

New technology

Towards Networking of Embedded Systems

Luciano Lavagno¹, Richard Zurawski²

¹Cadence Berkeley Labs, USA & Politecnico di Torino, Italy, ²ISA Group, USA

Abstract

The focus of this short article is on networking of embedded systems. It briefly discusses rationale for emergence of this kind of systems, their benefits, types of systems, diversity of application domains and arising from that requirements, as well as security issues. Subsequently, the chapter discusses the design methods for networked embedded systems, which fall into the general category of system-level design. The methods overviewed focus on two separate aspects, namely the network architecture design and the system-on-chip design. The design issues and practices are illustrated by examples from automotive application domain. References are provided to explore further the area, and specific application areas.

Networking of Embedded System

The last two decades have witnessed a remarkable evolution of embedded systems from being assembled from discrete components on printed circuit boards, although, they still are, to systems being assembled from IP components "dropped" on to silicon of the system on a chip. Systems on a chip offer a potential for embedding complex functionalities, and to meet demanding performance requirements of applications such as DSP, network and multimedia processors. Another phase in this evolution, already in progress, is the emergence of distributed embedded systems; frequently termed as networked embedded systems, where the word "networked" signifies the importance of the networking infrastructure and communication protocol. A networked embedded system is a collection of spatially and functionally distributed embedded nodes interconnected by means of wireline or/and wireless communication infrastructure and protocols, interacting with the environment (via a sensor/actuator elements) and each other, and, possibly, a master node performing some control and coordination functions, to coordinate computing and communication in order to achieve certain goal(s). The networked embedded systems appear in a variety of application domains to mention automotive, train, aircraft, office and commercial buildings, and industrial [1], [2] - pri-

marily for monitoring and control, environment monitoring, and, in future, control, as well.

There have been various reasons for the emergence of networked embedded systems, influenced largely by their application domains. The benefits of using distributed systems and an evolutionary need to replace point-to-point wiring connections in these systems by a single bus are some of the most important ones.

The advances in design of embedded systems, tools availability, and falling fabrication costs of semiconductor devices and systems have allowed for infusion of intelligence in to field devices such as sensors and actuators. The controllers used with these devices provide typically on-chip signal conversion, data processing, and communication functions. The increased functionality, processing and communication capabilities of controllers have been largely instrumental in the emergence of a widespread trend for networking of field devices around specialized networks, frequently referred to as field area networks.

The field area networks, or fieldbuses (fieldbus is, in general, a digital, two-way, multi-drop communication link) as commonly referred to, are, in general, networks connecting field devices such as sensors and actuators with field controllers (for

instance, programmable logic controllers (PLCs) in industrial automation, or electronic control units (ECUs) in automotive applications), as well as man-machine interfaces, for instance, dashboard displays in cars.

In general, the benefits of using those specialized networks are numerous, including increased flexibility attained through combination of embedded hardware and software, improved system performance, and ease of system installation, upgrade, and maintenance. Specifically, in automotive and aircraft applications, for instance, they allow for a replacement of mechanical, hydraulic, and pneumatic systems by mechatronic systems, where mechanical or hydraulic components are typically confined to the end-effectors; just to mention this two different application areas.

Unlike LANs, due to the nature of communication requirements imposed by applications, field area networks, by contrast, tend to have low data rates, small size of data packets, and typically require real-time capabilities which mandate determinism of data transfer. However, data rates above 10 Mbit/s, typical of LANs, have become a commonplace in field area networks.

The specialized networks tend to support various communication media like twisted pair cables, fiber optic channels, power line

