Poster Abstract : A Cyber-physical Middleware Framework for Continuous Monitoring of Water Distribution Systems

Mudasser Iqbal, Hock Beng Lim
Intelligent Systems Center, Nanyang Technological University, Singapore
mmiqbal@ntu.edu.sg, limhb@ntu.edu.sg

Abstract

The middleware for a cyber-physical system is crucial as it tightly integrates computation with physical processes to achieve better reliability, distributed coordination, higher precision and efficiency, and better autonomous control. In this work, we design and develop the cyber-physical middleware framework for a large-scale water distribution system monitoring effort.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and Embedded systems*

General Terms

Design, Experimentation, Measurement

Keywords

Cyber-physical Systems, Middleware, Modeling, Scheduling

1 Introduction

The design principles of the cyber-physical system (CPS) middleware are guided by past research and lessons from the development of distributed, real time and embedded (DRE) systems. These include: (a) reconfigurable component-based design, (b) functional specialization for operation under constraints to avoid overheads, (c) portable and retrofittable components, (d) support for distributed, concurrent and synchronized execution of multiple components, (e) end-to-end Quality of Service (QoS) under both global and local constraints, (f) periodic and aperiodic events processing, (g) real-time adaptive scheduling and (h) composable complex workflows.

Existing efforts in middleware design for CPS [1, 2] provide techniques to meet some but not all of the aforementioned design principles. In this work, we design a comprehensive and rigorous middleware framework with the goal to develop an event-driven, specialized and reconfigurable middleware for a strictly time and resource constrained CPS. In particular, this middleware design is driven by the "Continuous Monitoring of Water Distribution Systems" project, which is a large-scale water distribution system (WDS) monitoring project in Singapore (please refer to the Acknowledgements).

Our system will have certain intrinsic features such as measurements of hydraulic and water quality parameters using a large network of sensor nodes requiring real time and non-sporadic middleware operation. This data will be integrated and assimilated into hydraulic models for predicting/updating the hydraulic state of the system, which will in turn be fed into the physical network to optimize pumping operations for water conservation and efficient power management. Our system will also include extrinsic features such as the testing and evaluation of event detection scenarios (e.g. for leaks, pipe bursts and water contamination/security events that involve multiple geographically distributed sensor nodes) which may require distributed, concurrent and specialized operation of extrinsic middleware components under stringent time and resource (e.g. sensor node CPU, memory and power) constraints as defined by the events. The system will also evolve over time and this will add to the uncertainty and the complexity of the physical system being monitored, which mandates the middleware's ability to construct complex workflows and their prioritized scheduling for in-network or centralized execution.

2 Middleware Design

Figure 1 shows the middleware design as a component-based layered framework based upon Feature Oriented Software Design (FOSD) [3]. This layered approach categorizes the components as either intrinsic or extrinsic, so that the middleware can avoid overheads due to unnecessary components in the execution flow.

The components in the middleware comprise of entities

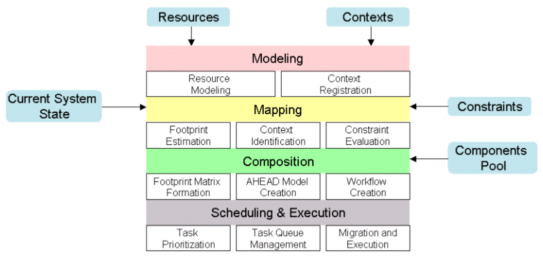

Figure 1. Layered middleware design for CPS.

SenSys'09, November 4–6, 2009, Berkeley, CA, USA.
ACM 978-1-60558-748-6

called agents and actors which may move across the network of sensor nodes in the field and the computational backbone to allow adaptive load balancing. An agent is a system component (intrinsic or extrinsic) that specializes in monitoring periodic/aperiodic events triggered by the physical system or another agent. Upon detection of an event, workflows are created and executed that may be as simple as activating an actor to carry out some tasks or as complicated as activating a sequence of agents and actors with stringent resource, time and context bounds. This is achieved by using the various middleware layers shown in Figure 1.

