Embedded Virtual Machines for Wireless Industrial Automation

(Demo and Poster)

Miroslav Pajic and Rahul Mangharam
Dept. of Electrical & Systems Engineering
University of Pennsylvania, U.S.A.
{pajic, rahulm}@seas.upenn.edu

Abstract

The factory of the future is the Wireless Factory - fully programmable, nimble and adaptive to planned mode changes and unplanned faults. Today automotive assembly lines loose over $22,000 per minute of downtime. The systems are rigid, difficult to maintain, operate and diagnose. Our goal is to demonstrate the initial architecture and protocols for all-wireless factory control automation. Embedded wireless networks have largely focused on open-loop sensing and monitoring. To address actuation in closed-loop wireless control systems there is a strong need to re-think the communication architectures and protocols for reliability, coordination and control. As the links, nodes and topology of wireless systems are inherently unreliable, such time-critical and safety-critical applications require programming abstractions where the tasks are assigned to the sensors, actuators and controllers as a single component rather than statically mapping a set of tasks to a specific physical node at design time. To this end, we introduce the Embedded Virtual Machine (EVM), a powerful and flexible runtime system where virtual components and their properties are maintained across node boundaries. EVM-based algorithms introduce new capabilities such as provably minimal graceful degradation during sensor/actuator failure, adaptation to mode changes and runtime optimization of resource consumption. Through the design of a micro-factory we aim to demonstrate the capabilities of EVM-based wireless networks.

Keywords: Real-time systems, embedded systems, wireless sensor networks, virtual machines.

1. Introduction

Embedded Wireless Sensor-Actuator-Controller (WSAC) networks are emerging as a practical means to monitor and operate automation systems with lower setup/maintenance costs. While the physical benefits of wireless, in terms of cable replacement, are apparent, automation manufacturers and plant owners have increasing interest in the logical benefits.

With multi-hop WSAC networks, it is possible to build modular systems which can be swapped out for off-line maintenance during faults. Modular systems can be dynamically assigned to be primary or backup on the basis of available resources or availability of the desired calibration. Modularity allows for incremental expansion of the plant and is a major consideration in emerging economies. WSAC networks allow for runtime configuration where resources can be re-appropriated on-demand, for example when throughput targets change due to lower price electricity during off-peak hours or due to seasonal changes in end-to-end demand.

While WSAC networks facilitate both planned and unplanned mode changes, runtime programmable WSAC networks allow for flexible item-by-item process customization. For example, a high demand for fuel-efficient Toyota Prius' will require major retooling of a traditional wired factory that is designed for the Toyota Camry chassis. With re-programmable WSAC, the assembly line stations can adapt to a schedule where every 3 Camrys are interleaved with 2 Prius' with synchronized changes in operation modes and assembly line operations.

To this end, we introduce the Embedded Virtual Machine (EVM), a powerful and flexible runtime system where virtual components and their properties are maintained across node boundaries. EVMs differ from classical virtual machines (VM). In the enterprise or on PCs, one (powerful) physical machine may be partitioned to host multiple virtual machines for higher resource utilization. On the other hand, in the embedded domain, an EVM is composed across multiple physical nodes with a goal to maintain correct and high-fidelity operation even under changes in the physical composition of the network. The goal of the EVM is to maintain a set of *functional invariants*, such as a control law and *para-functional invariants* such as timeliness constraints, fault tolerance and safety standards across a set

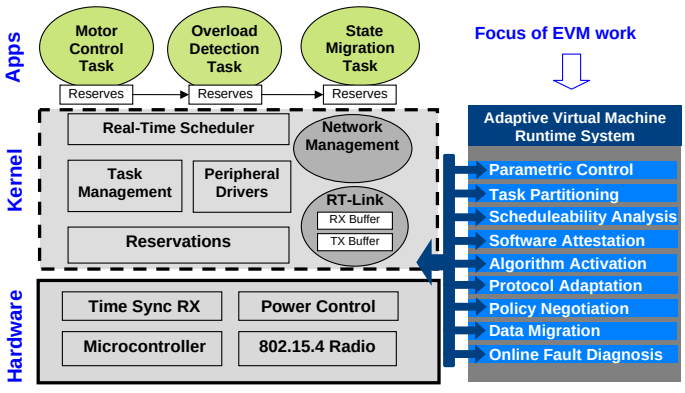

Figure 1. nano-RK sensor RTOS with interfaces to the EVM. EVM includes parametric and programmable control algorithms for runtime logical-task to physical-node mapping.