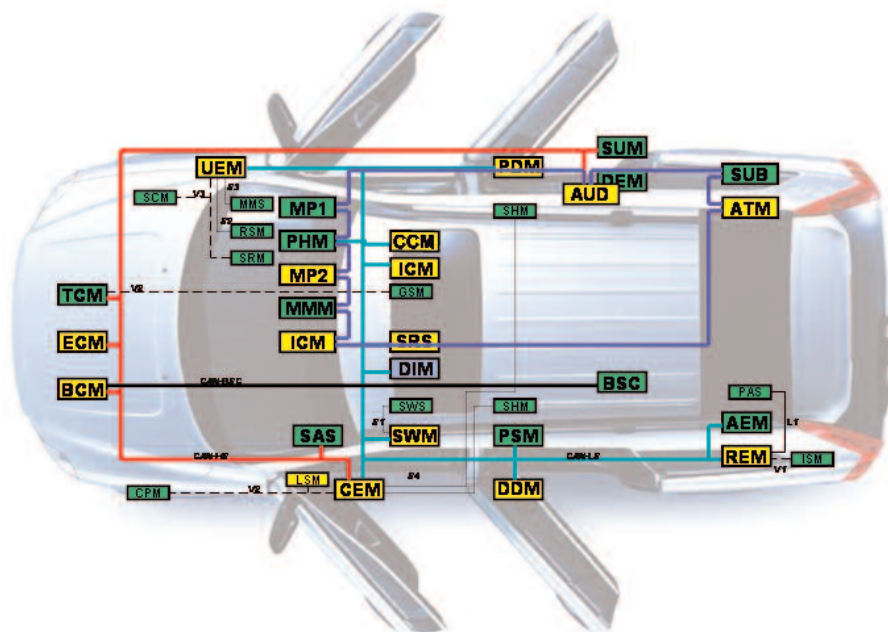


Fig.1. Network Infrastructure of Volvo XC90 (in "Real-Time in Embedded Systems", Hans Hansson, Mikael Nolin, Thomas Nolte, Embedded Systems Handbook, CRC Press 2005, with permission)

communication, radio frequency channels, infrared connections, etc. Based on the physical media employed by the networks, they can be in general divided in to three main groups, namely: wireline based networks using media such as twisted pair cables, fiber optic channels (in hazardous environments like chemical and petrochemical plants), and power lines (in building automation); wireless networks supporting radio frequency channels, and infrared connections; and hybrid networks composed of wireline and wireless networks.

Although the use of wireline based field area networks is dominant, the wireless technology offers a range of incentives in a number of application areas. In industrial automation, for instance, wireless device (sensor/actuator) networks can provide a support for mobile operation required in case of mobile robots, monitoring and control of equipment in hazardous and difficult to access environments, etc. In a wireless sensor/actuator network, stations may interact with each other on a peer-to-peer basis, and with a base station. The base station may have its transceiver attached to a cable of a (wireline) field area network, giving rise to a hybrid wireless- wireline system [3]. A separate category is the wireless sensor networks, envisaged to be largely used for monitoring purposes.

The variety of application domains impose different functional and non-functional requirements on to the operation of networked embedded systems. Most of them are required to operate in a reactive way; for instance, systems used for control purposes. With that comes the requirement for real-time operation, in which systems are required to respond within a predefined period of time, mandated by the dynamics of the process under control. A response, in general, may be periodic to control a specific physical quantity by regulating dedicated end-effector(s), or aperiodic arising from unscheduled events such as out-of-bounds state of a physical parameter or any other kind of abnormal conditions. Broadly speaking, systems which can tolerate a delay in response are called soft real-time systems; in contrast, hard real-time systems require deterministic response to avoid changes in the system dynamics which potentially may have negative impact on the process under control, and as a result may lead to economic losses or cause injury to human operators. Representative examples of systems imposing hard real-time requirement on their operation are fly-by-wire in aircraft control, and steer-by-wire in automotive applications, to mention some.

The need to guarantee a deterministic response mandates using appropriate scheduling schemes, which are frequently

implemented in application domain specific real-time operating systems or frequently custom designed "bare-bone" real-time executives.

The networked embedded systems used in safety critical applications such as fly-by-wire and steer-by-wire require a high level of dependability to ensure that a system failure does not lead to a state in which human life, property, or environment are endangered. The dependability issue is critical for technology deployment. One of the main bottlenecks in the development of safety-critical systems is the software development process.

