On platforms for CPS - adaptive, predictable and efficient

[Invited Presentation - Extended Abstract]

Lothar Thiele, Felix Sutton, Romain Jacob, Roman Lim, Reto da Forno, Jan Beutel
Dept. Information Technology and Electrical Engineering
Swiss Federal Institute of Technology Zurich (ETH), Zurich, Switzerland
firstname.surname@tik.ee.ethz.ch

CCS Concepts

•**Networks** → **Cyber-physical networks;** •**Computer systems organization** → **Sensor networks;** *Real-time system architecture;*

Keywords

cyber-physical systems; system design; predictability

1. CONTEXT

If visions and forecasts of industry come true then we will be soon surrounded by billions of interconnected embedded devices. We will interact with them in a cyber-human symbiosis, they will not only observe us but also our environment, and they will be part of many visible and ubiquitous objects around us. The information that is collectively gathered and analyzed is supposed to help us in our daily live, in making faithful decisions, but it will also directly be used for actuation and it will cause changes by means of local and global control loops.

These systems can be regarded as massively distributed embedded systems with sensing, processing, communication and actuation capabilities. Variations or subclasses are known and have been thoroughly investigated under keywords such as "wireless sensor network (WSN)", "cyber-physical system (CPS)", and "internet of things (IoT)". Potential application domains are too numerous to be listed here, but some examples are in personal health and medicine (measuring environment, context and physiological data and providing information to various recipients), in environmental sensing (indoor and outdoor air pollution, environmental status of our environment), building automation and control, large-scale energy distribution including micro-grids, factory automation, logistics and surveillance.

2. REQUIREMENTS

In many of these applications we are faced with a set of conflicting requirements. We will concentrate on the following three that are most important for developing platforms for a large class of applications.

RSP'16, October 06-07, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2990299.2990308

2.1 Resource Constraints

Many of the above mentioned application domains of WSN, CPS or IoT systems impose fierce resource constraints on the underlying distributed platform.

Looking at the expected number of electronic devices around us, it becomes clear that frequently changing batteries is not an option. Even if the devices are directly connected to a sustainable power source, a high energy consumption is prohibitive from an environmental protection point of view. As a result, the average power should be below certain bounds in order to allow for long-term autonomous, maintenance-free, cost-effective and environment-friendly behavior. This requirement is independent of whether (rechargeable) batteries are used or whether a sustained operation is achieved using energy harvesting.

In many application scenarios we are confronted with additional constraints in terms of the available memory on the various devices, computing capabilities, communication bandwidth and responsiveness, device size and weight. Some of these constraints are due to cost, others due to direct or indirect application requirements such as limited available space, unobtrusiveness of the intended devices, available frequency bands and protocols for communication, or power constraints.

2.2 Adaptive Behavior

The class of distributed embedded systems we are discussing is in a constant interaction with its environment. In many scenarios, it is expected that the functionality adapts due to (a) changes in the environmental context or due to (b) requirement changes caused by user interactions. Typical examples are surveillance applications where as the result of a change in the sensed environment a message should be sent to some host application, the current situation will be analysed based on the available data and as a result, other sensor modalities may be switched on in the distributed system. Thus, the necessary communication bandwidth as well as the computation requirements will drastically change in all or some of the devices. A similar change in usage scenarios can be found in early warning scenarios related to environmental sensing, in building automation or in wearable medical/health devices.

But even if the application could be implemented without the concept of different usage scenarios, they provide us with an enormous potential for reducing the average power consumption. Putting communication, computation, sensors

and actuators in deep sleep mode or duty-cycling/frequency-scaling them with the right level of activity has the potential of reducing the average power without compromising the overall functionality. As will be mentioned later, such a mechanism imposes great challenges in the system design in terms of predictability and reliability.

2.3 Reliability and Predictability

Scaling up the number of devices demands autonomous operation. We have the legitimate expectation that the individual devices as well as the overall system behaves in a reliable and predictable manner. This is an indispensable requirement as it is infeasible to constantly maintain such a large set of devices. In addition, there are many application domains where we rely on a correct and fault-free system behavior. We expect trustworthy results from sensing, computation, communication and actuation due to economic importance or even catastrophic consequences if the overall system is not working correctly, e.g., in industrial automation, distributed control of energy systems, surveillance, medical applications, or early warning scenarios in the context of building safety or environmental catastrophes. CPS and IoT will be an integral part of our society and we will interact with the corresponding distributed devices daily. Trustworthiness and reliability are mandatory for the societal acceptance of this human-cyber interaction and cooperation.

