Modeling and Simulation Alternatives for the Design of Networked Embedded Systems *

E. Alessio[‡], F. Fummi[§], D. Quaglia[§], M. Turolla[‡]
[§]Dipartimento di Informatica, Università di Verona, strada le Grazie 15, I-37134, Verona, Italy
[‡]Telecom Italia Lab, Via G. Reiss Romoli 274, I-10148, Torino, Italy
[fummi|quaglia]@sci.univr.it, [elisa1.alessio|maura.turolla]@telecomitalia.it

Abstract

This paper addresses the problem of modeling and simulating large set of heterogeneous networked embedded systems which cooperate to build cost-efficient, reliable, secure and scalable applications. The purpose of this task is an application-driven top-down design flow which starts from application requirements and then progressively decides the general architecture of the system and the type and structure of its HW, SW and network components. In the past, a considerable research effort has been done to create specific tools for each design domain –software, hardware and network–, and to integrate them for data exchange between models and their joint simulation. However, the advantages and drawbacks of different combinations of tools in the various stages of the design flow have not been discussed. The paper describes and discusses how to combine different modeling tools to provide different modeling and simulation alternatives for the design of networked embedded systems devoted to complex distributed applications. The problem is faced both theoretically and practically with a real application derived from a European project.

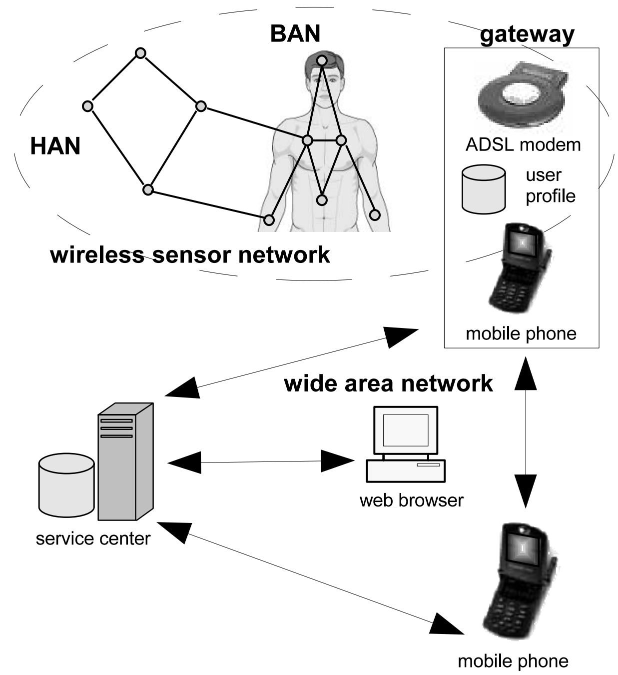

Figure 1. Example of network of heterogeneous nodes.

1 Introduction

The widespread use of embedded systems with wireless communication capabilities (e.g., PDAs, palmtop computers, cellphones, wireless sensors and actuators, etc.) has generated significant research for their efficient design and integration into heterogeneous networks. Fig. 1 shows an example of heterogeneous network composed by a wireless sensor network (WSN), a traditional wide area network (WAN), a gateway, and geographically distributed users and service centers.

A possible application-driven top-down design approach for this scenario consists in starting from the application to be provided and considering it as a kind of distributed application in which different HW and SW components in different network nodes interact through different communication networks to provide the desired functionalities. In this case, the design space consists in the choice of different *SW components* –application code, middleware and operating system– *HW components* –CPU, memory, radio interface, and application-specific devices– and *network parameters* –protocols, number of nodes, their role, network topology. Such components may be either subject of the design process or already available on the market; furthermore they should be representable through models to allow the simulation of the whole architecture. The modeling tools and

*Research activity partially supported by the European project FP6-2005-IST-5-033506 ANGEL

978-3-9810801-2-4/DATE07 © 2007 EDAA

languages should be specific for the different components to easier their representation, re-use, synthesis, and validation. Furthermore, it is desirable a cooperation among modeling tools to provide the joint optimization of the different parts of the system required by the challenging constraints of wireless networks.

