Interconnected Embedded Systems: Challenges and Main Problems to Solve*

Carlos Almeida
Instituto Superior Técnico -
Universidade Técnica de Lisboa,
Avenida Rovisco Pais,
1049-001 Lisboa, Portugal.
cra@comp.ist.utl.pt

José Rufino
Faculdade de Ciências da
Universidade de Lisboa,
Campo Grande - Bloco C8,
1749-016 Lisboa, Portugal.
ruf@di.fc.ul.pt

Abstract

Small scale embedded systems and their interconnection constitutes an hot topic in nowadays. With the technological evolution achieved in recent years, there is the possibility of having a diversity of pervasive applications with every increasing requirements. But, in order to fulfill this potential, many research work is still needed. In this paper we give an overview of the main characteristics and limitations of these systems and highlight some areas of related research that need to be addressed in order to satisfy applications requirements. We briefly present solutions and ongoing work that we are doing in operating system support for concurrency, real-time, power management, and study and analysis of several different communication technologies to interact with embedded systems.

1. Introduction

The technological evolution that the last few years have been watching, in what concerns electronic devices of small size with processing and communication capabilities, creates the potential for the development of new applications in several different domains. The improvement of processing and communication capabilities, associated with the reduction of size and cost, allows the proliferation of an almost infinity of small embedded systems, that can be interconnected, contributing to a common global goal. Areas such as industrial control, automotive, home automation, surveillance systems, sensor networks applied to the monitoring of wild life, environment, or buildings, are examples of such potential and ubiquity.

Interconnected embedded systems is thus an hot topic and fast growing research area. There are, however, several problems that need to be addressed in order to satisfy applications requirements. Those problems are related to both the ever increasing applications requirements and the restrictions associated with the environment under consideration. Some of the main problems to solve are related to the creation of conditions making it possible to: ensure real-time requirements; interconnection and communication facilities; safety, security and intrusion-tolerance; and power consumption reduction so as to increase the autonomy of systems dependent on batteries. Besides that, the resolution of most of these problems is still conditioned by additional restrictions caused by the shortage of resources due to the need of satisfying other requirements, namely, cost and system size.

This paper is organized as follows: in the next section we present the main characteristics and limitations of embedded systems; in Section 3 the main areas of related research are presented; in Section 4 we give an overview of some proposed solutions and describe ongoing work. The paper ends with the conclusions and references to future work.

2. Characteristics and limitations of embedded systems

An embedded computational system is characterized by being a system where the existing computational element(s) are integrated with the application, that is, they have a specific function related to the given application, and are not computers for generic use. Depending on the specific application, a given embedded system may present a set of specific characteristics related to that application. However, in a generic way, most embedded systems show a set of common characteristics/requirements. For example, they need to be concerned with:

- the right functionality and performance desired for the application, which may include a combination of real-time, reliability and fault-tolerance guarantees;
- power consumption management, mainly in situations where batteries are used, in order to increase autonomy;
- keeping, in many scenarios, system size small;
- ensure that the cost keeps low, mainly in situations where the number of units is very large.
- and all of this in scenarios of low resources, including CPU processing power and available memory.

Besides that, when the embedded systems are interconnected, the aspects related to communications are also

*This work was partially supported by FCT, through Project POSC/EIA/56041/2004 (DARIO).

very important. When the embedded system corresponds to a small node, for example in a network of sensors, the way the communication is handled has a significant impact in the overall system. Depending on the used technology, we will have different characteristics in what concerns power consumption, bandwidth, covered distance, and cost, for example.

Having a communication network also introduces other concerns such as security problems. Confidentiality may, or may not, be an important aspect, but integrity and intrusion detection/tolerance that might cause denial-of-service, are concerns that can not be ignored in many scenarios.

Another aspect related to communications and remote access to information, is the possibility of having support to handle mobility and connected / disconnected operation, which may be a very important issue when in association with power consumption concerns.

