

# Towards Automatic Synthesis of a Class of Application-Specific Sensor Networks\*

Amol Bakshi  
Dept. of EE-Systems  
University of Southern  
California  
Los Angeles, CA - 90089  
abakshi@usc.edu

Jingzhao Ou  
Dept. of EE-Systems  
University of Southern  
California  
Los Angeles, CA - 90089  
ouj@usc.edu

Viktor K. Prasanna  
Dept. of EE-Systems  
University of Southern  
California  
Los Angeles, CA - 90089  
prasanna@usc.edu

## ABSTRACT

Automatic synthesis of sensor network-based systems can be described as the process of translating a formal specification of application functionality into a particular task mapping, settings of available hardware knobs, and communication and coordination mechanisms among the sensor nodes, so as to meet the performance requirements and constraints. We propose a general methodology to tackle a specific class of this problem, based on analytical performance modeling, multigranularity system simulation, and automatic refinement of model parameters. To demonstrate the utility and feasibility of our proposed methodology, we define a system model for a class of sensor networks, and implement a software framework for its modeling and simulation. Our graphical design environment supports plug-and-play integration of different performance models, simulation and visualization suites, and even automatic design space exploration and optimization tools.

## Categories and Subject Descriptors

C.4 [Performance of Systems]: Measurement techniques, Modeling techniques; C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Wireless communication*; I.6.7 [Simulation and Modeling]: Simulation Support Systems—*Environments*

## General Terms

Performance, Design, Experimentation

---

\*This work is supported by the DARPA Power Aware Computing and Communication Program under contract no. F33615-C-00-1633 monitored by Wright Patterson Air Force Base.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CASES 2002, October 8–11, 2002, Grenoble, France.  
Copyright 2002 ACM 1-58113-575-0/02/0010 ...\$5.00.

## Keywords

sensor networks, energy efficiency, design environments, automatic synthesis

## 1. INTRODUCTION

Inexpensive, low-power sensor nodes that monitor the environment and perform limited processing on the samples to detect events of interest are fast becoming a reality. The applications of such networks range from target tracking based on acoustic signatures and line-of-bearing estimation to climate control, intrusion detection, etc. Energy efficiency is of special interest to the sensor network community because it is undesirable, difficult or impossible to replenish the energy resources available to a sensor node, once deployed. Maximizing the life of sensor nodes is an overriding priority, and different energy optimization techniques are being developed [6, 10, 15, 16, 21, 22] to address computation/communication tradeoffs.

With increasingly sophisticated applications, and advances in digital and RF technology, designers of sensor network-based systems<sup>1</sup> will soon be faced with a very large set of design decisions. Each of the choices will affect the overall system performance in ways that might not always be ‘cleanly’ modeled. In addition to the research challenges in design and optimization, design tools for real-world sensor networks are equally important. It is the latter aspect that is the focus of this paper. For example, the ability of the design framework to allow rapid specification and evaluation of a particular network configuration is crucial for a more exhaustive exploration of the design space. A design environment for future sensor networks should provide tools and formal methodologies that will allow designers to model, analyze, optimize, and simulate such systems. Extensibility, i.e., the ability of the framework to accommodate new tools for simulation, optimization, and/or visualization is also desirable. Ultimately, the design framework should require only a high-level specification of the desired application functionality, available resources, and performance requirements, and should be capable of *automatically synthesizing* efficient sensor network configurations. In this paper, we present a methodology for automatic synthesis of a class

---

<sup>1</sup>“Sensor networks” and “sensor network-based systems” are used interchangeably in this paper to denote the overall application functionality implemented using a network of sensor nodes as the underlying hardware.

of sensor networks, and a design environment to illustrate its feasibility and usefulness.

We consider sensor networks where the nodes can be placed randomly, but are not mobile. Also, the sampling frequency is pre-determined and does not change at run time. Sections 2 and 3 describe our model in more detail. Using the taxonomy of wireless micro-sensor network models proposed in [19], the class of sensor networks we address can be described as *static* sensor networks with a *continuous* data delivery model, and a *one-to-one unicast* communication with the cluster-head. This model has been chosen to later illustrate how our methodology and framework can be used to aid automatic synthesis of the class of sensor networks it represents. The design methodology and the capabilities of our design environment are not restricted to this specific class of systems.

Most sensor networks are designed to execute a few specific applications, and are not meant for general-purpose computing and communication. For such application-specific systems, a proper allocation and coordination of tasks among sensor nodes (at design time) is as important as an efficient network protocol. Energy savings of upto 60% have been demonstrated merely by parallelizing computation through the network, and exploiting power management hooks such as dynamic voltage scaling, *without changing the network protocols* [21]. We therefore put as much emphasis on node-level modeling and performance analysis as is usually (and sometimes exclusively) placed on network-level issues.

The first main contribution of this paper is a methodology for automatic synthesis of application-specific sensor networks, based on high-level performance modeling, multi-granularity system simulation, and model refinement. The second is a design environment for sensor network modeling and simulation that supports this methodology. Our design tool allows rapid graphical specification of the target system, and automatically configures and executes a set of low-level simulators to obtain accurate node-level and network-level performance statistics.

This paper is organized as follows. The general synthesis problem for sensor networks is defined in Section 2. The system model for the class of application-specific sensor networks we consider in this paper is the subject of Section 3. Section 4 describes our general design methodology and presents a high-level performance model. Section 5 provides details of the design environment using an illustrative case study. Comparison with related work is made in Section 6, and we conclude in Section 7.

