

# Refinement of UML/MARTE Models for the Design of Networked Embedded Systems

E. Ebeid, F. Fummi, D. Quaglia, F. Stefanni

Dep. of Computer Science — University of Verona, Italy

[emadsamuelmalki.ebeid|franco.fummi|davide.quaglia|francesco.stefanni]@univr.it

**Abstract**—Network design in distributed embedded applications is a novel challenging task which requires 1) the extraction of communication requirements from application specification and 2) the choice of channels and protocols connecting physical nodes. These issues are faced in the paper by adopting UML/MARTE as specification front-end and repository of refined versions of the model obtained by both simulation and analytical exploration of the design space. The emphasis is on using standard UML/MARTE elements for the description of networked embedded systems to allow re-use, tool interoperability and documentation generation. The approach is explained on a case study related to building automation.

## I. INTRODUCTION

Distributed embedded applications are becoming more and more complex, involving many different tasks spread over hundreds or thousands of heterogeneous network nodes connected through different types of channels and protocols [1]. For instance, Figure 1 shows a portion of a temperature control application in which tasks and corresponding data flows are distributed over five rooms of a floor-plan. A module of the application (in Figure 1 denoted by  $t1$ ) performs an initialization phase; then, for each room, a sensing task (denoted by  $t5$ ) sends data to a centralized controller (task  $t2$ ), which sends commands to actuators in each room (task  $t4$ ) according to a global heating policy. A controlling task (denoted by  $t3$ ) is also embedded into mobile devices to let people set the preferred temperature of the environment around them.

In this context, computer-aided design should be fruitfully applied not only to each node, as currently done, but also to the communication infrastructure among them. The design flow goes through a sequence of *refinement steps*; after each of them new details are added to the model of the application until it is ready for the actual implementation. A possible list of details to be progressively added for the design of the previous application is:

- computation and communication requirements of the application tasks;
- the position of sensors, controllers and actuator as well as the network topology;
- allocation of tasks over networked embedded systems with suitable computation and communication capabilities;

Research activity partially supported by the European project FP7-ICT-2011-7-288166 TOUCHMORE.

978-3-9810801-8-6/DATE12/©2012 EDAA

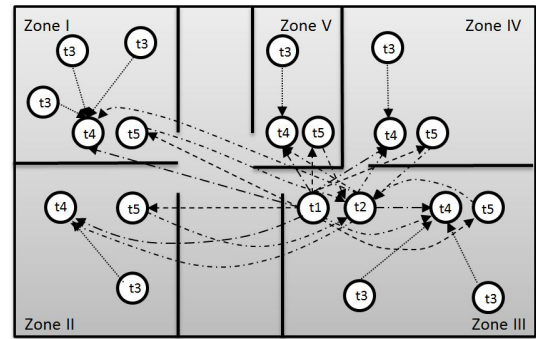


Fig. 1. Data flows and topology of a distributed embedded application.

- description of physical channels among nodes with the choice of the medium (i.e., wired or wireless);
- description of communication protocols (e.g., reliable vs. un-reliable, best effort vs. priority-based).

In literature, such refinement steps have been addressed especially in the context of wireless sensor networks [2], [3] and network-on-chips [4]. In particular, the so-called *network synthesis* problem has been stated as follows [5], [6]; given 1) a set of end-to-end delay, throughput and packet error rate constraints, 2) the environment geometry, and 3) a library of actual components together with their performance and cost characterization, a synthesis algorithm produces a network implementation that satisfies all end-to-end constraints and that is optimal with respect to some metrics (e.g., cost or power consumption). For instance, the COmmunication Synthesis Infrastructure (COSI) [5] describes and solves a mixed integer linear programming problem in the specific context of building automation without considering the design of each node. The Communication-Aware Specification and Synthesis Environment (CASSE) [6] is a more general approach which describes the problem in term of tasks, data flows among them, nodes, abstract channels and environmental constraints. Abstract channels are a generalization of physical links taking into account also protocols. CASSE solution consists in mapping tasks onto nodes, data flows onto abstract channels and positioning nodes in the environment.