In literature several modeling approaches can be found; they range from the use of general-purpose languages [4] to specific languages like SystemC [10, 2], the Software Description Language (SDL) [5], VHDL, and Matlab. Among the network simulators we should mention NS-2 [12]. SW components can be executed in emulation environments provided by instruction set simulators [1]. Each tool usually emphasizes a particular aspect of the model and the combination of different tools through the design process can contribute to give a comprehensive view of the problem.

The European project named ANGEL (Advanced Networked embedded platform as a Gateway to Enhance quality of Life) aims to develop methods and tools for building heterogeneous systems in which wireless sensor networks and traditional communication networks cooperate to monitor and improve the quality of life in common habitats, e.g., home, car and city environment. The use of networking systems in health-related application is not new; however, without a deep investigation on the design issues of the involved embedded systems, it may be difficult to develop cost-efficient, reliable, secure and scalable applications. This project propose an embedded-system-centric view of the problem of building applications based on intelligent devices where application requirements, functional aspects, hardware constraints, and network interactions are taken into account at early stages of the design flow of the whole application.

Starting from the experience of the ANGEL project, the paper describes and discuss how to combine various modeling tools to provide different modeling and simulation alternatives for the design of applications based on wireless embedded systems.

The paper is organized as follows. Section 2 presents the application-driven top-down design flow at the base of the following discussion. Section 3 describes the requirements of a modeling and simulation framework with reference to an actual application scenario. Simulation techniques to support system/network partitioning are described in Section 4. Experimental results supporting the discussion are reported in Section 5. Finally, conclusions are drawn in Section 6.

2 Application-driven top-down design flow

Fig. 2 shows the proposed design flow; it starts from the specification of application requirements (e.g., functions, cost, speed, area, and power consumption). From these requirements, a model of the whole distributed application is build as the interaction of modules connected by communication channels. Communication primitives are provided by the modeling tool and, at this stage, there is no distinction between intra-node communications and inter-node communications. Among various system modeling languages, SystemC [10] can be used for this approach for its great flexibility in describing systems at different abstraction levels and for its support of Transaction Level Modeling (TLM) [13] whose communication primitives can be employed at this stage of the design flow. SystemC is a C++ class library that provides the constructs required to describe modules performing actions and sensible to inputs.

Then *System/Network partitioning* is applied to this model to map modules to network nodes; an integer number of modules can be assigned to each node. We refer to the model of each node as *system model*. Communications between modules belonging to different nodes are described as network communications and are part of the *network model* which reproduces the behavior of network protocols such as TCP/IP or ZigBee/802.15.4. System and network models can be simulated by different tools even if they should interact to exchange data and to have a common simulation time scale (horizontal arrow in the Figure); well-known co-simulation techniques [8, 6] can be used for this purpose.

In the next stage, system models which are outside the design scope remain unchanged while a traditional design flow can be applied to each system model to be developed (e.g., transaction level modeling [13]). In particular, *HW/SW partitioning* is performed on the system model to map functionalities to HW and SW components according to several constraints (e.g., speed, cost, and component availability). HW components outside the design scope remain unchanged while others are refined down to RTL and gate level. In this phase simulation can be done with different tools for SW and HW and network components but they should cooperate to exchange data and to achieve a common simulation time scale. In particular, SW components interact with HW components and HW components interact with the network model (horizontal arrows in the Figure); interactions between HW and SW components and between HW and network models can managed by well-known co-simulation techniques [1, 7, 9] and [8, 6] respectively.

In the final stage, HW and SW models are mapped to actual actual components. HW components can be either synthesized or taken from the market. For what concerns software, an actual operating system is introduced and SW functionalities become application code with calls to the OS services. An actual network is deployed according to its model.

It is worth remarking that refinement, simulation and validation can be accomplished on the models at each step of the design flow until they become actual objects. In particular, simulation is important to guide partitioning phases; in Section 4 the role of different simulation approaches is de-

