A Routing-Algorithm-Aware Design Tool for Indoor Wireless Sensor Networks

A. Puggelli, M. Mozumdar, A. L. Sangiovanni-Vincentelli
Department of Electrical Engineering and Computer Science
University of California - Berkeley
Berkeley, CA - USA
{puggelli,mozumdar,alberto}@eecs.berkeley.edu

L. Lavagno
Department of Electronics
Politecnico di Torino
Torino, Italy
lavagno@polito.it

***Abstract*—We present a design tool to assist the rapid prototyping and deployment of wireless sensor networks for building automation systems. We argue that it is possible to design networks that are more resilient to failures and have longer lifetime, if the behavior of routing algorithms is taken into account at design time. Resiliency can be increased by algorithmically adding redundancy to the network at locations where it can be maximally leveraged by routing algorithms during operation. Lifetime can be increased by placing routers where they are most needed according to the expected data traffic patterns, to improve the quality of the transmission. The network synthesis problem is formulated as an optimization problem: we propose a mixed-integer linear program to solve it exactly, and a polynomial-time heuristic that returns close-to-optimal results in a shorter time.**

***Index Terms*—resiliency; power consumption; routing algorithms; sensor network; graphical user interface.**

I. Introduction

In the last decade, Wireless Sensor Network (WSN) applications have been extending rapidly in many fields such as factory automation, environmental monitoring and security systems. Recently, efforts have been made to enable a large scale deployment of WSN technology also in the field of Building Automation Systems (BAS). Applications in this domain range from health-care monitoring to home automation, and, even more importantly, to the automation of power management. Recent studies show that building operations (such as lighting, Heating, Ventilation and Air Conditioning (HVAC)) represent around 40% of the total energy consumption in the United States [1]. It is widely believed that controlling these operations effectively can reduce energy consumption from 30% up to 70%. Wireless technology is highly promising, since its deployment costs are substantially lower than the ones associated with a wired solution. Moreover, the large number of existing facilities that could be the target of WSN-based systems would guarantee high returns on investment. On the down-side, the complexity of the design of WSNs will most likely require multi-disciplinary teams to be involved in specifying, designing, implementing, and maintaining WSN solutions. These teams could involve architects as well as civil, electronics and telecommunication engineers, all with a common need to share a unified representation of the WSN node placement to optimize sensing, actuation and communication, only to mention a few concerns.

In previous papers, we illustrated tools and methodologies for the modeling, simulation and automatic code generation of WSN applications [2]. Here, we extend our results by proposing a tool for network synthesis. In particular, the first contribution of the paper is the introduction of a tool that optimizes network topology, i.e. the location of network nodes, so that its resiliency to failures and lifetime are maximized. The tool facilitates users by reducing design time and by improving the quality of the network topology with respect to a simulation-based approach, where designers have to simulate several different topologies and select the most performing one, with no guarantee of optimality. Users can interact with the tool through a GUI, and add new information about the network behavior and the deployment environment, according to their fields of expertise. The tool takes this information into account to incrementally adapt the node positions, and it provides feedback to the designers by analyzing network performance. Network simulators (e.g. NS-2 [3]) may then be used on the optimized topology to verify functionality (e.g. packet scheduling), and to finely tune the network behavior at lower levels of abstraction.

The problem of network synthesis has already been addressed in the past. Contrarily to [4], which considers networks only made of sensors, we consider heterogeneous networks, made of end-devices (sensors and actuators) and routers. We assume that end-device locations are predefined and fixed, since in BAS applications end-device density is often standardized (e.g. fire alarm sensors), and full sensing coverage is usually not required (e.g. HVAC systems) [5]. Our goal is to determine optimal locations for the routers.

In [6, 7], the authors present design tools for the automated synthesis of WSNs that satisfy connectivity and Quality of Service (QoS) constraints. The very general synthesis algorithm presented in [6] is based on a Mixed-Integer Linear Program (MILP). A possible strategy to make the MILP approach scalable is to decompose the synthesis problem into an optimal number of local subproblems [7]. The obtained results can be close to the globally optimal solution (albeit it is not possible to guarantee it or to give a tight bound of the distance to the optimal solution) because most BAS networks indeed have a structure with mostly local interconnections. Our framework treats QoS as a set of constraints for the synthesis problem, and it implements polynomial time heuristics to find a locally optimal solution. We differentiate from previous work, because our proposed algorithms can synthesize network structures that are much more general than the ones analyzed in [7], and in a much shorter time with respect to [6]. Moreover, we optimize the synthesized network with the specific goal of increasing its resiliency to faults and reducing its power consumption in order to extend battery life.