As opposed to applications mandating hard real-time operation, such as the majority of industrial automation controls or safety critical automotive control applications, building automation control systems, for instance, seldom have a need for hard real-time communication; the timing requirements are much more relaxed. The building automation systems tend to have a hierarchical network structure and typically implement all 7 layers of the ISO/OSI reference model [4]. In case of field area networks employed in industrial automation, for instance, there is little need for the routing functionality and end-to-end control. As a consequence, typically, only the layers 1 (physical layer), 2 (data link layer, including implicitly the medium access control layer), and 7 (application layer, which covers also user layer) are used in those networks.

This diversity of requirements imposed by different application domains (soft/hard real-time, safety critical, network topology, etc.) necessitated different solutions, and using different protocols based on different operation principles. This has resulted in plethora of networks developed for different application domains.

With the growing trend for networking of embedded system and their interconnecting with LAN, WAN, and the Internet (for instance, there is a growing demand for remote access to process data at the factory floor), many of those systems may become exposed to potential security attacks, which may compromise their integrity and cause damage as a result. The limited resources of embedded nodes pose considerable challenge for the implementation of effective security policies which, in general, are resource demanding. These restrictions necessitate a deployment of lightweight security mechanisms. Vendor tailored versions of standard security protocol suites such as Secure Sockets Layer (SSL) and IP Security Protocol (IPSec) may still not be suitable due to excessive demand for resources. Potential security solutions for this kind of systems depend heavily on the specific device or system protected, application domain, and

extent of internetworking and its architecture.

Design Method for Networked Embedded Systems

Design methods for networked embedded systems fall into the general category of system-level design [5]. They include two separate aspects, which will be discussed briefly in the following. A first aspect is the network architecture design, in which communication protocols, interfaces, drivers and computation nodes are selected and assembled. A second aspect is the system-on-chip design, in which the best HW/SW partition is selected, and an existing platform is customized, or a new chip is created for the implementation of a computation or a communication node. Both aspects share several similarities, but so far have generally been solved using ad-hoc methodologies and tools, since the attempt to create a unified electronic system-level design methodology so far have failed.

When one considers the complete networked system, including several digital and analog parts, many more trade-offs can be played at the global level. However, it also means that the interaction between the digital portion of the design activity and the rest is much more complicated, especially in terms of tools, formats and standards with which one must inter-operate and interface.

In the case of network architecture design, tools such as OpNet and NS are used to identify communication bottlenecks, investigate the effect of parameters such as channel Bit Error Rate, and analyze the choice of coding, medium access and error correction mechanisms on the overall system performance. For wireless networks, tools such as Matlab and Simulink are also used, in order to analyze the impact of detailed channel models, thanks to their ability to model both digital and analogue components, as well as physical elements, at a high level of abstraction. In all cases, the analysis is essentially functional, i.e. it takes into account only in a very limited manner effects such as power consumption, computation time, and cost. This is the main limitation that will need to be addressed in the future, if one wants to be able to model and design in an optimal manner low power networked embedded systems, such as those that are envisioned for wireless sensor network applications.

At the system-on-chip architecture level, the first decision to be made is whether to use a platform instance or design an Application-Specific Integrated Circuit from scratch. The first option builds on the availability of large libraries of Intellectual Properties (IP), both in the form of processors, memories and peripherals, from

major silicon vendors. These IP libraries are guaranteed to work together, and hence constitute what is termed a platform. A platform is a set of components, together with usage rules that ensure their correct and seamless inter-operation. They are used to speed up time-to-market, by ensuring rapid implementation of complex architectures. Processors (and the software executing on them) provide flexibility to adapt to different applications and customizations (e.g. localization and adherence to regional standards), while hardware IPs provide efficient implementation of commonly used functions.

A platform thus is a single abstract model that hides the details of a set of different possible implementations as clusters of lower level components. The platform, e.g. a family of micro-processors, peripherals and bus protocols, allows developers of application designs to operate without detailed knowledge of the implementation (e.g., the pipelining of the processor or the internal implementation of the UART). At the same time, it allows platform implementors to share design and fabrication costs among a broad range of potential users, broader than if each design was a one-of-a-kind type.

Design methods that exploit the notion of platform generally start from a functional specification, which is then mapped onto an architecture (a platform instance) in order to derive performance information and explore the design space.