Both approaches do not rely on a *standard representation* of requirements (from the initial user specification) and solutions; this standard format could be exploited by state-of-the-art tools for embedded system design and as well as to generate

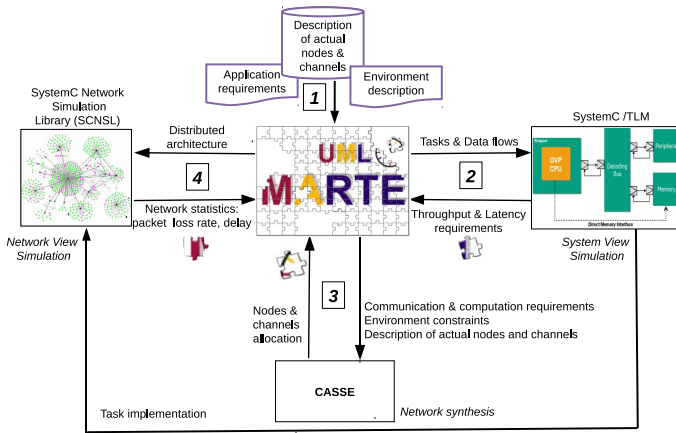


Fig. 2. Proposed methodology.

documentation.

Model-driven design and, in particular, UML profiles such as SysML and MARTE have been proposed to introduce all the information required in the first steps of the design process. The use of standard frameworks enable tool interoperability and the generation of widely-understandable documentation. The latest version of MARTE introduces the Generic Quantitative Analysis Modeling (GQAM) [7] sub-profile which provides some interesting stereotypes for the representation of computation and communication aspects of networked embedded systems:

- GaExecHost denotes a processor that executes steps;
- GaCommHost denotes a *physical* communication link;
- GaCommChannel denotes a *logical* communication layer connecting SchedulableResources.

Recent work proposes a methodology to generate SystemC executable models from UML [8], SysML [9] and MARTE [10]–[12] thus showing the need to generate executable models for estimation and validation; however they restrict the role of UML and MARTE at the first step of the design flow and it does not address the modeling of the network.

On the contrary, the contribution of this work is a design methodology for networked embedded systems which combines UML/MARTE, network synthesis, and simulation; in particular the methodology:

- 1) puts UML/MARTE models at the center of the design flow so that they are not only used at the beginning of the design flow, but they are refined progressively through simulation and analytical design space exploration;
- 2) uses the new GQAM sub-profile to represent computation and communication aspects of the system;
- 3) allows to model the result of network synthesis in UML/MARTE so that not only each single node but the whole distributed system is represented; this point is achieved by using the GQAM sub-profile;
- 4) relies on SystemC executable models not only for functional simulation but also for network simulation;

5) provides, as side-effect, SystemC executable models available for further design steps.

The rest of the paper is organized as follows. Section II presents the proposed methodology. Section III describes how UML/MARTE elements are used to describe requirements, constraints, nodes and channels. Section IV describes how communication requirements are extracted through the generation of a functional SystemC executable model. Section V describes how UML/MARTE is used to model the solutions of network synthesis, i.e., the whole distributed embedded application. Section VI describes how network solutions are validated and further refined through the generation of a network-aware SystemC executable model. Experimental validation is done through a case study on building temperature control which is used along the paper to clarify the proposed concepts. A summary of experimental results and conclusions are drawn in Section VII.

## II. PROPOSED METHODOLOGY

The proposed methodology is shown in Figure 2. UML/MARTE models are at the center of the methodology as a standard and interoperable representation of the system to be designed. Models are progressively built and refined (i.e., new details are added) by using the three surrounding tools for simulation and network synthesis; arrows are used to represent the flow of information between UML/MARTE models and such tools; numbers are used to show the ordering sequence of the design flow.

The input of the flow consists of application requirements, environment constraints and a description of actual nodes and channels. All this information is modeled in UML/MARTE by assigning a semantics to some elements and by separating computation from communication. A deep investigation has been performed to model design constraints. Section III details this step.