The work in this paper was motivated by the Automatic Target Recognition (ATR) challenge application of the DARPA Power-Aware Computing/Communications program. We generalize and significantly extend our preliminary work on modeling and simulation of this 7-node sensor network [3].

## 2. THE AUTOMATIC SYNTHESIS PROBLEM

Automatic synthesis for application-specific sensor networks involves a holistic view of design and optimization both at the node level and the network level. This paper does not define a specific technique for system-wide energy optimization. Instead, we propose a general methodology that addresses key issues in system-wide design, typically not considered by research work focusing on just one aspect

of the overall design problem. As an example, the energy-efficient communication protocol in [10] assumes that all sensor nodes are homogeneous, and even distribution of data fusion tasks among nodes helps in maximizing the system life span. However, the degree to which this reduced communication energy impacts the overall system will depend on factors such as the application itself, node architecture, task mapping, communication schedule, switching schedule between idle/active modes of node components, etc. For instance, each node in [10] will require sufficient resources to be able to act as a cluster-head when required. Storage cost might be an important factor, determined by the size of data samples, sampling frequency, latency of task execution, type of memory used in the node, etc. Such concerns cannot be ignored at the system-level. From an automatic system-wide synthesis perspective, the designer should be able to evaluate, say, the combined effect of system partitioning [21] and a particular medium access control (MAC) protocol [16].

This paper deals with wireless sensor networks that are based on the ‘conventional’ clustering model. Sensor nodes are grouped into clusters based on geographical proximity, and are equipped with low-power (short-range) radios. One of the nodes in each cluster is designated *a priori* as the cluster-head, whose job is to transmit the result of data fusion performed within the cluster, to the observer. Similar to [20], the cluster-head will be a high-energy node that could be equipped with a GPS receiver, long-range radio, etc. Our network is *static* - i.e., nodes can be arbitrarily distributed in a 2-D region, but they are not mobile. Since the functionality of each node (i.e., whether it is a sensor node or a cluster-head) is determined a design time, and the communication network is a simple single-hop wireless transmission, no self-organization takes place within a cluster at run time. However, at the higher level of hierarchy, cluster-heads will typically set up an ad-hoc network among themselves to route data in an efficient manner to the eventual destination(s).

*Automatic synthesis of an application-specific sensor network can be defined as determining the allocation of logical tasks to physical sensor nodes, and synthesizing the inter-node communication and co-ordination mechanism to accomplish the desired overall functionality - while meeting user-specified performance requirements and constraints [2].* The synthesis problem can take different forms. One variation can be determining an energy-efficient node placement when the number of sensor nodes, task-to-node mapping and node architecture is fixed. Another design problem can be deciding the best node architecture for a system where task-to-node mapping is known but sensor nodes can be randomly deployed.

The version of the synthesis problem addressed in this paper assumes that the end-to-end application is described as a task graph. The specification of the hardware architecture of radios, sensors, processors, memories, etc., is also available. Sampling frequency, sample size of sensors, and end-to-end latency and throughput constraints are known. Given this ‘static’ description, the output of synthesis is the allocation and schedule of tasks among different nodes in the sensor network, an energy-efficient schedule for intra-cluster and inter-cluster data transmissions, the settings of available architecture knobs, such as dynamic voltage scaling, etc.

This paper does not claim to solve the general automatic

synthesis problem even for the well-defined system model presented in the next section. What we propose is a design methodology for automatic synthesis, based on the key concepts of high-level performance modeling, and multi-granularity simulation for model refinement. Our software framework provides a platform for automatic synthesis based on this methodology, and is a significant step towards a specific class of the general problem.

### 3. SYSTEM MODEL FOR A CLASS OF APPLICATION-SPECIFIC SENSOR NETWORKS

Let  $C$  ( $C \in \mathbb{N}$ ) denote the number of sensor types in the network. Examples of sensor types include acoustic, seismic, thermal, pressure, etc. Each sensor type  $t$  ( $1 \leq t \leq C$ ) is associated with a 3-tuple  $(N_t, D_t, F_t)$ , where  $N_t$  is the number of sensors of type  $t$  in the system,  $D_t$  is the size of one data sample (in bytes), and  $F_t$  is the sampling frequency for that sensor type.

We partition the computation tasks into two categories - *sensor processing* and *collaboration*. Some amount of processing is usually done for every sample separately. The result of this processing can be a binary value (with associated semantics), or a modified data set. The results from all the sensors are then processed to obtain the overall picture of the state of the environment, such as the location of a target being tracked, weather information, etc.

For each sensor type  $t$  ( $1 \leq t \leq C$ ), the processing required for each data sample  $D_t$  is specified in the form of a data flow graph  $G_t = (V_t, E_t)$ .  $G_t$  is a directed acyclic graph. Each directed edge  $e_i^t \in E_t$  connecting the pair  $(v_i^t, v_j^t)$  represents a data dependency of  $v_j^t$  on  $v_i^t$ . Each vertex  $v_i^t$  of the graph denotes an atomic computation step on the input to the task.

There is a single source node  $v_{sns}^t$  for each graph  $G_t$  which represents data input from the sensor. In our model, the sensor and a radio receiver are the two ways a node can acquire data, and for the *sensor processing* category of task graph, the sensor is the only relevant source. No processing is associated with the node  $v_{sns}^t$ . There is a single destination (sink) node  $v_{ch}^t$  node for each graph  $G_t$  which represents the result of the sensor processing phase.