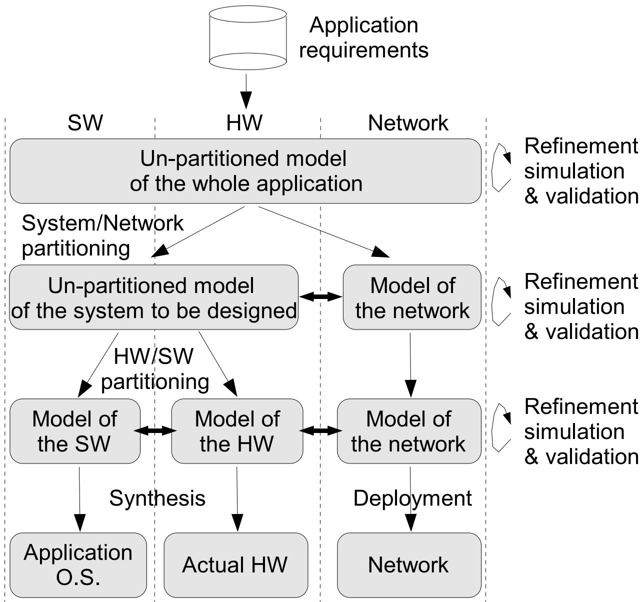

Figure 2. Application-driven top-down design flow.

scribed in the context of system/network partitioning while in Section **??** it is described in the context of HW/SW partitioning.

3 Description of the application scenario

This Section describes the application scenario depicted in Fig. 1 with details about functionalities and role of each node of the heterogeneous network; furthermore, the requirements of the simulation tool and the expected outputs will then be derived and mapped to the design flow shown in Fig. 2. The application scenario refers to personal wellness and related services that can be offered to either healthy people or persons with some medical disease, to improve the quality of everyday life, providing continuous indoor and outdoor remote assistance.

A body area network (BAN) is foreseen, made up of portable or wearable devices to monitor health parameters (e.g., heart rate, blood pressure) and body activity (e.g., to detect falls, running, absence of movement). The BAN can also be extended to a home area network (HAN) including sensors that are not attached directly to the person like a bathroom scale and sensors for temperature and humidity that, combined with body data, can be used to derive more precise health suggestions and assistance. A gateway connects the sensor network to the telecom operator distribution network; the gateway is an embedded system with higher processing and memory capabilities with respect to sensor nodes and it can be used to *1)* configure, monitor and control the sensor network, *2)* collect and send data to a remote service center, and *3)* locally process data, for example to manage alarms to be sent to the service center or specific users. The gateway can also contain user-related information which cannot be stored elsewhere for privacy reasons (e.g., health profile or regular values for blood pressure and weight). In indoor environments, the gateway can be represented by a fixed telecom appliance (e.g., an ADSL router) while in outdoor environments a mobile terminal can be used. The remote service center collects and processes data, sends alarms to specific remote users, and provides counseling sessions. Historical data, its interpretation and personal alarms can also be accessible through an access-restricted web site.

A first requirement for the simulation framework regards the study of the *reliability* of both the network and its supported services; in this context the evaluation of parameters such as node failure rate, packet loss/error rate and delays are key points. These parameters depend on the design of both the network and nodes. For what concerns the network, the design choices are the number of nodes, their role, distance, the presence of redundant paths, and radio interference. For what concerns nodes, the power consumption of internal components and battery capacity are the main design variables.

Another requirement of the simulation tool is the evaluation of *performance* for both the network and its supported services; in this context significant outputs are throughput, delays but also the number of intermediate nodes requested to support a specific end-to-end application. Also in this case outputs depend on both network and system design choices. For what concerns the network, delay depends on end-to-end round trip time, node association time, and access contentions; for what concerns the system, throughput and delay depend on CPU power, SW complexity and duty cycle (which also affects power consumption).

One other major problem when designing a wireless sensor network is its *liveness* which depends on power consumption; the design of nodes that provide different power modes and the optimized development of software and applications having a proper duty cycle is a crucial point, as well as the right choice of batteries trading-off among size (e.g., for wearable devices), capacity and cost.

Another simulation requirement, directly related to the design of a single node (either a sensor or the gateway itself) is the evaluation of hardware resources, i.e., memory, CPU, and application-specific HW.

4 Simulation for Sys/Net partitioning

This Section presents different modeling and simulation strategies supporting system/network partitioning. For each solution, the inputs and outputs of the framework will be described and advantages/drawbacks will be highlighted.

4.1 SystemC only

The use of SystemC is a possible solution for the modeling of a set of interacting embedded systems. Since in this case a single tool is used, simulation speed is high thus accelerating the development-validation cycle. This solution is clearly applicable to the first stage of the design flow depicted in Fig. 2. In this case, communication requirements cannot be verified since network is not modeled.