Full exploitation of the notion of platform results in better re-use, by decoupling independent aspects, that would otherwise tie, e.g. a given functional specification to low-level implementation details. The guiding principle of separation of concerns distinguishes between:

- Computation and communication. This separation is important because refinement of computation is generally done by hand, or by compilation and scheduling, while communication makes use of patterns.
- Application and platform implementation, because they are often defined and designed independently by different groups or companies.
- Behavior and performance, which should be kept separate because performance information can either represent non-functional requirements (e.g. maximum response time of an embedded controller), or the result of an implementation choice (e.g. the worst-case execution time of a task). Non-functional constraint verification can be performed traditionally, by simulation and prototyping, or with static formal checks, such as schedulability analysis. Tool support for System-On-Chip architectural design is so far mostly limited to simulation and interface generation. The first

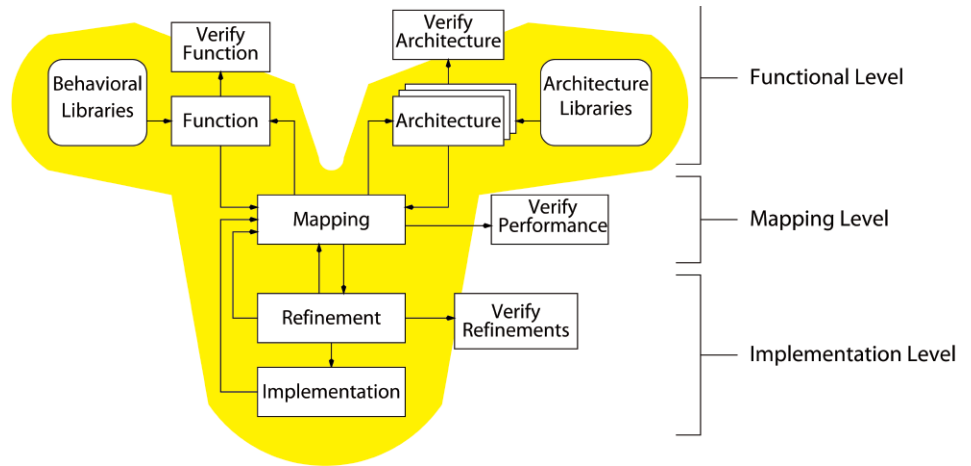


Figure 2. Design methodology for embedded system (in "Design of Embedded Systems", Luciano Lavagno and Claudio Passerone, Embedded Systems Handbook, CRC Press 2005, with permission)

category includes tools such as NC-SystemC from Cadence, ConvergenSC from CoWare, and SystemStudio from Synopsys. Simulators at the SOC level provide abstractions for the main architectural components (processors, memories, busses, HW blocks) and permit quick instantiation of complete platform instances from template skeletons. Interface synthesis can take various forms, from the automated instantiation of templates offered by N2C from CoWare, to the automated consistent file generation for SW and HW offered by Beach Solutions.

A key aspect of design problems in this space is compatibility with respect to specifications, at the interface level (bus and networking standards), instruction-set architecture level, and Application Procedural Interface level. Assertion-based verification techniques can be used to ease the problem of verifying compliance with a digital protocol standard (e.g. for a bus).

Let us consider in the following an example of a design flow in the automotive domain, which can be considered as a paradigm of any networked embedded system.

Automotive electronic design starts, usually 5-10 years before the actual introduction of a product, when a car manufacturer defines the specifications for its future line of vehicles.

It is now accepted practice to use the notion of platform also in this domain, so that the electronic portion (as well as the mechanical one, which is outside the scope of this discussion) is modularized and componentized, enabling sharing across different models. An Electronic Control Unit (ECU) generally includes a micro-controller (8, 16 and 32 bits), memory (SRAM, DRAM and Flash), some ASIC or FPGA for interfacing, one or more in-vehicle network interfaces (e.g. CAN or FlexRay), and several sensor and actuator interfaces (analog/digital and digital/analog converters, pulse-width modulators, power transistors,

display drivers and so on).

The system-level design activity is performed by a relatively small team of architects, who know the domain fairly well (mechanics, electronics and business), define the specifications for the electronic component suppliers, and interface with the teams that specify the mechanical portions (body and engine). These teams essentially use past experience to perform their job, and currently have serious problems forecasting the state of electronics 10 years in advance.