A crucial aspect is the functional validation of the model and the estimation of throughput and latency requirements for each data flow. For this purpose, the second step of the flow is devoted to the generation of an executable SystemC model from UML/MARTE information about tasks and data flows. Let us denote this simulation view as *System View* since the emphasis is on functionality and information exchange without deciding if communications are inside a single chip or implemented by an external network (e.g., Ethernet or WiFi). TLM primitives are used to represent communication and the previous literature on SystemC generation can be exploited [9], [10], [12]. By executing the SystemC code, the implementation of tasks can be refined and the communication requirements of the data flows are assessed and annotated back in the UML/MARTE diagrams. The evaluation of computational requirements of the tasks is not described in this work but it could be provided as proposed in previous literature [13]. Section IV details this step.

The third step is devoted to the *Network Synthesis*. The complete distributed system is created by mapping tasks onto actual nodes, data flows onto channels, and by placing nodes

in the environment. All the information about user constraints, communication requirements and actual channels and nodes are extracted from the UML/MARTE model and the resulting architecture is represented as a refined UML/MARTE model. As far as we know this is the first time that UML/MARTE is used to describe the whole distributed system together with the packet-based network. Section V details this step.

The network synthesis step finds solutions by using an analytical model of the channel which may not capture complex inter-node interactions; therefore the optimal solution or the set of candidate solutions must be validated through simulation; furthermore, it could be interesting to test the analytical solution in a larger scenario which can be handled more efficiently through simulation. For these purposes, an executable network scenario is generated from the MARTE description of the distributed system; task implementation is taken from Step 2 while nodes and channels are modeled by using the SystemC Network Simulation Library (SCNSL) [14] which reproduces the behavior of packet-based networks over SystemC and generates the corresponding statistics (e.g., packet loss rate and delay). Let us denote this simulation view as *Network View* since system functionality is simulated in a full network scenario. Information from Network View simulation are used to further refine the UML/MARTE model; at the end of this flow, the resulting MARTE model can be used to generate documentation and to support further refinement steps as traditionally done in embedded system design. Section VI details this step.

Among different tools for UML modeling, Papyrus [15] has been used in this work for its support of the latest MARTE sub-profiles.

### III. MODELING REQUIREMENTS, NODES AND CHANNELS IN UML/MARTE

The main aspects to be represented in UML/MARTE are tasks, nodes, channels and the external environment. For this purpose, a UML/MARTE class diagram has been created (Figure 3). Data types from MARTE Non-Functional Properties (NFP) are used to represent attributes such as throughput, price, and power consumption. Application requirements, both functional and non-functional, and environment constraints are represented through relationships between classes. Three stereotypes from the GQAM sub-profile are used in this diagram to specify the semantics of some classes and their attributes.

1) *Task*: A task represents a basic functionality of the whole application; it takes some data as input and provides some output. From the point of view of network synthesis the focus is on its computation and mobility requirements which affect the assignment of tasks to network nodes. A task  $t$  is defined as follows:

$$t = [c, m] \in \mathcal{T} \quad \text{where : } c \in \mathbb{R}^n \quad m \in \mathbb{B} \quad (1)$$

For this reason the *Task* class is stereotyped as *GaExecHost*. The  $c$  attribute represents computation requirements, i.e., CPU and memory utilization. For this

attribute the new data type named *ComputationAttr* has been created and stereotyped as *GaExecHost* so that its fields *mem\_size* and *CPU* are linked to *memSize* and *utilization*, respectively. The attribute named  $m$  is a boolean attribute representing the possible mobility requirement of the task.