3. Main areas of research

Finding solutions to the problems presented above, requires the resolution of a comprehensive set of challenges. The related research includes both processing and communications. And, in most cases, there are several aspects were an overall approach must be used because the problems cannot be solved in an isolated fashion – the interaction between processing and communications must be taken into account. That is the case of power consumption management and security, for example.

3.1. Processing

Many embedded systems are small in size and composed by rather simple elements, including the processor (CPU). This poses a problem with respect to operating system support. There might not be enough resources and processing power to use a traditional operating system. On the other hand, the support provided by the operating system, namely concurrency and scheduling support, would still be desirable for most applications, thus avoiding the extra complexity in application development.

Even real-time multitasking kernels (at least most available ones) might be too heavy, in terms of memory footprint and processing power required, for small microcontrollers that are used in those embedded systems. In those situations, what is needed is a very small kernel with minimal functionality.

A recent and well known operating system for small sensor network nodes is TinyOS [3]. It addresses many important issues for this type of systems, and significant research has been done, but there are still lots of improvements that can be introduced. In some situations we need even simpler systems for very small microcontrollers. In other cases we would like to have improved support to handle aspects such as specification and validation of real-time requirements, scheduling offering timeliness guarantees and power consumption management, for example.

There are some public domain real-time multitasking kernels such as eCos [1] and RTEMS [2] that are adequate for many embedded systems (medium scale) but too big for others. Anyway, another aspect concerns the support for the interaction between real-time systems and general-purpose systems. Some functional restrictions may imply the coexistence between components with real-time requirements and general-purpose components. In these scenarios it is of utmost importance to obtain a confinement of possible timing faults at the generic components level, in such a way that they not jeopardize the operation of the real-time components. This scenario implies, on one hand, the need to create "two virtual machines" with different characteristics: one offering guarantees in the temporal domain, and another one for other generic operations. On the other hand, being it necessary to have interaction between those "two virtual machines", that interaction must be carefully addressed so as to not jeopardize temporal aspects. Obtaining those "two virtual machines" may imply a careful and controlled use of interrupts (as is done in rt-linux [4]), and/or the use of additional processors such as the ones provided by symmetrical multiprocessors (SMP), Dual-Core and Hiperthreading architectures.

3.2. Communications

In what concerns communications, the restrictions in resources usually associated with the type of environments that we are addressing, such as sensor networks, makes it very difficult to build a system that satisfies all the desired requirements. Although there are some embedded systems that are connected through a wired network to provide remote access, in most cases this interconnection is done using wireless networks and with severe restrictions in power consumption.

This is an active area of research and in recent years much work has been done, but there are many aspects in the interconnection of embedded systems where there are still challenges to solve and/or the need to improve existing solutions. Besides choosing the technology that best fits in a given scenario, taking into consideration aspects such as covered distance, supported bandwidth, possible interferences, power consumption, size and cost of communication devices, it is also necessary to address several other issues.

The topology of the network may play an important role, for example. Having an *ad-hoc* network or an hierarchical structured network with access points, will have implications on the way communications are structured, namely in routing algorithms and procedures to overcome the short distances covered by individual nodes. Other issues are related to access control and quality-of-service (QoS) support. The solutions to these problems may also depend on the used technologies. Some wireless communication technologies suitable for the interconnection of embedded systems are: Radio Frequency Identification (RFID), infrared (IrDA) [5], ZigBee[6], Bluetooth[7] and

WiFi (802.11)[8].

3.3. Aspects involving both processing and communications

Power consumption

Power consumption, although a very important aspect in the case of systems that are not connected to a power grid, and that need to have a significant autonomy in order to be useful, is a subject that can not be addressed in a completely isolated fashion. It must be addressed in conjunction with the other problems. We want to obtain a minimal power consumption, but one that grants the desired functionality. Otherwise, there would be a trivial solution consisting in turning off the system!