Each edge  $e_i^t$  is associated with an integer  $d_{ij}$  that quantifies the amount of data transfer (in bytes) between the tasks  $i$  and  $j$  connected by that edge. Note that we use a block data processing model as against stream processing. One of the reasons for this is that low-level cycle-accurate simulation of tasks on the processor core will yield more accurate per-byte cost estimates if the overheads are amortized over a larger block of data. From a simulation accuracy point of view, it is the designer's responsibility to ensure that the size of the data block processed by the actual software implementation associated with the task is the same as the size ( $d_{ij}$ ) specified in the model.

All collaborative computation is represented by a single task graph  $G_{cc} = (V_{cc}, E_{cc})$  for the entire system. The vertex and edge semantics are the same as described in the *sensor processing* graphs. The significant difference is in the source and destination nodes. The task graph  $G_{cc}$  has  $C_N$  number of distinct source nodes, where  $C_N = \sum_{t=1}^C N_t$ . The single destination node  $v_{xmt}$  represents the result of the collaborative computation that is transmitted from the cluster-

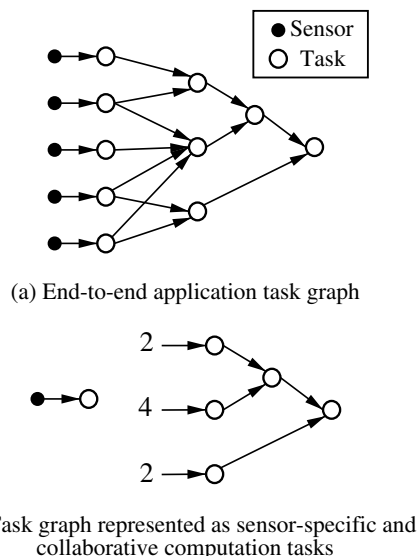


Figure 1: Application Modeling

head to some entity outside the cluster. Again, there is no processing associated with the source and destination nodes themselves. This representation allows for parallel computing paths in the collaboration task graph; but all have to converge to the single destination node.

Figure 1(a) shows an expanded application graph that represents data from 5 sensors being processed, first individually, then collectively. The representation of this task graph in terms of our application model is shown in Figure 1(b). In the particular instance shown in the figure, all sensors are of the same type and all inputs to the collaborative tasks are equivalent, i.e. there is no association of a specific sensor node with a particular input to the collaborative task. The model for more complicated applications will be different than the example shown.

From the hardware resource modeling perspective, let  $H$  be the number of sensor nodes in the system. One node can have more than one sensor. The sensors can all be of different types or even the same type, depending on the functionality of the sensor. For example, multiple seismic sensors on the same node will probably not add any value, but multiple cameras monitoring different directions from the same sensor node can be envisaged. In the simple scenario where only one sensor of a given type is present on a particular node,  $H \geq \max(N_t), 1 \leq t \leq C$ .

In addition to the sensor(s), each sensor node  $S_i (1 \leq i \leq H)$  includes a processor core  $P_i$ , a memory module  $M_i$ , and a radio card  $R_i$ . Some parameters that could be varied at design time are the voltage/frequency of the processor core, the size of the memory module, the transmit and receive power of the radio, and the bandwidth of the radio. Varying the parameters at design time can translate into setting different values of the same 'knob' on the same hardware module (if the facility is available) or choosing a different hardware module entirely.

The source nodes  $v_{sns}^t (1 \leq t \leq C)$  in the *sensor processing* category of task graphs are associated with the corresponding hardware node which contains that sensor. Also, the destination node  $v_{xmt}$  of the *collaboration* task graph is

bound to the cluster-head which contains the transmission facility to an entity outside the cluster. For a given performance criteria, the mapping of all other tasks is to be determined as part of the synthesis problem.

## 4. DESIGN METHODOLOGY

### 4.1 Approach

The first step in our methodology is the use of simple analytical models for performance analysis at both the node level and network level. High-level models are essential for obtaining rapid and reasonably accurate performance estimates in the initial phase when a large design space has to be narrowed down to a manageable set of good designs. Our design environment allows the user to manipulate many system parameters and thereby model different hardware architectures, mappings, network protocols, etc. Next, plug-and-play integration of low-level simulators allows the investigation of performance in greater detail. Finally, results from low-level simulations can then be used for (semi-)automatic refinement of the analytical model parameters. This general approach is also at the heart of the MILAN project [4, 12].

Coarse performance models are critical for automatic application synthesis. Simulating a large number of design choices using fine-grained models has two disadvantages: low-level simulations are time consuming, and the effect of many parameters on performance is not significant enough to justify the cost incurred by including them into the simulation model. The level of detail that should be considered to correctly analyze and optimize for the particular design problem at hand is unclear, especially for wireless networks [9]. For example, increase in communication energy due to interfering transmissions can be accurately computed by considering, among other things, the precise geometry of the network. However, if the actual geometry can be determined only at run time, the effect of actual node positions on transmission delays needs to be approximately modeled at the high level.

Even cycle-accurate system-wide simulation has many challenges. None of the existing network simulators to our knowledge models the internal architecture of the processing nodes in the network, because the focus of network simulation has traditionally been on protocol development and empirical analysis. Also, most processor simulators do not model the environment outside the node boundary. To obtain detailed performance estimates for the entire system, it is necessary to make simulators interact with each other in some way. In [3], we proposed a technique to automatically generate scenarios for the ns-2 network simulator [14] based on results from node-level simulations using Wattch [5]. Watch is based on the SimpleScalar simulator that models a superscalar RISC architecture. In addition to SimpleScalar statistics, Watch also provides an estimate of energy consumption in the processor for the simulated application program. This ‘horizontal’ simulation is accomplished without requiring the simulators to directly interact with each other. Such coarse-grained integration might compromise the accuracy of the system-wide results to some extent, but the effort saved by not modifying existing simulator code outweighs the (possible) marginal errors.