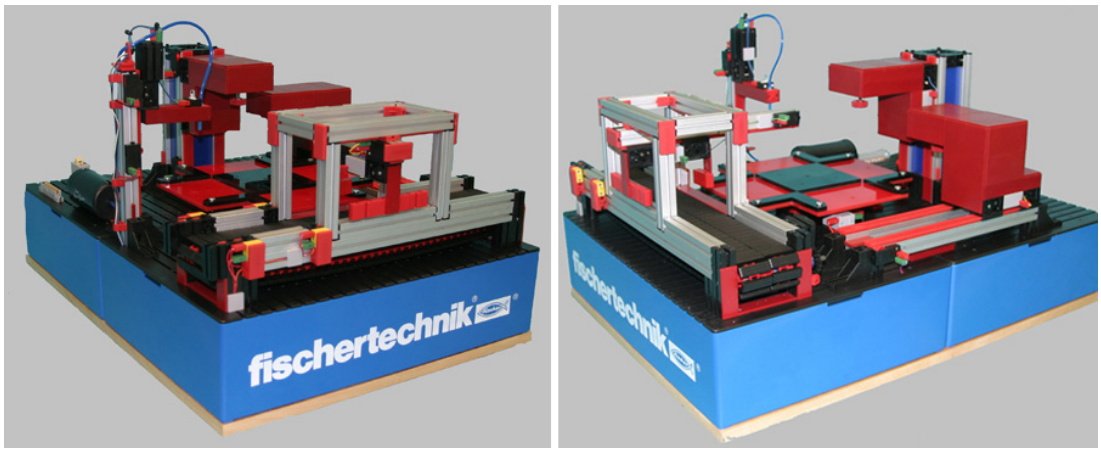

Figure 2. FischerTechnik work-cell module with a conveyor belt and 21 sensors and actuators

of controllers given the spatio-temporal changes in the physical network. By incorporating EVMs in existing and future wireless automation systems, our aim is to realize:
1. Predictable outcomes in the presence of controller failure
2. Provably minimal QoS degradation without violating safety
3. Composable and reconfigurable runtime system
4.Adaptive Resource Re-appropriation and Optimization

2. EVM Architecture and Algorithms

The system under consideration includes a number of wireless sensors, actuators and controllers composed into a Virtual Component. The Virtual Component acts as a single entity for the control algorithm execution. The EVM provides a flexible runtime system to share state and responsibilities across physical nodes and allows multiple EVM-enabled nodes to be composed into a single logical entity.

The EVM architecture and algorithms are built on a modified version of the FireFly sensor network platform [1] and nano-RK sensor real-time operating system (RTOS) [2]. The EVM is implemented in the form of a virtual machine abstraction layer on top of the RTOS and executes as a special task within nano-RK. As a special task, the EVM has both parametric and programmable control of the entire operating system and hardware resources.

We now consider the design of the EVM within the nano-RK RTOS framework. The EVM describes its own instruction set for efficient control, task and fault management between nodes. As with Mate [3], the EVM is based on a FORTH-like interpreter. The interpreter runs within nano-RK as a super task. However, unlike Mate, the EVMs instruction set is extensible at runtime. Furthermore, EVM instructions are focused on node-to-node communication and control rather than PC-to-node control. The EVM architecture has two main components - EVM node-specific operations and object transfers for efficient node-to-node communication. Both will be explained in the poster and demonstrated in the micro-factory automation test-bed described below.

3. EVM Evaluation

Our focus is on the fault-tolerance of controllers only. When a particular backup controller detects a series of faults in the primary controller, it triggers a task migration operation to the backup controller. This operation includes a capabilities check and the migration of the task control block, stack, data and timing/precedence-related metadata. The backup controller is activated and the primary controller switches to a passive 'indicator' mode.

We have implemented the parametric control capability of the EVM on the FireFly nodes over the nano-RK sensor RTOS. This allows remote runtime triggering of individual sensor drivers, modification of task reservations and network time-slot assignment. Through a discrete control case study, we evaluate the programmable control, more specifically the fault tolerant capability, of the EVM.

We will demonstrate the functioning EVM in a factory simulation module using the FischerTechnik model factory, as shown in Fig. 2. Such factory is used by companies such as BMW prior to building a real factory. Each module in the model factory consists of 22 sensors and actuators that are to be controlled in a coordinated and timely manner. All modules use use wireless control with FireFly nodes controlling all sensors and actuators. This test-bed will allow us to evaluate the EVM's communication, coordination and adaptation capabilities in a more realistic setting.

In summary, the specific objectives of this effort are:

1. Ability to deploy control algorithms in a virtual component defined over a grid of wireless controllers.

2. On-line capacity expansion where more controllers can be added to share the load and trigger re-distribution of tasks.

3. Algorithm replication to a set of nodes capable of performing the same control function for throughput adaptation.

4. Fault tolerance to node and communication failures.

5. Control algorithm execution with high-speed operation (1/4 second or less control cycle) and with a small latency ($\leq$1/3 of the control cycle).

References

[1] R. Mangharam, A. Rowe, and R. Rajkumar. FireFly: A Cross-layer Platform for Real-time Embedded Wireless Networks. *Real-Time System Journal*, 2007.

[2] nano-RK Sensor RTOS. http://nanork.org.

[3] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks . *ACM ASPLOS-X* , 2002.