Predictablity and reliability concern the observable behavior such as end-to-end functionality and timing, trustworthiness of sensor values, expected system life-time (batteries) or availability (energy harvesting), but they also concern all system components including wireless communication, processing, sensing, data-base/cloud operation, and actuation.

3. CHALLENGES

The above three requirements are in major conflict with each other. The following observation appears to be well-known to everyone designing this type of embedded systems: The closer a system operates at its resource constraints the less predictable and reliable it will behave, in general. The reason for such a dependency between resource constraints and reliability is quite obvious. The resource requirements in terms of power, memory, computing and communication change at run-time as applications typically have a time-varying dynamic functionality. To make things worse, the complexity of the implementation and the non-deterministic behavior of the environment makes it very difficult to formally provide correct and accurate upper bounds on the resource usage. Design processes very often do not support the use of formal methods that would allow to formally prove the correctness of the design. Massive over-provisioning is not an option either due to resource constraints. As a result, there may remain a substantial probability for non-deterministic erroneous system behavior which is further degraded by the potentially enormous number of devices.

But the problem not only concerns malfunctioning in the classical sense but also the predictability of behavior in terms of non-functional properties such as reaction time, real-time behavior or end-to-end timing constraints. One application or part of an application may use resources dynamically that, as a consequence, are not available to other applications. Even if this resource interference does not lead to a system failure, it may lead to a delayed execution of sensing, actuation, computation or communication tasks.

Besides this conflict between predictability and reliability on the one hand and resource constraints on the other hand, it appears that the adaptivity requirements makes the problem even more challenging. Adapting the system to changes in the environment at run-time requires efficient network-wide coordination which again leads to increase resource usage and interference between application tasks. The time to wake-up, turn-on components and/or consistently change the operation rate adversely impacts responsiveness and adaptivity.

In addition, adaptivity substantially increases the non-determinism in resource usage with negative effects on predictability and reliability as described above. One typical example is the use of duty-cycling, different clock domains or event-driven operation in order to increase the energy efficiency. When incorporating power management techniques in order to achieve low-energy-footprint, additional complexity is added to the system, further exacerbating the effects of resource interference.

4. APPROACH

We now present a prototyping and design process for cyber-physical systems that attempts to overcome the above difficulties. It is by no means the only possibility to solve the above listed challenges. Related approaches are cited and assessed in the referenced publications. The following sections introduce the main underlying architectural principles and a possible testing infrastructure. All the elements closely fit together and form a consistent design and prototyping process for cyber-physical systems.

4.1 FlockLab and Testing Environment

An essential part of the proposed prototyping and design process is the availability of a suitable testing environment. Unfortunately, distributed embedded systems are usually hardly observable and controllable due to their resource constraints in terms of computing power and communication bandwidth. On the other hand, simulation appears not to be sufficient due to the missing system context and the high degree of uncertainty in the system environment, e.g., quality of wireless links.

FlockLab [2] is a distributed testing and measurement infrastructure that is used by many research groups worldwide to prototype new devices, nodes, communication protocols as well as complete applications. It allows for the prototyping of distributed embedded systems in terms of various functional and non-functional properties and enables realistic and large-scale experiments. It combines the capability of a logic analyzer, a serial data logger, a dynamic power analyzer and a programmable power supply with deep local storage and network synchronization [4]. It supports multiple targets, and therefore allows the comparative and explorative analysis of applications and protocols on a single testbed.

Each FlockLab observer node is built using Linux running on an embedded processor, an FPGA, and a SoC with an RF core for high precision time synchronization. Event tracing is possible with a time resolution of $0.5\mu s$ [3], a peak event rate of 10^8 events/s, an average event rate of $3 \cdot 10^5$ events/s, and concurrently recorded traces are aligned within $1\mu s$ with an empirical probability of 99.9%.

4.2 Bolt and Interference Avoidance

As described above, one major difficulty is the interference of hardware and software components in terms of time, power and clock domains. A multitude of tasks such as reading sensors, processing data, communicating over radio must be executed concurrently and may lead to resource interference. Tasks must compete for shared resources such as clock cycles, memory and peripherals.