We have implemented a framework that can be used by optimization tools for different design problems. The following sections describe the model and our tool in detail.

### 4.2 Performance Model

Assume that the task graph is partitioned along the edge  $e_{ij}^t$ , tasks  $(v_{sns}^t \cdot v_i^t)$  are executed on the sensor node and  $(v_j^t \cdot v_{xmt})$  are executed at the cluster-head.  $e_{ij}^t$  represents data transfer over the wireless link. Each task is executed once for each sample. In the absence of buffering, the task graph is executed  $F_t$  times per second. As our data delivery model is *continuous*,  $F_t$  does not vary with time.

Let  $(t_1, \dots, t_k)$  denote the set of  $k$  tasks  $(v_{sns}^t \cdot v_i^t)$ . Let  $c_i$  be the number of computations in task  $t_i$ , and  $e_i$  and  $l_i$  the average energy consumed and average cycles required per computation respectively. If  $f_p$  is the processor frequency of the node, it follows that the percentage of time that the processor is in active mode (i.e., its utilization) is  $(F_t \sum_{i=1}^k c_i l_i) / f_p$ .

Power dissipation in the node is given by

$$F_t (\sum_{i=1}^k c_i e_i) + \alpha [1 - (F_t \sum_{i=1}^k c_i l_i) / f_p] + \beta,$$

where  $\alpha$  is the power dissipation in idle mode, and  $\beta$  is the average power consumed in switching between active and idle modes<sup>2</sup>. A similar calculation can be done for the remainder of the tasks mapped onto the cluster-head.

On an ideal wireless link, the power required for data transmission to the cluster-head is  $\gamma d_{ij} F_t$ , where  $\gamma$  is the transmission energy per byte of data. In the real world, additional factors need to be accounted for. For purposes of our high-level power modeling, we classify all such effects into two categories - overheads due to data encoding, and overheads due to data retransmissions that result from collisions, interference, etc. The error rate of the channel can be modeled in terms of encoding overhead; because a more robust encoding will be required for a high bit error rate (BER). Also, depending on the MAC protocol used, different energy cost for communication can be associated with an idle node - i.e., an idle but listening node consumes much more energy than an idle node which shuts off the radio completely.

The power consumed for communication is therefore given by

$$\gamma (\epsilon \delta F_t d_{ij}) + \mu (1 - (F_t d_{ij} / b))$$

where  $\epsilon (\geq 1)$  represents the overhead due to retransmissions,  $\delta (\geq 1)$  represents the overhead due to encoding,  $\mu$  is the power dissipation in idle mode of the network, and  $b$  is the bandwidth (bytes/sec) of the radio.

This model is useful because it captures the node architecture, network configuration, and also application-level issues such as the task-to-node mapping. It is therefore suitable for system-wide performance analysis from an application-specific system synthesis point of view. A similar coarse-grained system model that considered both node and network-level parameters was proposed in [15], and used to demonstrate possible design optimizations for the 7-node ATR application. Compared to [15], our model explicitly captures communication parameters such as retransmission, error rates, idle time power consumption, etc.

In this paper, we do not use this model for optimization of the case study described in the next section. Our GME-based design environment does support plug-and-play integration of performance models into a high-level estimation framework, and also automatic optimization tools based on

<sup>2</sup>Idle time in our node-level model excludes the radio, whose on/off schedule is a part of the active/idle time at the network level

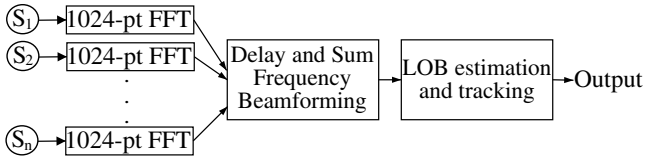


Figure 2: The ATR Application

performance models. Therefore, the designer has complete freedom of choosing whichever high-level model and/or optimization tool he/she desires to use for a particular design problem. [13] discusses automatic design space exploration for heterogeneous embedded system architectures in the MILAN framework, which is employing the same general methodology and a GME-based modeling and simulation environment as described in this paper, mainly focusing on single-node architectures.

## 5. THE DESIGN ENVIRONMENT

The capabilities of our design environment are:

1. To graphically describe the target application, node architecture, network configuration, and task-to-node mapping.
2. To change (reconfigure) the system model to explore alternate designs.
3. To automatically configure and execute simulators at different levels of granularity and obtain system-wide energy and latency estimates.
4. To automatically update high-level model parameters using low-level simulation statistics.
5. To graphically visualize simulation results and (manually) inspect and identify performance bottlenecks in the design.
6. To facilitate plug-and-play integration of desired performance models and automatic optimization tools.

Using the illustrative example described in the next subsection, we demonstrate the above.

### An Illustrative Example: Automatic Target Recognition

The block diagram of the Automatic Target Recognition (ATR) application is shown in Figure 2. Seven nodes form a sensor array that performs real-time target tracking. Each of the nodes is equipped with microphones to monitor acoustic signals from the environment. One of the nodes (the ‘cluster-head’) transmits a line-of-bearing (LOB) estimate of the target to the outside world. By gathering LOB estimates from multiple such clusters, the exact location of the target can be obtained. There are three main stages in this computation. First, the sample of each channel is Fourier transformed. A frequency domain beamforming (BF) algorithm estimates target bearing along twelve different directions (beams), and delay-sum BF delays each signal by a time specified by the microphone array geometry for coincidence on a given beam. Once the signal energy in each of the twelve directions is obtained, the LOB estimator finds the direction with the largest signal energy and transmits it to the end user.