Network resiliency is a fundamental property, both to increase the effectiveness of the provided service and to lower maintenance costs. In [8], network lifetime is extended by maximizing the time before the first device exhausts its battery. On the other hand, resiliency depends not only on device lifetime but also on other factors, such as node failures and the quality of the transmission links. Since it is very difficult to thoroughly account for these factors at design time, network resiliency can be increased by adding redundancy to it [9]. From our perspective, we are interested in synthesizing networks with redundant paths, along which packets can be routed when the main path becomes faulty, at a minimal penalty in terms of extra dissipated power.

The authors of [9] propose a set of polynomial time algorithms for the synthesis of robust networks. While these algorithms select redundant paths only based on connectivity, we propose to synthesize redundant paths based on the predicted behavior of the Routing Algorithms (RAs) that operate in the WSN. RAs route packets based not only on connectivity but also on the data traffic patterns, and they rank paths according to metrics across the OSI layers. In particular, the second contribution of the paper is the introduction of network-synthesis algorithms that allow designers to model most traffic patterns that are commonly supported by WSNs (e.g. unicast, multicast, Peer-to-Peer (P2P), mobile nodes) [5]: the algorithms place routers along the shortest paths from sources to destinations, by ranking paths according to the same metrics used in WSN-oriented RAs [10]. Since wireless transmission is the major source of power consumption in a WSN [11], a synthesis flow based on the emulation of the behavior of RAs also reduces the network power consumption: it minimizes the total number of hops of the wireless transmission, and it increases the link quality along the paths, so that fewer transmissions (and re-transmissions) are needed. Moreover, our algorithms take QoS constraints into account, and we propose heuristics whose complexity is lower than the one reported in [9].

The rest of the paper is organized as follows. In Section II, we give some background on WSN-oriented routing algorithms; Section III describes the proposed tool and how it supports the design flow of a WSN; in Section IV, we show details on how to formulate the synthesis problem, and we propose algorithms to solve it; some final conclusions are drawn in Section V.

II. Background

The large variety of BAS and the severe constraints on power consumption suggest the use of heterogeneous traffic patterns to route packets, so that each application can choose the one that results in the best performance [5]. The basic traffic pattern to be supported is gateway/end-device unicast, since each device needs to communicate with the gateway during its lifetime. Multicast allows a packet to be transmitted only once, while reaching several destinations, thanks to the shared nature of the wireless link. P2P communication is a specialization of unicast where two end-devices are connected directly without routing through the gateway. Finally, RAs should also support mobile devices (e.g. remote controllers).

Every RA ranks possible paths from source to destination according to some predefined cost function. RAs for WSNs should contemporaneously minimize the number of hops from source to destination, at the network layer of the OSI model, and maximize the quality of the links along the path, at the MAC and PHY layer [10]. In the following, the link quality is evaluated in terms of the estimated Propagation Loss (PL) between two devices: even it this metric is subject to large variations in real scenarios, it is widely used in RAs to rank paths because it can be easily computed on the device (e.g. using the Received Signal Strength Indicator (RSSI), and knowing the transmitted power) [12].

III. Design Flow

In this section, we present the developed Graphical User Interface (GUI), developed using the Matlab GUI Development Environment [13]. To show how the tool can assist engineers during the design of a WSN for building applications, we study a concrete example, the design of a WSN for the Donald O. Pederson Center at Cory Hall, Berkeley. The network needs to fulfill two tasks: 1) report temperature measurements from each room of the Center to a base station for monitoring purposes; 2) send actuating commands from a central panel to the lights located in the two meeting rooms. Figure 1 shows the different phases of the network design, commented in the following paragraphs.