Control algorithms are defined in the next design phase, when the first engine models (generally described using Simulink, Matlab and StateFlow) become available, as a specification for both the electronic design and the engine design. An important aspect of the overall flow is that these models are not frozen until much later, and hence both algorithm design and (often) ECU software design must cope with their changes. Another characteristic is that they are parametric models, sometimes re-used across multiple engine generations and classes, whose exact parameter values will be determined only when prototypes or actual products will be available. Thus control algorithms must consider both allowable ranges and combinations of values for these parameters, and the capability to measure directly or indirectly their values from the behavior of engine and vehicle. Finally, algorithms are often distributed over a network of cooperating ECUs, thus deadlines and constraints generally span a number of electronic modules.

While control design progresses, ECU hardware design can start, because rough computational and memory requirement, as well as interfacing standards, sensors and actuators, are already known. At the end of both control design and hardware design, software implementation can start. As mentioned above, most of the software running on modern ECUs is automatically generat-

ed (Model-Based Design).

The electronic subsystem supplier in the HW implementation phase can use both off-the-shelf components (such as memories), ASSPs (such as micro-controllers and standard bus interfaces), and even ASICs and FPGAs (typically for sensor and actuator signal conditioning and conversion).

The final phase, called system integration, is generally performed by the car manufacturer again. It can be an extremely lengthy and costly phase, because it requires the use of expensive detailed models of the controlled system (e.g. the engine, modeled with DSP-based multiprocessors) or even of actual car prototypes. The goal of integration is to ensure smooth subsystem communication (e.g. checking that there are no duplicate module identifiers and that there is enough bandwidth in every in-vehicle bus). Simulation support in this domain is provided by companies such as Vast and Axys (now part of ARM), who sell both fast Instruction Set Simulators for the most commonly used processors in the networked embedded system domain, and network simulation models exploiting either proprietary simulation engines, e.g. in the case of Virtio, or standard simulators (HDL or systemC).

Concluding Remarks

This article has presented an overview of trends for networking of embedded systems, and their design. The networked embedded systems appear in a variety of application domains to mention automotive, train, aircraft, office building, and industrial automation. With the exception of building automation, the systems tend to be confined to a relatively small area covered and limited number of nodes, as in case of an industrial process, an automobile, or a truck. In the building automation controls, the networked embedded systems may take on truly large proportions in terms of area covered and number of nodes. For instance, in a LonTalk network, the total number of addressable nodes in a domain can reach 32385; up to 248 domains can be addressed.

The wireless sensor/actuator networks, as well as wireless - wireline hybrid networks, have started evolving from the concept to actual implementations, and are poised to have a major impact on industrial, home and building automation - at least in these application domains, for a start.

The networked embedded systems pose a multitude of challenges in their design, particularly for safety critical applications, deployment, and maintenance. The majority of the development environments and tools for specific networking technologies do not have firm foundations in computer science, and software engineering models and practices making the development

process labor-intensive, error-prone, and expensive.

References

- [1] The Industrial Communication Systems, Special Issue, Proceedings of the IEEE, ed. R. Zurawski, 93(6), June 2005.
- [2] The Industrial Communication Technology Handbook, CRC Press, FL, ed. R. Zurawski, 2005.
- [3] J.-D. Decotignie, P. Dallemagne, and A. El-Hoiydi, "Architectures for the interconnection of wireless and wireline fieldbusses," in Proc. 4th IFAC Conference on Fieldbus Systems and Their Applications 2001 (FET '2001), Nancy, France, 2001.

[4] Zimmermann H., "OSI Reference Model: The ISO model of architecture for open system interconnection", IEEE Transactions on Communications, 28(4): 425-432, 1980.