There are a number of design parameters that affect the time and energy performance of this system. For a given

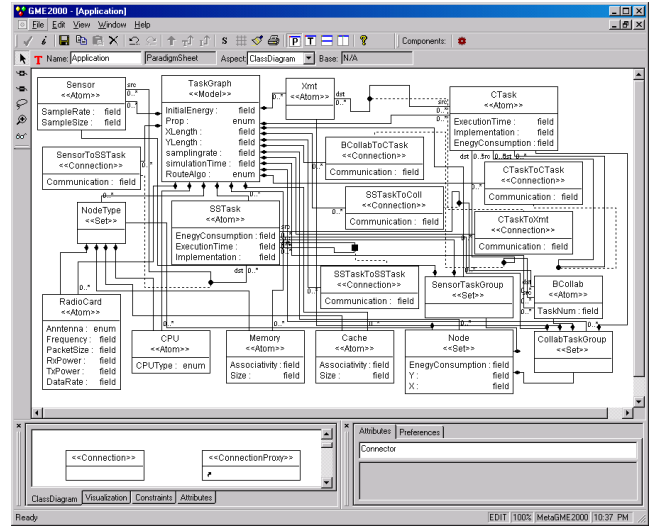


Figure 3: MetaModel

throughput requirement, mapping the FFTs onto individual sensor nodes and BF and LOB to the cluster-head is an energy-efficient solution, compared to a design where the sensor nodes transmit raw data and all computation is performed at the cluster-head [21]. The designer might also want to experiment with architecture parameters at the node level, such as cache configurations, voltage/frequency settings, transmit and receive power of the radio, etc. Network issues such as geometry and the propagation model will impact the communication cost in terms of both time and energy. We demonstrate how our design environment described below can be used to model and simulate this ATR sensor network, and analyze the effect of one or more of these parameters on the overall system performance.

### User Interface

We use the Generic Modeling Environment (GME2000) [8] to synthesize a domain-specific modeling interface for the system model defined in the previous section. The designer uses this interface to describe a specific system in the domain. In this case, the domain is a class of sensor networks. The set of building blocks and composition rules that are provided by the modeling interface to the designer is called a ‘modeling paradigm’. GME provides a graphical meta-modeling language to formally define the modeling paradigm, and automatically synthesizes a modeling environment based on the paradigm.

Figure 3 shows the meta-model used to design a modeling and simulation environment. We omit details of the meta-modeling due to space limitations. Briefly, the designer can draw task graphs for sensor-specific processing and collaborative processing, specify sample size and frequency, amount of data transferred between tasks, source code for the task<sup>3</sup>, etc. From the resource modeling perspective, the designer can specify the architecture of individual sensor nodes, i.e., the cache configuration, processor frequency, radio parameters, memory size, etc. Also, each sensor node is to be

<sup>3</sup>Since we have integrated the Wattach simulator for low-level power/performance analysis of a single sensor node, the source code should be in C.

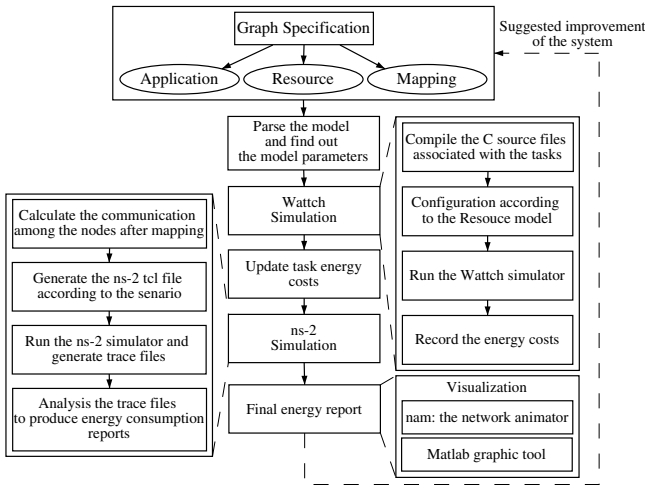


Figure 4: Simulator Integration

labeled with 2-dimensional coordinates to model the geometry of the network. The designer explicitly specifies the partitioning of tasks among the sensor node and cluster-head. Our design environment provides a set notation where the user can group the tasks into different sets and associate them with the sensor nodes described in the resource model.

### Simulator Integration

Once the desired system configuration is specified, its performance can be analyzed using the integrated simulation tools. Our framework supports plug-and-play integration of various simulators, made possible through the use of *model interpreters*, which access model information from the central database, and configure and execute the integrated simulators. Two aspects of such integration need to be emphasized. First, all the information that is needed by the simulation tool(s) should be captured through the modeling paradigm. If a new simulator to be integrated models some hardware entities or architecture features that are not captured by the existing modeling paradigm, the modeling paradigm needs to be changed, and a new environment synthesized. Second, writing a model interpreter requires an in-depth knowledge of the simulator being integrated. Once the integration is complete, however, the users of the framework need not know the details of the tool in order to use it.

We integrated the Wattach simulator for node-level power/performance analysis, and the ns-2 network simulator for the network level. Wattach is based on the SimpleScalar simulator, which models a RISC architecture, and has no knowledge of the network. Similarly, ns-2 models only the network and not the details of the nodes that transmit data over the network. For automatic synthesis of sensor network-based systems, system-wide power/performance analysis is a necessity. In the absence of a system-wide low-level simulator, we use node-level simulation results to automatically generate data transfer scenarios for the network simulation. After analyzing the results, the designer has the option of (manually) tuning parameters such as the task-to-node mapping, node architecture, routing protocol for the network, radio characteristics, etc.