The solutions to obtain, imply a trade-off between several goals, that might be antagonistic. For example, in order to meet a given deadline, it may be necessary to use more processing power, which may imply an increase of power consumption. A correct management of these resources (CPU, power) implies a tight interaction between process scheduling and power consumption management modules.

The use of power saving modes of operation (sleep modes) that might turn off temporarily unused functional units, and changes in power voltage and CPU clock frequency (e.g. dynamic voltage scaling [9]) are examples of techniques to improve autonomy without sacrificing the desired functionality.

There is also a trade-off, in power cost, between processing/storage and communication. The cost in power to transmit information is usually much higher than basic processing operations on that information. So, in some cases it might be interesting to consider some form of information pre-processing to reduce the amount of data to transmit. On the other hand, this may imply more processing power and more memory to handle the data.

Anyway, the solution to the power consumption problem should be addressed at several different levels: application, middleware, operating system (scheduling) and device drivers. Whenever possible, standard approaches should be followed (e.g. ACPI (Advanced Configuration and Power Interface) [10]).

Security

From the point-of-view of security there are several aspects to be addressed. Besides a secure access, making sure that only authorized accesses are allowed, it is also important to detect possible attempts of unauthorized access (intrusion detection), and create the conditions to protect critical operation, avoiding denial-of-service. This implies the existence of protection mechanisms ("firewalls") able to ensure correct operation despite those attempts. This "correct" operation must be achieved in several different domains, including temporal domain (real-time guarantees) and power consumption domain (reserving power for critical operations).

Another aspect is related to secure remote loading. When the system boots from a remote site, or there is the need to remotely update or load new versions, it is important to be able to make these operations in a secure way, avoiding potential external interferences. For example, the identity of servers and clients must be preserved.

Other aspects

Besides the issues described above, there are some other aspects that can be addressed if one wants to improve generic operation or solve specific problems. For example, if we have a sensor network we may want to try to have an uniform power consumption in order to have the same longevity for all nodes. That may imply specific algorithms for route diversity, load balancing and task allocation. On the other hand, if the objective is to find the shortest path between a given node and an access point, the approach to use may imply different work loads for different nodes.

An integrated scheduling of processing and communication activities is also an important issue to achieve global goals that may even address aspects such as connected / disconnected operation.

4. Overview of solutions and ongoing work

We have been addressing some of these problems and are currently doing research work in these areas. In what concerns operating system support for small embedded systems, taking into account public domain systems such as TinyOS [3] and uCos [11], we are specifying and building a small kernel, with minimal functionality, able to run on small microcontrollers (e.g. PIC family from Microchip [12]). This minimal functionality is at least a simple support to task concurrency and interrupt handling. It will work as a cyclic executive calling the procedures associated with tasks and when necessary handling interrupts and interrupt handlers. Depending on the available resources in a given microcontroller, this minimal functionality can be extended with support for periodic tasks, with specification and monitoring/handling of real-time requirements, and power consumption management, using power saving modes of operation when available (a more detailed description is in preparation and will appear in a future document).

Another complementary work that we have been doing concerning operating system support is the introduction of extensions in existing real-time multitasking kernels to control external event handling and ensuring real-time guarantees in the presence of event overload [13]. This work was done using the public domain real-time multitasking kernel RTEMS [2], where we also introduced a new console called VITRAL, a simple text mode windows manager that also includes the event control referred above, associated with the keyboard [14].

In what concerns power consumption management, we have already done some preliminary work at an higher level using a laptop running Linux, where we developed

an infrastructure with servers able to deal with application requirements and tuning the system to increase autonomy without compromising the desired functionality. We are now addressing the consumption problem in the context of smaller embedded systems.

Remote interaction with an embedded system and wireless communication support is also an important issue that we are addressing. Besides providing remote access using a standard internet connection and building a HTTP server, we are also looking at providing that access through a wireless connection using Bluetooth.