2) *DataFlow*: A data flow represents communication between two tasks; output from the source task is delivered as input to the destination task. For network synthesis, the focus is on the communication requirements between tasks which affect the choice of channels and protocols between nodes hosting the involved tasks. A data flow  $f$  is defined as follows:

$$f = [ts, td, c] \in \mathcal{F} \quad \text{where : } ts \in \mathcal{T} \quad td \in \mathcal{T} \\ c = [throughput, maxdelay, maxerrorrate] \in \mathbb{R}^3 \quad (2)$$

For this reason the *DataFlow* class is stereotyped as *GaCommChannel*. The attributes of the data flow are source and destination tasks and communication requirements. For this last attribute named  $c$  the new data type named *CommunicationAttr* has been created with three fields: *max\_throughput* is the maximum amount of transmitted information in the time unit; *max\_delay* is the maximum permitted time to deliver data to destination; *max\_error\_rate* is the maximum number of errors tolerated by the destination. This data type is stereotyped as *GaCommHost* so that its fields *max\_throughput* and *max\_delay* are linked to *throughput* and *blockT*, respectively.

3) *Node*: A node can be seen as a container of tasks. At the end of the application design flow, nodes will become HW entities with CPU and network interface and tasks will be implemented either as HW components or as SW processes. From the point of view of network synthesis, the focus is on the resources made available by the node to host a number of tasks. Formally, the node  $n$  is a tuple defined as follows:

$$n = [t, c, p, k, \gamma, m] \in \mathcal{N} \quad \text{where :} \\ t \subseteq \mathcal{T} \quad c \in \mathbb{R}^n \quad p \in \mathbb{R} \quad k \in \mathbb{R} \quad \gamma \in \mathbb{R}^n \quad m \in \mathbb{B} \quad (3)$$

For this reason the *Node* class is stereotyped as *GaExecHost*. The attribute named  $t$  contains the set of tasks associated to the node. The attribute named  $c$  represents the resources available on the node and it has the same data type as the  $c$  attribute of the *Task*; the former represents the computation resources provided by the node while the latter represents the computation requirements needed by the task. The power budget of the node is denoted by  $p$ . The economic cost of the node is denoted by  $k$ . The vector of coefficients named  $\gamma$  is used to calculate the contribution to nodes power consumption of each task assigned to the node; a scalar product connects node's coefficients  $\gamma$  to tasks resource requirements  $c$ . The attribute named  $m$  is a boolean attribute indicating if the node can be mobile.

4) *AbstractChannel*: An abstract channel is a generalization of network channels since it contains the physical channel,

and all the protocol entities. Formally, the AC  $a$  is a tuple characterized as follows:

$$a = [n, c, p, k, d, w] \in \mathcal{A} \quad \text{where :} \\ n \subseteq \mathcal{N} \quad p \in \mathbb{R} \quad k \in \mathbb{R} \quad d \in \mathbb{R} \quad w \in \mathbb{B} \quad (4) \\ c = [\text{maxthroughput}, \text{delay}, \text{errorrate}] \in \mathbb{R}^3$$

The abstract channel connects network nodes. For this reason the AbstractChannel class is stereotyped as GaCommHost. The  $n$  attribute contains the set of nodes that communicate together by using the given abstract channel. The  $c$  attribute represents the communication resources of the given abstract channel and it has the same type of  $c$  attribute for the DataFlow; the former represents the communication resources provided by the abstract channel while the latter represents the communication requirements needed by the data flow. The economic cost is denoted by  $k$ . Since in the proposed model an abstract channel may includes intermediate systems, their economic cost shall be also considered. The attribute named  $d$  is the maximum distance between nodes which are connected by the given abstract channel; it could represent the length of a wire or the maximum range of a wireless link. The attribute named  $w$  is a boolean attribute indicating if the abstract channel is wireless. If at least one node bound to the abstract channel is mobile, then the abstract channel must be wireless.

5) *Zone and Contiguity*: To capture environment information in the UML/MARTE model, the space is partitioned into *zones* which contain nodes and the notion of *contiguity* between zones is introduced. Formally, the zone  $z$  and the contiguity  $c$  are characterized as follows:

$$z = [n, s, e] \in \mathcal{Z} \quad \text{where : } n \subseteq \mathcal{N} \quad s \in \mathbb{R}^n \quad e \in \mathbb{R}^m \quad (5) \\ c = [z1, z2, d] \in \mathcal{C} \quad \text{where : } z1 \in \mathcal{Z} \quad z2 \in \mathcal{Z} \quad d \in \mathbb{R} \quad (6)$$