To overcome these difficulties, we propose a multi-processor architecture whereby the tasks interact through inter-processor communication using asynchronous message passing [5]. This concepts allows to decouple system components with respect to time, power and clock domains. It enforces the principle of composability of components in order to give the flexibility to select processors satisfying the needs of the application. It ensures that the interconnection of the processors does not change the properties of the integrated parts. In stark contrast to classical approaches such as shared memory or bus-based communication which violate timing composability and the possibility of independent design.

The main ingredient of the architecture is the "Bolt" interconnect which implements in hardware and software an efficient communication via message queues that allows for a combination of event- and time-triggered operation. It provides a well-defined interface semantics and formally proven timing properties using modeling and formal verification based on timed automata. The architectural concept has been extensively tested using FlockLab to validate the correctness of the timing bounds as well as to confirm that the power overhead of the stateful interconnect is negligible.

This way, we reduce the interference in terms of power domain, time and memory accesses to support composability, and we leverage the very recent trend towards ultra low-power multi-processor architectures that can be chosen to match the needs of the application and the networking protocol in an efficient manner.

4.3 Event-Based Design and Adaptability

Adaptability of the overall system is enabled by means of an event-driven system paradigm. In particular, the whole system architecture including hardware and software is partitioned into logical components that are event-driven. Examples are event detection, event characterization and filtering, and wireless multi-hop communication.

These components are characterized by a novel event-triggered abstract model. Each component adheres to a novel event-triggered interface specification that consists of input and output event streams as well as data streams. The functional behavior is specified by a non-deterministic finite state machine that is used to (a) determine the timing behavior and power consumption and to (b) determine the statistical timing properties of the emitted event streams. The design and implementation of each logical component follows the guideline "sleep whenever possible, wake-up quickly, and operate efficiently in every mode": Each component must be able to rapidly switch between the various operation modes and perform its designated tasks efficiently. The above mentioned component model and the associated interface allows to estimate the average power consumption as well as the end-to-end responsiveness of the system in a quantitative manner.

4.4 End-to-end Guarantee and Predictability

If put together, the above architectural design principles can be used to design systems with a guaranteed end-to-end timing behavior, see [1]. In particular, it can be guaranteed that messages that are received by the target application interface do so before provided end-to-end deadlines. The principles of "Bolt" and the interference avoidance allows to freely compose existing hardware and software components to satisfy the application requirements, without altering the properties of the integrated parts. To satisfy adaptivity requirements, the architecture can adapt to dynamic changes in the system and real-time traffic requirements.

On top of "Bolt", a novel distributed real-time protocol has been designed which provably guarantees that message buffers do not overflow and that all messages received by the target application interfaces meet their end-to-end deadlines. The protocol dynamically establishes at run-time a set of contracts depending on the current real-time traffic demands in the system. Contracts define the mutual obligations between distributed devices and the networking protocol in terms of minimum service provided and maximum demand generated. This enables end-to-end guarantees without impairing the decoupling of communication from application tasks.

Some of the above requirements, challenges and design principles will be exemplified by design experiences in implementing an ultra-low power distributed embedded systems for the detection of acoustic events, including the necessary sensor interface, an event-triggered characterization pipeline, and an event-based low-power multi-hop wireless protocol.

Acknowledgements

The project XSENSE has been supported by the Nanotera.ch, a research program from the Swiss Confederation.

5. REFERENCES

[1] R. Jacob, M. Zimmerling, P. Huang, J. Beutel, and L. Thiele. End-to-end real-time guarantees in wireless cyber-physical systems. In *37th IEEE Real-Time Systems Symposium (RTSS)*, November 2016.

[2] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 153–165, 2013.

[3] R. Lim, B. Maag, B. Dissler, J. Beutel, and L. Thiele. A testbed for fine-grained tracing of time sensitive behavior in wireless sensor networks. In *IEEE 40th Local Computer Networks Conference Workshop (LCN)*, pages 619–626, 2015.

[4] R. Lim, B. Maag, and L. Thiele. Time-of-flight aware time synchronization for wireless embedded systems. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 149–158, 2016.

[5] F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele. Bolt: A stateful processor interconnect. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (ENSS)*, pages 267–280, 2015.