In order to have a better understanding of possibilities and restrictions in several different scenarios, we are also conducting a study to analyze several wireless communication technologies, such as RFID, IrDA, ZigBee, and Bluetooth, suitable for small embedded systems. The idea is to be able to choose the technology that best fits a given set of requirements, with the possibility of specifying a hierarchical structure taking advantage of specific characteristics, namely in what concerns: distance, interferences, bandwidth, power consumption, and cost.

Another important issue is related to the provision of real-time guarantees and support for quality-of-service (QoS) adaptability. This may require mechanisms for access control. We have done some preliminary work with Bluetooth using a server to control QoS, and increase real-time guarantees for specific clients.

From the point-of-view of security, we started with the addressing of aspects related to remote system boot. A DHCP server and clients were enhanced to overcome potential security problems such as denial-of-service (server availability), wrong client identity, and false server, for example. However, this is a topic where there is still much research work to be done.

Servers for embedded systems

Another issue deserving some attention is related to the generic use of portable embedded systems, of small size, such as PDAs or cellular phones, whose autonomy depends on the management of power consumption supported on batteries.

On one hand, the ubiquity of such devices makes it desirable to have an increase number of applications with every increasing functionality and requirements. On the other hand, the existent limitations at the several levels, related to the lack of available resources, due to the desired small size, low cost and low power consumption, may affect in a very significant way the application functionality that is possible to offer.

One possible solution to minimize this effect consists in resort to the use of assistant servers. Not having directly the needed resources to offer a given quality-of-service, one may recur to other entities ("outsourcing") to obtain those services, thus increasing the functionality and reducing power consumption. Some possible services are:

- screen service – to overcome the small size of screens available on portable devices
- storing service – for large amounts of data and repositories
- processing service – for more demanding computational operations
- agent ("proxy")
  - to access communication networks
  - to support connected / disconnected operation

This way, the portable device can thus be used essentially as an access device, easy to carry around, of generic use, allowing a "kind of" specialized remote control, that makes it possible to access a diversified set of services.

5. Conclusions and future work

The area of interconnected embedded systems is a very important and active research area. Although there has been several contributions in recent years, there are still many problems to be solved, and solutions to be improved. This is particular true due to the ever increasing application requirements, and limitations of supporting environments.

The study of available technologies allows us to have a better knowledge of their characteristics in order to be able to use them in the best possible way. However, in order to solve existing problems and challenges, there is still lots of research work to be done. This work concerns both the adaptation of previous work to these new environments, and finding new ideas to address specific problems.

References

[1] Red Hat, Inc., *eCos Reference Manual*, 2003.

[2] On-Line Applications Research Corporation (OAR), *RTEMS C User's Guide*, edition 4.6.2, for rtems 4.6.2 edition, August 2003, (The RTEMS Project is hosted at http://www.rtems.com.).

[3] "TinyOS", http://www.tinyos.net.

[4] M. Barabanov and V. Yodaiken, "Introducing Real-Time Linux", *Linux Journal*, vol. 1997, no. 34, February 1997.

[5] http://www.irda.org.

[6] http://www.zigbee.org.

[7] http://www.bluetooth.com.

[8] http://www.wi-fi.org.

[9] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems", in *ACM Symposium on Operating Systems Principles*, 2001, pp. 89–102.

[10] http://www.acpi.info.

[11] J. J. Labrosse, *MicroC/OS-II: The Real-Time Kernel*, CMP Books, 2002.

[12] Microchip Technology Inc., "PIC18FXX2 Data Sheet", http://www.microchip.com, 2002.

[13] M. Coutinho, J. Rufino, and C. Almeida, "Control of Event Handling Timeliness in RTEMS", in *Proceedings of the 17th IASTED International Conference on Paralel and Distributed Computing Systems - PDCS 2005*, November 2005, Phoenix, Arizona, USA. IASTED.

[14] M. Coutinho, J. Rufino, and C. Almeida, "VITRAL: A Text Mode Windows Manager for RTEMS", in *Terceiras Jornadas de Engenharia de Electrónica e Telecomunicações e de Computadores*, Novembro 2005, Lisboa, Portugal.