The attribute named  $n$  contains the set of nodes placed in the given zone. The spatial features of the zone (e.g., its extension) are represented by the attribute named  $s$ . The environmental information (e.g., the value of temperature) is described by the attribute named  $e$ . The designer has the responsibility to complete the semantic value of  $s$  and  $e$  according to design goals. The contiguity between zones is introduced to put constraints on the reachability of the corresponding nodes. This way, the creation of network topologies can be controlled during network synthesis. The Contiguity class has two attributes named  $z1$  and  $z2$  representing the zones involved in the relationship. The attribute named  $d$  is the distance between the zones. It must be of the same type as the  $d$  attribute of the AbstractChannel since it represents the minimum value of length that an abstract channel shall have to connect two nodes placed in the corresponding zones. This implies that two nodes placed in the same zone are always able to communicate.

Objects are instantiated from these classes to represent the case study requirements and the library of available nodes and channels. Instances of the Task and DataFlow class are used to represent application requirements. Instances of Zone

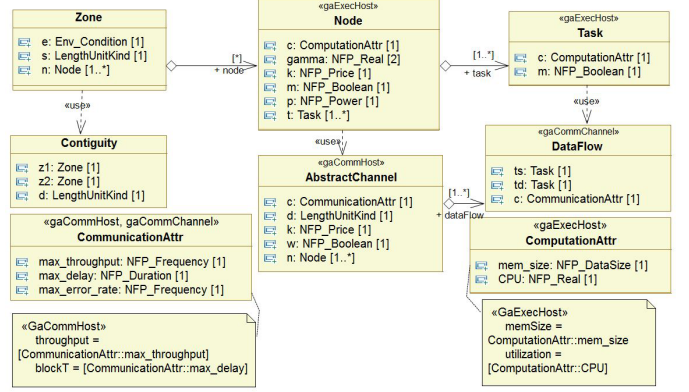


Fig. 3. Class diagram of the elements of a distributed embedded application.

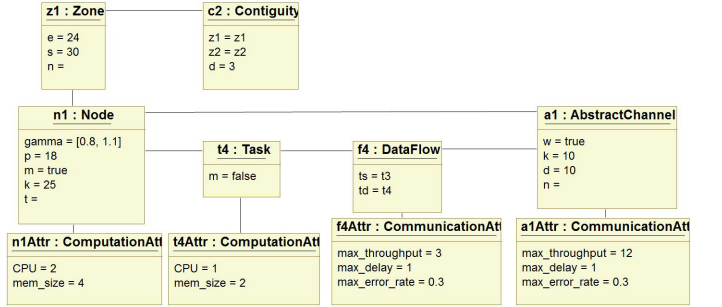


Fig. 4. Example of class instances for the building automation case study.

and Contiguity class are used to represent environment constraints. Instances of Node and AbstractChannel class are used to represent the library of actual nodes and channels. Figure 4 shows an example of class instances for the building automation case study.

It is worth to note that in MARTE specification there is no predefined attribute for the information about `max_error_rate`.

#### A. Modeling of constraints

Application constraints can specified by using cardinality on the relationships between classes. For instance, the case study has the following constraints:

- one instance of  $t4$  should be placed in each zone;
- one instance of  $t5$  should be placed in each zone;
- maximum three instances of  $t3$  can be present in the same zone, simultaneously;
- maximum one instance of  $t3$  can be assigned to a single node;
- an instance of  $t3$  is able to communicate only with other tasks in the same zone.

Figure 5 shows the modeling of these constraints with the assumption that there are five zones.