SystemC can also be used to model network behavior and thus be applied to the second stage of Fig. 2. In literature SystemC was already used to describe network-on-chip architectures [3], to simulate the lowest network layers of the Bluetooth communication standard [2], and to reproduce the behavior of wireless sensor networks [9]. The main drawback of this solution is that the designer has to implement network protocols since, as far as we know, there is no available library.

4.2 NS-2 only

This case consists in the use of a single simulator tool, i.e., a network simulator named NS-2 [12] to model the wireless sensor network, the gateway and the wide area network. NS-2 is a well-known network simulator in the network research area; even if it is traditionally used to model TCP/IP networks, wireless features have been recently introduced and, in particular, the IEEE 802.15.4 standard for low-rate personal area networks. This feature can be used to model the wireless sensor network while traditional TCP/IP models can be used to model the wide area network.

For what concerns the wireless sensor network, different parameters can be set, i.e., distance between nodes, signal attenuation law, TX power, RX sensitivity, and speed of mobile nodes. Furthermore NS-2 provides a simple power model which takes into account the power spent by the node during transmission, reception and in sleep mode. For what concerns interferences, only a basic statistical model can be used by specifying packet loss rate and radio channel swap probability. Also the interaction with the application layer and the physical sensors can be described only in terms of probability distribution of user's requests and arrival of stimuli, respectively. For what concerns the wide area network, TCP/IP can be considered a good abstraction over various access technologies. In this case, bandwidth and delay can be set for each link connecting the gateway with remote users and service centers.

This approach allows to find the throughput and delay of the sensor network as a function of the number of nodes, their role in the ad-hoc network (e.g., coordinator), their distance, their speed. Reliability can be assessed by providing redundant paths in the topology. A rough result about power consumption can be obtained as a function of network activity.

The advantage of this approach is that simulation is fast for the presence of a single tool. If compared to SystemC, NS-2 simulations requires more memory and this can be a problem when the network contains more than 1,000 nodes. Furthermore the NS-2 model of sensor nodes and the gateway does not catch their internal structure as a set of interacting HW/SW components; a consequence of this limit is that resulting performance does not take into account delays due to computations within the nodes.

4.3 NS-2 and SystemC

As reported in the description of the application scenario, simulation outputs such as throughput and node failure rate depend on both the network and the system design choices. The use of specific tools for network and system simulation can simplify the creation of models. The network simulator can also be used to rapidly create a realistic testbed for system validation; in particular, the implementation of the communication structure of the embedded system can be validated through the successful interoperation with the protocol stack described at higher abstraction levels by the network modeling tool. System/network co-simulation is mandatory when we want to test the effect of HW and SW on the reliability and performance of the application

With respect to single-tool approaches, system/network co-simulation is slower because of the communication between the two simulation kernels. A possible solution to this issue is two-step approach; in the first step network statistics (e.g., number of transmitted/received packets) is generated from a pure networking simulation; the obtained information can then be fed in a second step into an instruction-level/hardware model to obtain the detailed behavior of the system. However, this approach cannot be used when the behavior of the node also affects network parameters as in the case of flow-control algorithms.

5 Test case

To show the proposed modeling flow we designed a simple test application for the scenario described in Section 3. Sensor nodes receive stimuli from the environment in the form of integer values. When a stimulus is received, the corresponding node sends a packet reporting its value, the location of the node and the timestamp of the reception. Stimuli are generated every 1 s. Since wireless sensor networks do not provide direct node addressing, a routing algorithm and multi-hop transfers are needed for data delivery from sources of stimuli to the gateway; for simplicity's sake received packets are re-sent in broadcast by all nodes except the gateway; more complex routing algorithms can be used [11].

The gateway collects data from the received packets and creates a table in which, for each stimulus location, the av-

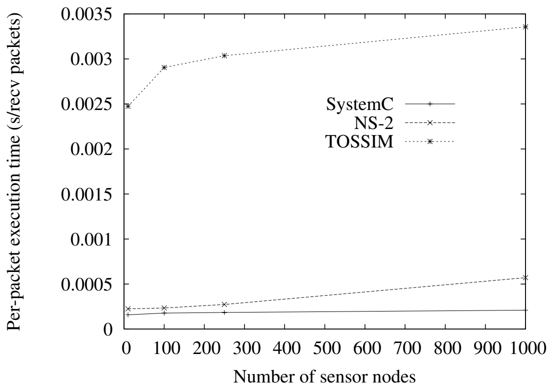

Figure 3. Per-packet execution time as a function of the number of nodes for different simulation tools.