[5] Luciano Lavagno and Claudio Passerone, "Design of Embedded Systems", Embedded Systems Handbook, CRC Press, Boca Raton, Florida, Ed. R. Zurawski, 2005

Copyright Notice

This article is an excerpt from "Embedded Systems - Towards Networking of Embedded Systems", Luciano Lavagno and Richard Zurawski authors, in Embedded Systems Handbook, CRC Press, Boca Raton, Florida, Ed. R. Zurawski, 2005

from page 11:

Answers to 10 Questions by the two Candidates for 2007 IEEE President-Elect

John Vig: The president's duties are to: chair the meetings of the IEEE Board of Directors, Executive Committee and Assembly; perform ceremonial functions such as meeting with dignitaries, presentation of awards, opening remarks at conferences, etc.; promote the objectives of the IEEE; and be "the Chief Executive Officer of the IEEE."

I would make maximum use of the presidency to advocate the IEEE's agenda, both within and outside the IEEE.

I would set at least one lofty (man-on-the-moon-like) goal for the IEEE, aimed at inspiring and mobilizing the volunteers and staff.

The Board of Directors has been too inward-focused. I would propose the establishment of a council of advisors - consisting of prominent, mostly outside experts and leaders - to advise the IEEE leadership.

In the 2005 IEEE elections, only 14% of the membership voted. What, if anything, would you do to increase members' participation in IEEE elections?

Lew Terman: I think what we are doing this year is pretty good - talking to the Regions and other entities which invite us (with Q&A sessions where time permits), sending these 10 questions to the Newsletters, participating in the Philadelphia debate and making available recordings of the debate and presentations of the candidate platforms on the IEEE web site, and making additional information available on our personal web sites.

John Vig: In 1975-77, when a controversial candidate, Irwin Feerst, ran for IEEE president, 36% voted. In those days, the membership was more involved in IEEE issues than they are today. Today, the membership is rarely informed of controversial issues. For example, last year, I received reports of meetings where readings from the Koran and Christian prayers were parts of the program. Why not report such events and ask the membership whether or not such religious expressions should be allowed as parts of IEEE events? "THE INSTITUTE is the newspaper of the IEEE" claims The Institute's website but, The Institute is more a "house organ" than a newspaper. As president, I would propose to the Board of Directors, and The Institute's Editorial Board, that The Institute become a real newspaper of the IEEE. The office holders in IEEE, especially the President and the other members of the Board of Directors, make decisions about matters that are important to the membership and the future of IEEE. Voting in the annual IEEE election is

the chance members have to choose the decision makers. With only 14% voting, 7+% of the members can decide the fate of IEEE.

What have been your three most important contributions to IEEE?

Lew Terman: In the late 1990's, I was instrumental in the conversion of the Solid-State Circuits Council to the Solid-State Circuits Society. This was very successful; the SSCS is now the 5th largest Society in the IEEE, and the Journal of Solid-State Circuits records the highest number of hits in IEL. I served as the first SSCS president elected by the Society.

In the mid 90's, IEEE and TAB were going through financial difficulties. I was appointed TAB treasurer, stabilized the situation and improved the communication with TAB, and served a second term as Treasurer.

In 2001, I was on the Board as the bottom fell out of the IEEE financial situation. As part of a team effort, we were able to put in place a number of changes which arrested the slide.

John Vig: My three most important contributions are:

The IEEE Sensors Council, i.e., I proposed it, shepherded it through the approval processes, and was elected its founding president, in 1999. In 2005, the Council's journal published 1500 pages, and its conference had >500 registered participants. Between 1999 and 2002, the IEEE's reserves declined >\$50M (>40%), due, in large part, to the decline in the value of IEEE's investments. Up to this point, the IEEE had no formal investment policy. I wrote the first draft of the Investment Operations Manual (IOM), then worked with investment professionals, volunteers and staff to finalize it and get it passed by the Board. Contained in the IOM is an investment policy which has reduced the risks and increased the transparency of IEEE's investments. I brought what is now the IEEE Int'l Frequency Control Symposium into the IEEE. I negotiated the takeover of this conference by an IEEE society (UFFC). This conference is now the premier international conference in its field.

What would be your single and most recognized contribution that will distinguish your IEEE Presidency from those of others?

Lew Terman: I would like my presidency to result in the elimination of any silos between IEEE operating units, and attacking IEEE problems with coordinated efforts across IEEE.

John Vig: The president under whose leadership innovation flourished in IEEE.