Figure 4 shows the sequence of steps that is automati-

cally executed by the environment when the user chooses to simulate a particular system configuration. The input task graph and task-to-node mapping information is parsed. The user is expected to associate with each task, the C source code that defines its execution. The architecture parameters for the different nodes are obtained from the resource model. Using this information, the source code is compiled and power and latency information is obtained by invoking Wattach. Although the graphical interface (GME2000) is Windows-based, our model interpreters can automatically invoke the Wattach and ns-2 simulators on remote machines with different operating systems.

As mentioned in Section 4.2, we assume that the entire task graph mapped onto a given sensor node executes  $F_t$  times per second. Using latency information from Wattach, we calculate the maximum sampling frequency that can be supported by the user-specified design. If the maximum frequency is greater than the sampling rate of the sensor, the user is notified that the system is over-designed. A notification is also made if the system is too slow to keep up with the sampling rate of the sensor. In either case, the designer has the option of continuing with the network simulation using the lower of the two frequencies as the frequency of data transmission between the sensor node and cluster-head.

For the ns-2 simulation, parameters such as the routing protocol and radio specifications are extracted from the resource model. The size of data transmitted between nodes is obtained from the task graph and the frequency of transmission is determined as described in the previous paragraph. In the case when communication time for a block of data exceeds the computation time for the task graph, the communication latency determines the frequency of transmission. The nam [7] network animator is also configured automatically, and the designer can visualize data transfer in the network using the nam graphical interface. The complete design flow is shown in Figure 5. The different steps indicated in the figure are application modeling, resource modeling, node-level simulation, network simulation, and (manual) reconfiguration of the system after analyzing simulation results.

The implementation described in this section is of a framework and not restricted to any specific set of simulators and/or models. Wattach was integrated for the purposes of demonstrating the feasibility and usefulness of employing different simulators in a coordinated fashion to obtain system-wide power/performance analysis through a single system specification. Depending on the particular system instance, Wattach might be an inappropriate choice of simulator. Even in such a case, because there is a well-defined process for integrating any new simulator, all features of our framework can still be exploited.

### Automatic Model Refinement

A variety of high-level performance models for sensor networks exist in literature. [1] uses a node energy behavior model assuming that the node is purely a sensor or a relay, which means that computation at the node is not modeled. Average energy for sensing, transmission, and reception is used to approximate energy costs. [21] focuses on system partitioning to reduce energy consumption, and uses a simple energy model for computation based on the number of clock cycles, capacitance being switched per clock cycle, and operating voltage of the processor.