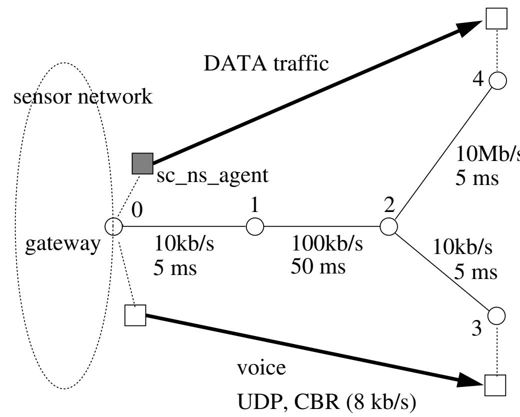

Figure 4. NS-2 model of the WAN.

erage of the sampled values is reported. Every 10 s the table is delivered over the Internet to a remote host.

5.1 Model of the sensor network

The sensor network is completely modeled in SystemC by using the simulator described in Section **??**. This tool enables the modeling of a set of heterogeneous nodes and favorites cross-layer design and synthesis. Furthermore, during preliminary tests, it provided a higher simulation speed with respect to TOSSIM and NS-2 as shown in Fig. 3.

A sub-class of the `Node` module is created to implement the specific relaying functionality. In this class, the reception either of a stimulus or of a packet triggers the execution of appropriate `SC_THREAD` methods which put a new packet into a queue. Another method performs carrier sense and transmits enqueued packets when the channel is free. An instance of this class has been created for each sensor node. Sources of stimuli are represented by instances of a sub-class of the `Stimulus` module in which a clocked `SC_THREAD` generates a random value every 1 s.

5.2 Model of the wide area network

The wide area network is modeled by NS-2 to exploit its full support of standard protocols and traffic models. Fig. 4 graphically represents the NS-2 model. A classical bottleneck topology has been created to represent a backbone with access links. Node 0 represents the gateway. The 10 kb/s links between node 0 and node 1 and between node 2 and node 3 represent mobile connections. The link between node 1 and node 2 models a geographical backbone with 100 kb/s of available capacity and 50 ms delay. Node 3 represents another mobile user talking with the gateway. Node 4 represents a host connected to Internet through a 10 Mb/s access link. Two agents reproduce an UDP connection carrying sampled data from node 0 (the gateway) to node 4 (the Internet host). The `sc_ns_agent` on node 0 connects NS-2 with the network interface of the gateway modeled in SystemC; the technique described in [6] is used. Other two agents model a UDP flow carrying voice from the gateway to node 3 (the listener); an 8 kb/s constant bit-rate (CBR) traffic model is applied to the agent on node 0 to reproduce a mobile voice connection. Statistics on delay and packet drops are generated by NS-2 and used to evaluate the quality of service.

5.3 Model of the gateway

The gateway has been modeled at system-level in SystemC. Three modules have been created: the interface to the WSN, the interface to the wide area network, and the processing module. The interface to the WSN is a sub-class of the `Node` module of the WSN and interacts with other nodes through the SystemC simulator. The interface to the wide area network contains special ports to interact with the NS-2 simulation kernel as described in Section 4.3. The processing module is connected to both interfaces through signals and queues; it implements the creation of the table containing the average of the sampled values.

5.4 Experimental results

In the experimental setup, nodes are equally spaced over a line. The node at one end of the line receives environment data every 1 s and broadcasts a packet containing the sampled value over the radio interface. Antennas are omni-directional and the distance between nodes is such that each node can reliably communicate with the adjacent nodes only. This scenario leads the simulator to handle the problem of hidden terminals since it can happen that nodes that are not able to hear each other when they are sending, disturb each others transmissions in a receiving node. When

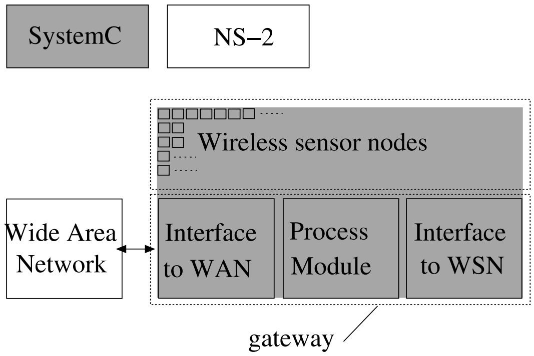

Figure 5. Co-simulation framework with a system-level description of the gateway.