2.1 Modeling

Each event and the corresponding actions uses some resources and occurs in a particular context. This information is critical to decide which agents/actors must comprise the action workflow such that the required task is done in the correct context and under given constraints. For example, in the case of leak detection, the agents are used to process both pressure and acoustic data in real time to find the patterns that characterize a pipe leak. Upon detection of such a pattern, a workflow will be created to localize the leak. For each agent and actor, the middleware uses the initial system state to define models represented as:

$$M_a = \{R_a, C_a\} \tag{1}$$

where R and C are the descriptors of the resources an entity a uses and the possible contexts under which it can exist, respectively.

2.2 Mapping

Once an event occurs at time t, the entity models as defined by (1) will need to be qualified for t; i.e. given the basic entity model, what will be the values of R for entity a for the same values of C and the constraints Q under which it must operate. The mapping layer carries out this task by taking basic entity model and current state of system resources as input. The output of this layer is an entity map describing in detail the foot print that the entity will make on system resources, if executed. For instance, the execution latency and memory consumption map for entity a at time t looks like:

$$E_{a(t)} = LProcessor_{a(t)} + LNetwork_{a(t)} + LSensor_{a(t)} \tag{2}$$

$$R_{a(t)} = FRAM_{a(t)} + FROM_{a(t)} + FPermanent_{a(t)} \tag{3}$$

where E represents the execution latency of an entity a at time t due to processor, network and sensor hardware latencies. Likewise, R describes the memory footprint of entity a in terms of its RAM, ROM and permanent storage utilization. It is important to note that the inclusion of current system state ensures that those entities which may be able to ensure performance delivery under certain bounds in certain system conditions, but cannot deliver the same under other system conditions, must be regarded as overheads. These maps are useful diagnostic tools for evaluating the middleware's efficiency. They can also be used for understanding the performance of the sensor nodes, which may lead to the need to refine the sensor node hardware.

2.3 Composition

Once the maps for the entities in the system have been generated, workflows are composed using the principles of Algebraic Hierarchical Equations for System Design (AHEAD) [3], which is an implementation of FOSD. We use AHEAD to construct equations that represent ranking of each entity in terms of its ability to participate in the workflow being composed. AHEAD provides a tool called Origami Matrix to derive feature compositions. In our design, the rows of the matrix indicate the features sought in an entity for a task in the workflow and the columns indicate all possible features provided by an entity. The values in the cells of the matrix are the resource footprint values. When the matrix is folded along rows and columns, an algebraic equation is formed which will be the sum of all the feature footprints for the entity involved. Doing so for each entity and sorting them gives an entity that can do the task with the least footprint and possibly within stringent constraints.

2.4 Scheduling and Execution

The entities are then scheduled for execution as thread pools. The position of each entity in the thread pool is determined by its priority. For this purpose, we use End-to-End Deadline Monotonic Scheduling (EMDS) [1], so that a task with a shorter deadline is given a higher priority. When each new entity arrives in the thread pool for execution, the priorities for all of the entities in the thread pool are re-evaluated to ensure that the most time-critical tasks are always executed ahead of the others.

3 Conclusions and Future Work

We have proposed a middleware framework for cyber-physical systems based on rigorous techniques from the design of DRE systems. From the current prototype of eight sensor nodes deployed in downtown Singapore, our project will be scaled up to over hundred sensor nodes in the future. The proposed middleware framework will handle real-time data management, in-network collaborative event detection and hydraulic modeling.

4 Acknowledgments

This work is supported by the Singapore National Research Foundation (NRF) through a subcontract between the Singapore-MIT Alliance for Research and Technology (SMART) Center for Environmental Sensing and Modeling (CENSAM - http://censam.mit.edu) and the Intelligent Systems Center (IntelliSys) of Nanyang Technological University. We would like to acknowledge our colleagues (Mike Allen, Cheng Fu, Lewis Girod, Amitsur Preis, Seshan Srirangarajan, Andrew Whittle, Kai-Juan Wong) for their contributions to this project.

5 References

[1] Y. Zhang, C. Gill, and C. Lu. “Reconfigurable real-time middleware for distributed cyber-physical systems with aperiodic events,” In *Proc. of ICDCS*, pp. 581–588, 2008.

[2] A. Dabholkar and A. Gokhale. “An approach to middleware specialization for cyber physical systems,” In *Proc. of WCPS*, 2009.

[3] D. Batory, J. N. Sarvela, and A. Rauschmayer. ”Scaling stepwise refinement,” In *IEEE Trans. on Soft. Eng*, Vol. 30(6), pp. 355–371, 2004.