic execution of replenishment of the CPU budgets and handling the event where a thread exhausts its budget. In this case, its priority is reduced.

The Linux implementation of these modules relies on the use of a fixed-priority preemptive scheduler, which is provided by the kernel. It is compliant with POSIX specification. It is possible to define the scheduler that will handle a particular thread. If the mentioned scheduler is the one selected, then the managed thread gets some real-time properties. For example, these threads have higher priority than standard ones.

The implementation that has been used in this work is ported to the 2.6.21 Linux version, running on a PC with a processor with one core. A version for a ARM processor is under development.

5. REAL-TIME SUPPORT IN THE MORE CORE

The Core Management Service (CMS) is the most important component in the MORE middleware. Section 2 describes its main features. Figure 1 shows a simplified version of the MORE architecture, where this service is integrated. In this figure are only included those components that are relevant for describing the required changes for improving predictability.

The CMS (called eu.more.core.cardinal in the software distribution) is the central component in a MORE system. All nodes must execute this service. From the behavioural point of view, it relies on a thread pool for the execution of the remote invocations to services in the given node. An in-depth study of the code, shows that it is based on a set of threads that are created at system initialization,

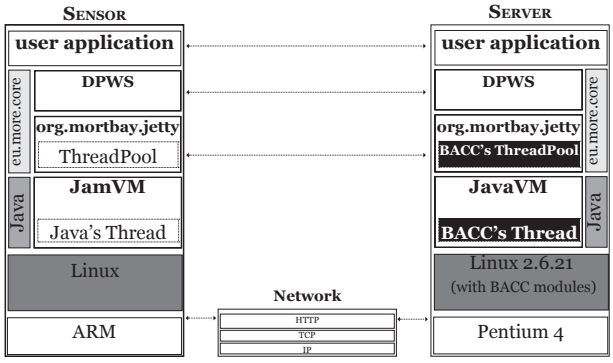

Fig. 1. Architecture of a MORE system

which aim is to execute clients' requests. These threads are suspended, until a new request arrives. Then, one of them is activated for handling it. Finally, the requested service is executed. Given that all of this handling is performed by the same thread, the class that serves for creating them has been modified in order to use the threads provided by the BACC, instead of the standard ones.

The ThreadPool class includes the implementation of a threads pool that uses the CMS for handling the SOAP messages that are received and the responses. It is a class that belongs to Apache org.mortbay.jetty package. It is called from the implementation of the DPWS in the core.

A class has been included that encapsulates the Java code required for performing calls to the methods in the RTC module, which is the external interface to the CPU budgets, and that is coded in C. This code, that has a very low level, perform operations performs type conversions and parameter passing, required for the use of the shared libraries.

In summary, the most important modification has been the change of the ancestors of the ThreadPool class, in order to extend the BACC threads in Java, instead of the native ones. BACC threads requires a more complex initialization. Special code has been added for hiding this features to the applications programmer.

6. SYSTEM TESTING

6.1 Testing CPU budgets

A large number of executions has been performed in order to validate the implementation of the CPU budgets. Initially, the Linux kernel modules developed (RTC and BACC) has been validated. For this purpose, a number of tests has been prepared using different programming languages: Ada, Java y C.

Figure 2 shows the execution of a synthetic workload with a real-time thread (RTC, solid line) and another one with a standard Linux thread (NO_RTC, dotted line). This is a sample of a large number of executions showing similar results. The code of the synthetic load that is executed periodically (100 times in each test run) is the following:

```
for i in 1 .. 50000000 loop
    res := Sqrt (numero_grande);
    res := res * res / (res - 1.0);
end loop;
```

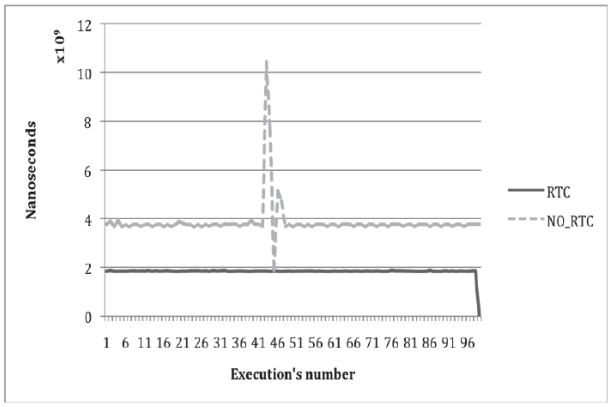

Fig. 2. Comparison of the execution of a standard and a BACC threads

The execution of the synthetic load of the BACC thread was more constant, stable and fast. In addition, it can be watched that the standard thread is interfered by other threads and by the operating system. The BACC gets nearly no interference.