## IV. SYSTEM VIEW SIMULATION

The UML/MARTE class description in terms of tasks and data flows is used to generate a SystemC executable model



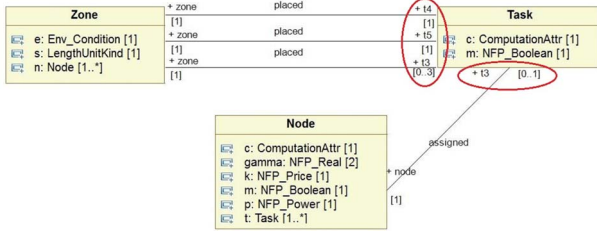


Fig. 5. Modeling of constraints for the case study application.

which reproduces the functional behavior of the application. Instances of Task class are modeled as `sc_modules` implementing the `tlm::tlm_fw_transport_if` and `tlm::tlm_bw_transport_if` interfaces. Data flows are modeled by using unidirectional transmission over TLM sockets; given a data flow from task  $t_i$  and task  $t_j$ , an instance of `tlm::tlm_initiator_socket` is created inside  $t_i$  and an instance of `tlm::tlm_target_socket` is created inside  $t_j$ ; both kinds of instances can be present inside the same task if it is source and destination of different data flows. The application code is split between the `b_transport()` member function, to handle received data, and the `do_task()` member function, to transmit data. These rules allows a semi-automatic mapping of UML/MARTE information onto SystemC/TLM. If the UML tool used to write the MARTE representation allows to specify member functions and their implementation, they are directly inserted into the generated SystemC classes.

The execution of the SystemC model allows to validate the functional behavior of the application and to fine-tune implementation details such as the content of exchanged messages and their sending rate. The knowledge of the actual format of messages and their sending rate allows to estimate the required throughput associated to each data flow. The maximum admitted delay can be assessed starting from UML/MARTE timing specification and then verified through the synchronized execution of tasks in the simulation. This information is annotated back on the UML/MARTE class diagram and it is used for the network synthesis step. Figure 4 shows these values for task  $t4$  and dataflow  $f4$ .

## V. NETWORK SYNTHESIS

All the information about user constraints, communication requirements and actual channels and nodes are extracted from the UML/MARTE model and translated into CASSE mathematical representation. For example:

$$\begin{aligned} Task(t4) &= [[1, 2], 0]. & Dataflow(f4) &= [t3, t4, [3, 1, 0.3]]. \\ Node(n1) &= [-, [2, 4], 18, 25, [0.8, 1.1], 1]. \\ Abstractchannel(a1) &= [-, [12, 1, 0.3], -, 10, 10, 1]. \\ Zone(z1) &= [-, 30, 24]. & Contiguity(c2) &= [z1, z3, 3]. \end{aligned}$$

CASSE finds different optimal assignments of tasks to nodes, flows to channels, and nodes to zones by solving a set of equations based on this mathematical representation [6]; for instance, Eq. 7 states that the tasks associated to a node

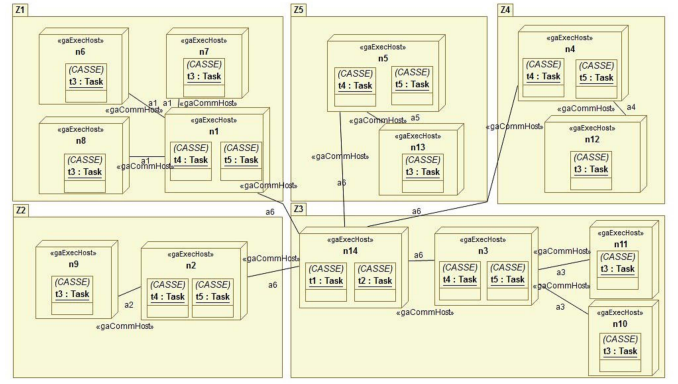


Fig. 6. Deployment diagram representing the solution of the case study.

shall not require more resources than the ones available on that node and that there must be at least one node for each zone.

$$\sum_{t \in n.t} t.c \leq n.c \quad \text{and} \quad Z_i = \{z \mid z.s = i\} \quad \forall i \bigcup_{z \in Z_i} z.n \neq \emptyset \quad (7)$$

This process is called network synthesis and its solutions are modeled by using the UML deployment diagram. This diagram consists of four types of elements, i.e., packages, HW resources, SW resources and communication paths. Zones are mapped onto packages, nodes onto HW resources, tasks onto SW resources and channels onto communication paths; several links with the same label represent a single instance of communication path, i.e., a single instance of channel shared among several nodes.