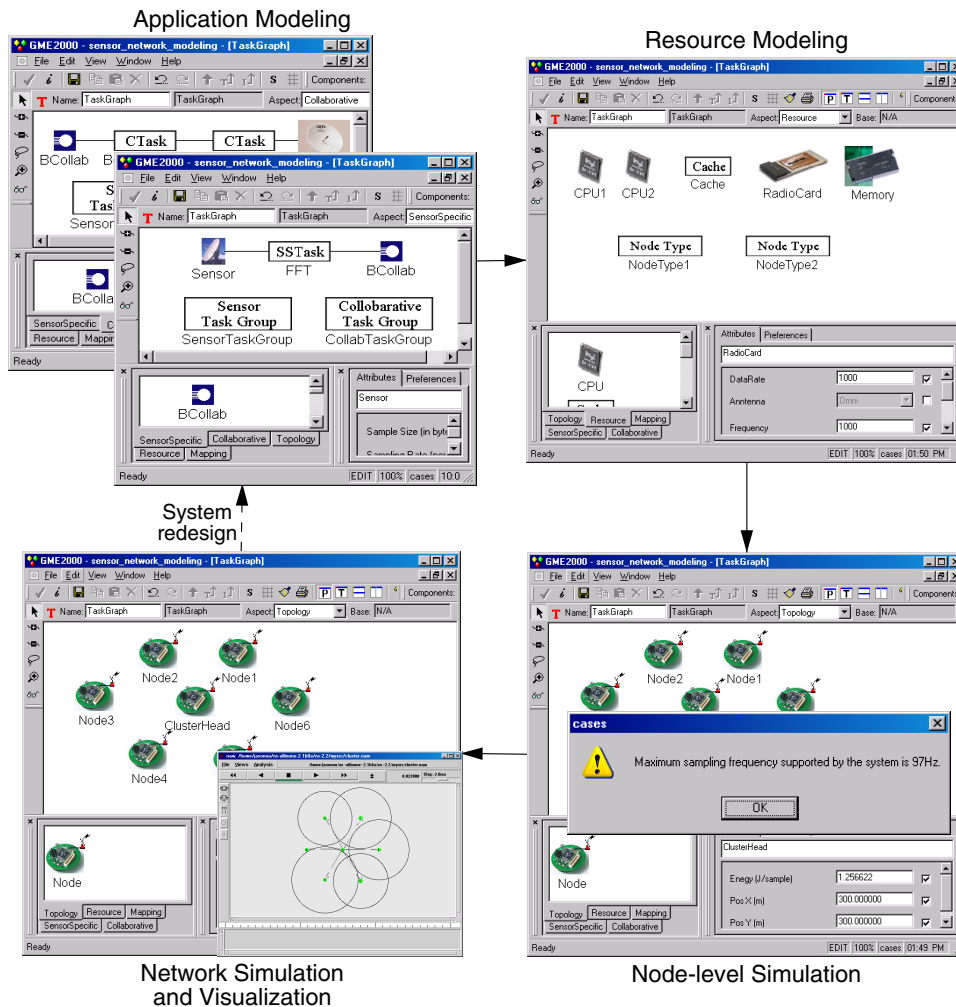


Figure 5: Design Flow

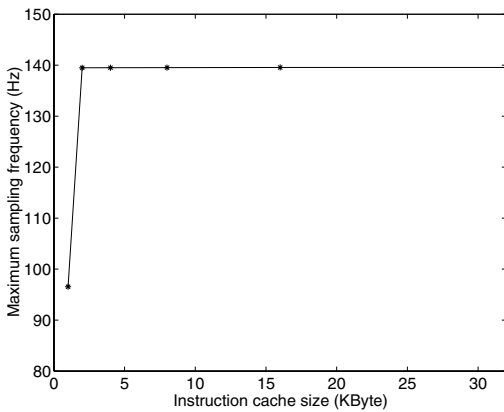
Unfortunately, communication costs and tradeoffs due to various effects in the network have traditionally been analyzed using simulation, perhaps because a qualitative understanding of the network behavior at the high level does not always provide a quantitative cost estimate for a specific scenario. For example, the efficiency of energy saving schemes proposed in [22] is evaluated empirically by simulating different network scenarios and not through analytical means using formal cost models. Especially for effects such as node placement, network density, etc., making high-level approximations that are also accurate is challenging.

Our framework offers a limited way to automatically update high-level model parameters using low-level simulation results. The usefulness of such refinement will naturally depend on the model used and the parameter(s) to be updated. The energy model in Section 4.2 needs the following parameters: (i) average energy per computation  $e_i$ , (ii) average cycles per computation  $l_i$ , (iii) data retransmission overhead  $\epsilon$ , and (iv) data encoding overhead  $\delta$ . Since high-level estimation usually precedes low-level simulation, initial values of these parameters will be provided by the user from sources such as data sheets, prior experimental results, and understanding of the application. For example,  $e_i$  can be

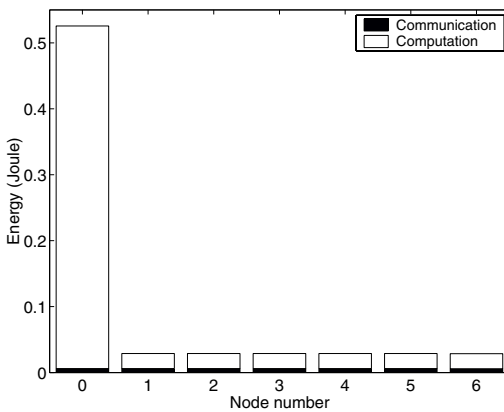
approximated using a table of energy consumed for each operation in the processor’s instruction set, and a rough estimate of the relative percentages of significant computations in the given task. The value can be refined when low-level simulation provides the exact amount of energy consumed. A similar refinement can be made for  $l_i$ .  $\delta$  can be quite accurately specified even initially, based on a knowledge of the encoding scheme used. For instance, if 1/3 rate Forward Error Correction (FEC) is used,  $\delta = 3$ .  $\epsilon$  is the most difficult of the approximations to characterize, because it depends on factors such as the MAC protocol used, network density, geometry of the sensor network, etc. Our framework adopts the simple approach of maintaining  $\epsilon$  as the cumulative average of  $\epsilon$  values obtained during all the ns-2 simulations for a particular task graph and resource model.

### Sample Results: Analysis

Our tool provides a convenient mechanism for the designer to specify and manipulate system configurations, and empirically analyze system-wide performance. For the ATR application that was implemented as a case study, two of the experimental results we obtained using the design environment are shown in Figures 6 and 7.



**Figure 6: Effect of instruction cache size on maximum sampling frequency supported by the node**



**Figure 7: Ratio of communication and computation energy for each node**

Figure 6 shows the effect of instruction cache size on the maximum sampling frequency supported by that node, i.e., the latency of execution of the task graph mapped onto that node. It can be observed that Wattch simulation shows that increasing the size of the instruction cache after 16KB has no effect on system throughput. The designer can conclude (based on his/her level of confidence in the simulator) that an instruction cache greater than 16K will only lead to wasteful energy consumption without improving throughput performance.

Figure 7 is a visualization of the system-wide energy consumption statistics output by our environment after a combined ns-2/Wattch simulation run. Node 0 is the cluster-head, which is also apparent by inspection of the results. For all the nodes, the simulation shows that computation energy is greater than communication energy, *which is usually not the case* with current sensor node architectures where transmitting a byte of data consumes an order of magnitude more energy than performing one computation. The result can be partly explained by the fact that Wattch models a relatively heavy-duty out-of-order superscalar processor that is not optimized for energy. The Wattch simulator was integrated as a proof of concept, and using a simulator for a low-power processor such as the Intel StrongARM will yield a different result. Ultimately, the choice of simulator is left to the user,

and the performance estimates provided by our simulation framework will only be as accurate as the simulators used. No guarantees about the accuracy of the simulation results are provided by the framework per se.

## 6. RELATED WORK

No other design environment to our knowledge provides the capabilities of graphical specification, multigranularity simulation, automatic co-ordinated low-level simulation, and automatic model refinement, at both node level and network level. The IMPACCT [6] tool also requires the designer to specify the target architecture and mapping. It allows simulation at various stages of the design flow, and derives power-aware computation schedules for tasks on a single node. However, IMPACCT is meant for exploring power management architectures and policies at the component and system levels of a single node architecture; and does not model networked systems in the level of detail that is facilitated by our framework. Automatic model refinement is also not featured in the IMPACCT tool.