These tests serves to show the behaviour of the implementation of the CPU budgets in a system with low load. However, the main goal is to ensure that the BACC allows for the predictable execution of real-time threads, even in the case system high loads. In order to validate this case, the same tests were executing running at the same time a large number of threads and some CPU consuming operations, such as a kernel compilation. As a result, system load during test execution was always close to 100%.

Figure 3 shows in this case the proper execution of the BACC threads. The results are always better when using them than in the case of Linux threads. As it is shown in the figure, the execution times of BACC threads are nearly constant. In fact, the difference of their execution times are less than 1% with those got when there is no high system load.

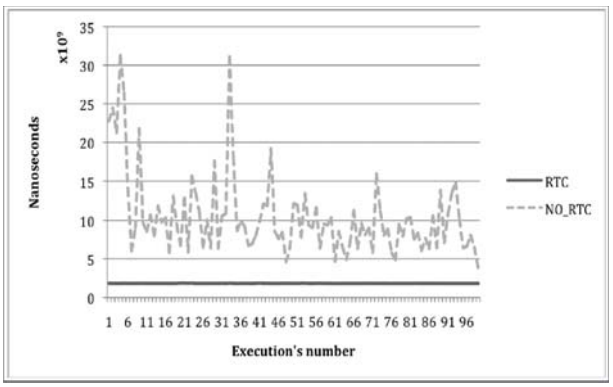

Fig. 3. Execution of BACC and native thread in a system with high load

On the other hand, Linux threads shows a results that are not appropriate when requiring some predictable behaviour. The mean execution time is much higher (around 300%) with respect to the values got with the case with low system load. In addition, the execution time is not predictable, and has large execution peaks. The measured difference between the minimum and maximum execution time is up to three orders of magnitude.

In summary, the execution of the BACC is correct and BACC threads shows a much better behaviour in terms of time response and predictability.

6.2 Validation of the MORE middleware

The next step in the validation of the modified version of the CMS, with the changes required for ensuring that remote invocations are handled by BACC threads. The testing platform was composed by two computers running MORE application. In one of them is executed the service with time requirements (e.g. handling a fire alarm) and the modified version of the CMS. The client is running in the second computer. It performs periodic invocations to a method of the urgent service. This method executes the code below, in order to simulate a certain CPU load. The synthetic load and running Linux programs caused a CPU load close to 100% during the tests execution.

```
while (i != 100)
   {
        SecureRandom random = new SecureRandom();
        new BigInteger(8, random);
        i++;
   }
```

There were problems when running the modified MORE core with the JamVM. The reason seems to be a library that performs some kind of busy wait in a communication operation. The source code of JamVM and related libraries is being inspected to detect the cause of the problem. The tests shown below were obtained using the Sun virtual machine. The tests using BACC in conjunction with this VM showed good results, although there is no way to inspect the source code to have a clear knowledge on issues, such as the relation between Java and native threads.

The result of a representative test run with this configuration is shown in 4 (RTC: modified core, NO_RTC: original core). It can be shown that in both cases there are peaks in the execution time, which implies interferences by other programs running in the computer. The interference of the Sun VM seems to be an important reason for this interferences. Although the reasons are not fully clear, due to the lack of information on the internals of the VM, the garbage collector is likely to be one of the most important sources of interference.

In any case, the results of the modified core are significantly better. When using standard threads, the response time of the service operation is 15% greater than its worst case execution time in a 80% of the test loops. When using BACC threads this figure is 20%. This result is consistent with a large number of repetitions of this test cases.

In summary, it can be concluded that the execution with BACC threads is much more predictable than standard threads. The modifications in the CMS have allowed to increase the predictability of the CMS and MORE applications. It is obvious that these results cannot be compared with the use of a real-time Java virtual machine. However, a certain support to services time requirements is included in the MORE core.