Figure 6 shows the deployment diagram of the solution of network synthesis for the case study; the presence of five zones is assumed according to the floor-plan depicted in Figure 1.

## VI. NETWORK VIEW SIMULATION

Starting from the description of the distributed system in MARTE, a new SystemC executable model can be generated with explicit representation of the network. This model uses the primitives provided by the SystemC Network Simulation Library (SCNSL) [14], i.e., nodes, channels, and protocols. For each node and channel instance of the MARTE diagram, the corresponding SCNSL object is created. Table I shows the relationship between UML/MARTE and SCNSL elements. Different types of channel models are provided by SCNSL, i.e., half/full-duplex links and wireless channels. Task implementation is taken from Step 2 of Figure 2; those `sc_modules` are connected to SCNSL nodes by using TLM sockets. During the connection of tasks to nodes, SCNSL also provides the possibility to create logical paths between tasks; this mechanism is used to represent data flows. Packet transmission and reception can be traced at task level thus obtaining statistics on packet loss rate and delay.

In particular, for the case study, wireless channel has been chosen together with the IEEE 802.15.4 CSMA/CA protocol [16] which is gaining attention in the building automation

TABLE I  
CORRESPONDENCE BETWEEN UML/MARTE AND SCNSL ELEMENTS

UML/MARTE	SCNSL
Node ( <i>nI</i> )	<code>nI = scnsI-&gt;createNode();</code>
Channel ( <i>ch</i> ) bound to node ( <i>nI</i> )	<code>CoreChannelSetup_t ccs;</code> <code>ch = scnsI-&gt;createChannel(ccs);</code> <code>BindSetup_base_t bsb1;</code> <code>scnsI-&gt;bind(nI, ch, bsb1);</code>
Data flow between task ( <i>tI</i> ) and task ( <i>t2</i> )	<code>CoreCommunicatorSetup_t ccoms;</code> <code>mac1 = scnsI-&gt;createCommunicator(ccoms);</code> <code>scnsI-&gt;bind(&amp;t1, &amp;t2, ch, bsb1, mac1);</code>

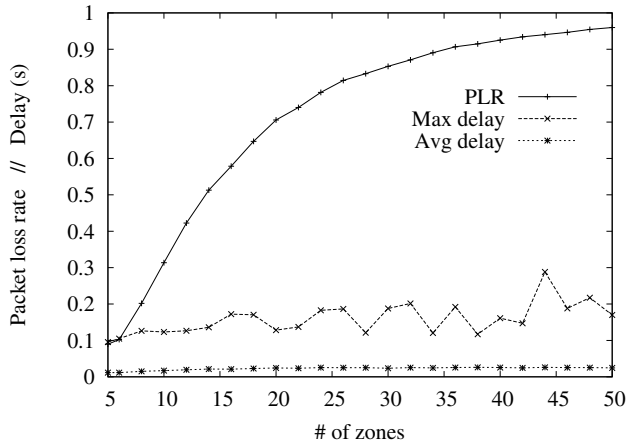


Fig. 7. Network simulation statistics: packet loss rate and delay of the communication among sensors, controller and actuators as a function of the number of controlled zones.

field. Figure 7 shows simulation statistics on packet loss rate (PLR) and average as well as max delay for the flows from sensors *t5* to the controller *t2* and from *t2* to the actuators *t4*; simulations have been performed as a function of the number of zones in which the temperature is controlled. PLR increases heavily after 15 zones due to the high number of collisions on the shared channel. The average value of delay does not increase significantly with the number of collisions since the reduced number of packets which win the contention still arrives at destination after about 0.025 s.

Network simulation provides more accurate information about the behavior of the distributed application; the max delay allows a worst case analysis which could be useful for hard real-time scenarios. Such information cannot be derived neither by the System View simulation in Step 2 nor by the analytical model in Step 3. As far as we know, no existing UML tool can provide this information. Based on network simulation statistics, the MARTE model generated after CASSE application (Figure 6) can be changed; for instance, if the distributed system must be deployed on a larger area, the number of instances of the *a6* channel can be duplicated to separate sensor and command flows.