Application Development. The application engineer is concerned with placing sensors and actuators where they are needed, and with defining the traffic patterns that regulate the flow of data among the nodes. The tool allows the upload of a 2D floor plan of the environment, where end-devices (ovals) and routers (rectangles) can be placed simply by clicking on the floor plan area. Entered nodes are indexed with an increasing number. In our example, we place: 1) one temperature sensor in each room, and a base station (represented as a router) in the Machine Room, for the temperature monitoring application, and; 2) for each meeting room, one sensor to represent the control panels, and one actuator for each light. Temperature sensors communicate to the base station via unicast, which is the default traffic pattern. Packets from the control panel to the light actuators are sent via either P2P or multicast traffic, depending whether the corresponding command activates one or multiple lights. We thus assign these nodes to both communication patterns: P2P communication can be set by entering the indices of the source-destination pair nodes, while a set of end-devices that communicate via multicast can be graphically selected by highlighting the floor plan area surrounding them. In general, nodes can be assigned to more than one traffic pattern. Even if not present in the example, also mobile end-devices can be taken into consideration in our tool, by selecting the area on the floor plan in which they can be moved. The result of our placement is shown in Figure 1(a): overall, we placed 50 devices. The tool is now able to synthesize a tentative layout of the network with the desired level of redundancy and QoS, based on the information entered up to this point. At this step, errors are introduced because the tool models the quality of the wireless link with Free-Space (FS) and Multi-Wall (MW) propagation models [14], and it assigns a default value of bit rate to nodes. Nevertheless, the topology shown in Figure 1(b) represents a good starting point for the subsequent refinement steps, which will require more information from the designer. The tool added 39 routers to meet the specifications (for the sake of readability, we only show the floor plan section of the GUI).

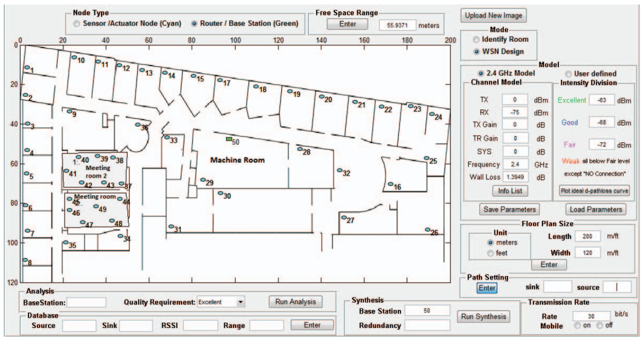
Network Analysis. The communication engineer can refine the design of the network by adding information that guides the synthesis flow towards a more accurate result.

First, the actual bit rate for each path can be added (including header sizes down to the MAC layer, if this information is available) to properly account for power dissipation in the network. In our example, temperature sensors send data to the base station every 5 minutes, so they transmit more packets than the control panel, which is usually activated only for short periods of time during business hours. A new network synthesis can be run after adding this information. Based on the result of the previous step, the synthesis algorithm first tries to incrementally reroute only those paths whose bit rate has increased: in this way, the optimized network is only perturbed where it is needed, and results are produced in a short time; if the incremental step does not work, all paths are rerouted to obtain a valid network.

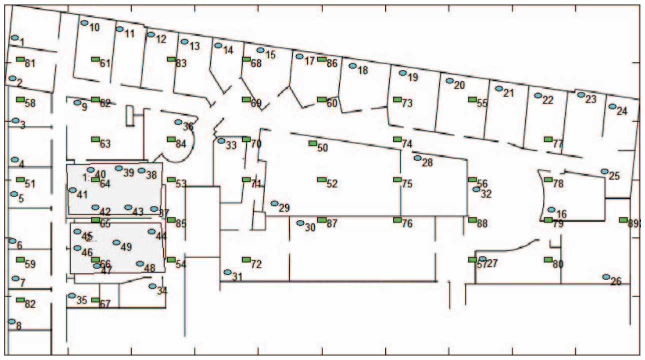
Second, all valid paths are processed to measure the power consumption of the network devices. The results of the analysis are shown graphically by changing the color of the nodes according to a color scale (e.g. red for nodes with high power consumption). The designer can mark some routers to be main-power supplied (i.e. the algorithm disables the power check for them), duplicate some routers to achieve a better power balancing across the network, and change the location of some routers: user-entered routers are marked to be the preferred choices to route paths in the subsequent steps of synthesis.