In [17], Co-design Finite State Machines (CFSMs), a model of computation, was introduced to allow the efficient capture of both the control and the data processing parts of the specification. Also, the design of a single-hop Intercom network, was discussed in detail as an application example. However, the paper focused primarily on communication protocol designs and not on the design of node architecture and energy cost of computation. In [11], a parallel simulation environment for mobile wireless networks was proposed. The focus was on simulation of large radio networks and involved detailed modeling of radio propagation, channel access schemes and interference patterns. The node architecture was not modeled in detail. Also, multi-granularity simulation was not supported. In [18], various component-based designs, methodologies for hardware synthesis from programming language descriptions, and energy-efficient system-level designs were proposed. However, there was no unified framework such as the one implemented in this paper that allows a designer to explore the system-wide design space at different levels of simulation granularity, and supports plug-and-play integration of simulation, visualization, and optimization tools.

## 7. CONCLUDING REMARKS

With advances in technology and innovative new uses of sensor networks, many more challenges will emerge for automatic synthesis than are currently being addressed. New system models, and performance analysis and optimization techniques will be devised, tailored to the particular design problem at hand. One of the contributions of this paper is a performance model for a class of sensor networks, and a framework for refining the model parameters based on low-level simulations. We proposed a specific methodology for automatic synthesis based on multi-granularity simulation, simulator integration, and model refinement. We also described a tool that has been implemented to support this approach. Specific optimizations based on the model were not suggested, partly because the approach is independent of the precise parameters being optimized and techniques for optimizing them. One of the main limitations of the system model in this paper is that self-(re)organization is not supported and that node functionality is fixed a priori.

We are in the process of defining a new high-level model for self-organization at both intra-node and inter-node level, focusing on communication and computation costs of such self-organization. Also, we are addressing the specific problem of synthesizing efficient communication schedules for wireless transmissions for such (multi-hop) sensor networks. The modeling paradigms, simulators, and system model described in this paper will eventually be integrated into the MILAN [4] framework.

## 8. REFERENCES

- [1] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, "Upper bounds on the lifetime of sensor networks," Proc. ICC 2001, June 2001.
- [2] A. Bakshi, "Automatic synthesis of application-specific sensor networks," PhD thesis (in preparation), University of Southern California.
- [3] A. Bakshi, J. Ou, and V. K. Prasanna, "Power-aware embedded system design using the MILAN framework," IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT), May 2002.
- [4] A. Bakshi, V. K. Prasanna, A. Ledeczi, V. Mathur, S. Mohanty, C. S. Raghavendra, M. Singh, A. Agrawal, J. Davis, B. Eames, S. Neema, and G. Nordstrom, "MILAN: A model based integrated simulation framework for design of embedded systems," ACM SIGPLAN 2001 Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES), June 2001.
- [5] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," 27th Intl. Symposium on Comp. Arch. (ISCA'00), June 2000.
- [6] P. H. Chou, J. Liu, D. Li, and N. Bagherzadeh, "IMPACCT: Methodology and tools for power-aware embedded systems," to appear in Design Automation for Embedded Systems, Special Issue on Design Methodologies and Tools for Real-Time Embedded Systems.
- [7] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, and H. Yu, "Network visualization with the VINT network animator nam," Technical Report 99-703b, University of Southern California, Mar. 1999.
- [8] Generic Modeling Environment, <http://www.isis.vanderbilt.edu/>.
- [9] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan, "Effects of detail in wireless network simulation," In Proceedings of the SCS Multiconference on Distributed Simulation, pp. 3-11, January 2001.
- [10] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," Proc. of the Hawaii Intl. Conf. on System Sciences, Jan. 2000.
- [11] W. Liu, C.C. Chiang, V. Jha, M. Gerla, R. Bagrodia, "Parallel simulation environment for mobile wireless networks," Proceedings of the 1996 Winter Simulation Conference, 1996.
- [12] MILAN: a model-based integrated simulation framework, <http://milan.usc.edu/>.
- [13] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis, "Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation," Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES), June 2002.
- [14] Network Simulator - ns-2, <http://www.isi.edu/nsnam/ns>.
- [15] M. Singh, and V. K. Prasanna, "System level energy trade-offs for collaborative computation in wireless networks," IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT), May 2002.
- [16] S. Singh and C.S. Raghavendra, "PAMAS - Power-aware multi-access protocol with signaling for ad hoc networks", ACM Communication Review, July 1998.
- [17] M. Sgroi, J. L. da Silva Jr., F. De Bernardinis, F. Burghardt, A. L. Sangiovanni-Vincentelli, and J. M. Rabaey, "Designing wireless protocols: methodology and applications," Proc. ICASSP, June 2000.
- [18] System-Level Design Tools and Hardware/Software Co-design, <http://akebono.stanford.edu/users/nanni/research/sys/>.
- [19] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," ACM Mobile Computing and Communication Review (forthcoming).
- [20] R. E. Van Dyck and L. E. Miller, "Distributed sensor processing over an ad-hoc wireless network simulation framework and performance criteria," Proc. of IEEE Milcom, Oct. 2001.
- [21] A. Wang and A. Chandrakasan, "Energy efficient system partitioning for distributed wireless sensor networks," ICASSP 2001, May 2001.
- [22] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," IEEE INFOCOM 2002.