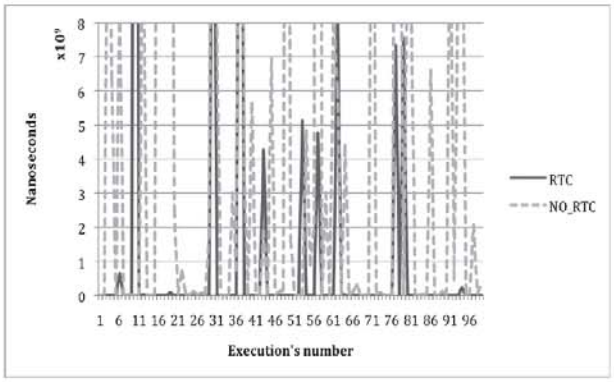

Fig. 4. Results of test with the original CMS and the modified CMS

Current activities related with this work are oriented towards solving the problems with the JamVM, in order to have a MORE detailed knowledge on the interferences that suffered and how to reduce them. The availability of the source code of the virtual machine is basic for accomplishing this goal.

7. CONCLUSIONS

The MORE middleware is an excellent platform for developing distributed applications, characterized by the interactions between embedded devices, mobile platforms and central servers. This framework provides mechanisms for the dynamic creation of services, automatic service discovery, and invocation in a transparent way with respect to the communication protocol. The provided services constitutes an appropriate basis for developing applications in the target domains in this project.

The use cases in this project have some time requirements. However, it has not been possible to include full support for handling them properly. In this work, CPU budgets are used as a mechanism for improving the response time predictability in the system. BACC and RTC are two Linux modules that allows to guarantee CPU shares to threads. A number of classes has been developed in order to access these facilities from Java threads. In this way, it is possible to provide some real-time support to urgent services, within the limitations of the used execution platform.

The MORE core (CMS) has been modified in order to assign CPU budgets to services handling urgent invocations. The results of the tests show that the time predictability has been greatly improved. Future work is focused towards improving this implementation: use only BACC threads in the core when dealing with urgent events, and improving even more time predictability.

ACKNOWLEDGEMENTS

The work described in this paper has been partially funded by the Spanish Ministry of Science and Innovation, project no. TIC2005-08665-C03-01 (THREAD) and by the European Union within the 6th Framework Programme, project no. 32929. The MORE consortium is composed by ProDV (Germany), Thales Communications (France), ALL (Hungary), Waterford Institute of Technology (Ireland), University of Dortmund (Germany), University of Debrecen (Hungary), University of Dresde (Germany) and Universidad Politécnica de Madrid (Spain). The authors wish to acknowledge the contribution of the members of the MORE consortium in the development of this work.

REFERENCES

Alejandro Alonso, Alejandro Sánchez-Rico, Miguel Lobo, José Ruíz. Portable Component for Resource Management In *28th IFAC/IFIP Workshop on Real-Time Programming*, 2004. Istambul, Turkey, 2004. Editor: Wolfgang A. Halang, Elsevier Science, 2004.

Xie Dan, Ying Shi, Zhang Tao, Jia Xiang-Yang, Liang Zao-Qing, Yao Jun-Feng, An Approach for Describing SOA In International Conference on Wireless Communications, Networking and Mobile Computing, 6. WiCOM 2006, págs. 1-4.

I. Lee, J. Leung, S. Son. Handbook of Real-Time and Embedded Systems. Chapman Hall, 2008

MORE projecet web page, https://wwws.ist-more.org/

The OSGi Alliance (2003), OSGi Service Platform, Release 3, IOS Press, pp. 604, ISBN 1586033115

A. Sleman, R. Moeller Integration of Wireless Sensor Network Services into other Home and Industrial networks; using Device Profile for Web Services (DPWS) Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, 2008, págs. 1-5.

Hyung-Jun Yim, Il-Jin Oh, Yun-Young Hwang, Kyu-Chul Lee, Kangchan Lee, Seungyun Lee Design of DPWS Adaptor for Interoperability between Web Services and DPWS in Web Services on Universal Networks, International Conference on Convergence Information Technology, 2007, pg. 1032-1039.

Lu Yiqin, Yuan Yao, Sun Yingkai, Yang Xiaodong An approach to service integration in the OSGi architecture of home networks 11th IEEE Singapore International Conference on Communication Systems, 2008, pg. 756-760.

Liang-Jie Zhang, Tutorial: SOA and Web Services International Conference on Web Services, 2006.