Nodes	SystemC+NS
10	30.9 s
100	504.3 s
250	534.4 s
1000	1717.9 s

Table 1. CPU time of the system-level simulation as a function of the number of nodes in the WSN.

a node receives a packet, it re-broadcasts the message contributing to deliver it to the other end of the network. This routing protocol, a kind of *unselective flooding*, is not the best choice for a real sensor network since maximizes the number of transmitted packets, but has the advantage to test the performance of the simulator under heavy load conditions. The simulation length is 40 s and total execution time is reported as a function of the number of nodes to test the scalability of the technique. Tests have been performed on a Linux workstation.

Table 1 reports the total execution time of the system-level simulation in which the gateway and the sensor network are modeled by using SystemC and the wide area network is modeled by NS-2. The total wall-clock time has been considered since the simulator consists of different concurrent processes and it is difficult to obtain the actual *aggregated* CPU time for them. Results show that the total elapsed time strongly depends on the size of the WSN. With ten nodes the simulation is faster than real-time.

6 Conclusions

We have presented and discussed different modeling and simulation alternatives to support the design of applications based on heterogeneous networked embedded systems. In particular, such alternatives consist in the combination of different modeling tools, each of them highlighting a specific aspect of the application. We have applied this approach to an application scenario derived from a European project and consisting in wireless sensor nodes interacting with traditional networks through a gateway. System-level simulation of a simplified scenario shows that the total elapsed time strongly depends on the size of the wireless sensor network; with ten nodes the simulation is faster than real-time.

References

[1] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino. Systemc cosimulation and emulation of multiprocessor soc designs. *IEEE Computer*, 36(4):53–59, Apr. 2003.

[2] M. Conti and D. Moretti. System level analysis of the bluetooth standard. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, volume 3, pages 118–123, Mar. 2005.

[3] D. Bertozzi et al. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):113–129, Feb. 2005.

[4] D. Desmet, M. Esvelt, P. Avasare, D. Verkest, and H. De Man. Timed executable system specification of an ADSL modem using a C++ based design environment: a case study. In *Proc. of IEEE CODES*, pages 38–42, 1999.

[5] D. Dietterle, J. Ebert, G. Wagenknecht, and R. Kraemer. A wireless communication platform for long-term health monitoring. In *Proc. IEEE International Conference on Pervasive Computing and Communications Workshop*, Mar. 2006.

[6] F. Fummi, P. Gallo, S. Martini, G. Perbellini, M. Poncino, and F. Ricciato. A timing-accurate modeling and simulation environment for networked embedded systems. In *Proc. ACM Design and Automation Conf. (DAC)*, pages 42–47, Jun. 2003.

[7] F. Fummi, M. Loghi, S. Martini, M. Monguzzi, G. Perbellini, and M. Poncino. Virtual hardware prototyping through timed hardware-software co-simulation. In *Proc. of the IEEE Design, Automation and Test in Europe Conf. (DATE)*, volume 2, pages 798–803, Mar. 2005.

[8] F. Fummi, S. Martini, G. Perbellini, M. Poncino, F. Ricciato, and M. Turolla. Heterogeneous co-simulation of networked embedded systems. In *Proc. IEEE Design Automation and Test in Europe Conf. (DATE)*, pages 168–173, Feb. 2004.

[9] F. Fummi, D. Quaglia, F. Ricciato, and M. Turolla. Modeling and simulation of mobile gateways interacting with wireless sensor networks. In *Proc. IEEE Design Automation and Test in Europe Conf. (DATE)*, Mar. 2006.

[10] T. Grotker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Springer, 2002.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of ACM MOBICOM*, Boston, MA, Aug. 2000.

[12] S. McCanne and S. Floyd. NS Network Simulator – version 2. *URL: http://www.isi.edu/nsnam/ns*.

[13] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Extending the Transaction Level Modeling approach for fast communication architecture exploration. In *Proc. of DAC*, pages 113–118, 2004.