## VII. SUMMARY OF EXPERIMENTAL RESULTS AND CONCLUSIONS

A MARTE-centric design flow for networked embedded systems has been presented and validated on a temperature control application. A class diagram has been created to

represent the building blocks of a distributed embedded application; attributes and class relationships have been used to represent user requirements and environmental constraints; a deployment diagram has been used to represent the solution of network synthesis. The obtained UML/MARTE diagrams have been successfully managed by Papyrus and used through the whole design flow and to generate documentation. Elements from the latest MARTE specification have been applied to the context of distributed embedded applications and some gaps in MARTE standard have been identified concerning the representation of constraints and attributes related to error rate information. SystemC code has been generated for both functional and network-aware simulation.

## REFERENCES

- [1] European Commission, "Hybrid control: Taming heterogeneity and complexity of networked embedded systems - HYCON," URL: <http://www.ist-hycon.org>, no. FP6-IST-511368-NOE, 2004.
- [2] A. Bonivento, L. Carloni, and A. Sangiovanni-Vincentelli, "Platform-based design of wireless sensor networks for industrial applications," in *Proc. IEEE Conf. on Design, Automation and Test in Europe (DATE)*, Mar. 2006, pp. 199–217.
- [3] L. Mottola, A. Pathak, A. Bakshi, V. K. Prasanna, and G. P. Picco, "Enabling scope-based interactions in sensor network macroprogramming," in *International Conference on Mobile Adhoc and Sensor Systems*, 2008.
- [4] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.
- [5] A. Pinto, M. D'Angelo, C. Fischione, E. Scholte, and A. Sangiovanni-Vincentelli, "Synthesis of embedded networks for building automation and control," in *Proc. of American Control Conference*, 2008.
- [6] F. Fummi, D. Quaglia, and F. Stefanni, "Modeling of communication infrastructure for design-space exploration," in *Proc. of ECSI Forum on specification & Design Languages (FDL)*, 2010.
- [7] Object Management Group, "A UML Profile for MARTE (version 1.1)," in *OMG document number: formal/2011-06-02*, June 2011, URL: <http://www.omgmarTE.org>.
- [8] Y. Vanderperren, W. Mueller, and W. Dehaene, "UML for electronic systems design: a comprehensive overview," *Design Automation for Embedded Systems*, vol. 12, no. 4, pp. 261–292, 2008.
- [9] M. Mura, A. Panda, and M. Prevostini, "Executable models and verification from MARTE and SysML: a comparative study of code generation capabilities," in *Proc. of DATE'08, Workshop on Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile*, 2008.
- [10] P. Peñil, E. Villar, H. Posadas, and J. Medina, "Executable systemc specification of the marTE generic concurrent and communication resources under different models of computation," in *Proc. of Satellite Workshop of the the 21st Euromicro Conference on Real-Time Systems*, 2009.
- [11] P. Marquet, S. Meftali, S. Niar, A. Etien, J. Dekeyser, E. Piel, R. Attitallah, and P. Boulet, "Gaspard2: from MARTE to SystemC Simulation," in *Proc. IEEE Conf. on Design, Automation and Test in Europe (DATE)*, 2008.
- [12] L. G. Murillo, M. Mura, and M. Prevostini, "Semi-automated Hw/Sw Co-design for embedded systems: from MARTE models to SystemC simulators," in *Proc. of ECSI Forum on Specification and Design Languages (FDL'09)*, 2009, pp. 1–6.
- [13] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. K. Brayton, and A. Sangiovanni-Vincentelli, "HW/SW partitioning and code generation of embedded control applications on a reconfigurable architecture platform," in *Proc. Int. Symp. on HW/SW codesign (CODES)*, 2002, pp. 151–156.
- [14] "SystemC Network Simulation Library – version 1," 2008, URL: <http://sourceforge.net/projects/scnsI>.
- [15] Sébastien Gérard et al., "Papyrus UML," URL: <http://www.papyrusuml.org>.
- [16] LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)," Sept. 2006.