Figure 1(c) shows the updated network in our example: we marked the added routers with stars, and the removed routers with crosses. The updated bit rates caused one router to be

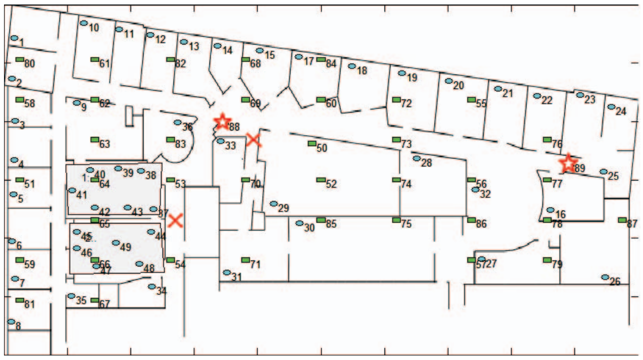
(a) Initial placement of end-devices and base station.



(b) Synthesized topology at the end of the *Application Development* phase.



(c) Updated topology at the end of the *Network Analysis* phase.



(d) Optimized network layout at the end of the design cycle.

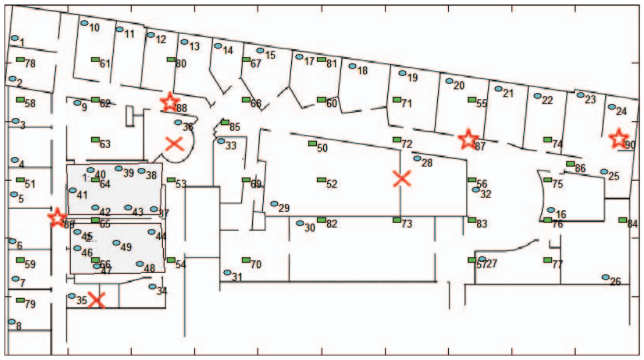


Fig. 1. The figure shows how the topology of the network gets refined at each design step.

removed, and one router to be added at a different location. Moreover, we moved one of the routers, since it had been placed in a non-suitable location (wall monitors are installed there).

Site Survey. A site survey is usually required to correctly evaluate the characteristics of the network working environment. Our framework gives the capability of integrating data collected during the site-survey, and to adjust the design of the WSN, thus combining at synthesis time the flexibility of propagation models to the accuracy of measurements [15].

At the network level, the field engineer can input in the GUI accurate values for the parameters of the FS and MW propagation models, determined through measurements. At the single link level, the GUI can store measured values of PL into a database. The database becomes important because it is difficult to fit the model parameters so that all the PL estimations are correct, due to the heterogeneity of the environment [15]. More accurate models (e.g. [17]) and a better environment description might result in better predictions, at the cost of increased computational and field data collection complexity. We instead opted for using simple models in the first steps of synthesis, and to refine the design when on-field measurements are available. First, the PL for each link synthesized in the previous steps should be measured and stored in the database. Second, the synthesis is run again, and the tool adjusts the network topology, by taking the new information into account. A few measurement iterations might be needed if the algorithm routes paths through different routers with respect to the previous step, since the quality of new links might need to be assessed. However, it will be shown in Section IV that a number of measurements only linear in the network size need to be taken, so data collection is simplified, and the database can be efficiently processed.

Figure 1(d) shows the final layout of the network, obtained after two iterations of measurements. The algorithm added one more router (for a total of 40 routers), and it changed the location of three routers after taking into consideration the accurate measurement results of the link quality.

IV. Network Synthesis

We cast the synthesis problem for resilient and power efficient WSNs into an optimization problem, formally defined as follows:

Problem Statement. **Given**: 1) a set of end-devices D and a set of fixed routers R, with their locations; 2) a set of source-destination pairs $Q = \{q{=}(s,d) \mid s,d \in D\}$ with the associated bit rate r^q, where Q is partitioned in $Q = Q_{uni} \cup Q_{multi} \cup Q_{mob} \cup Q_{p2p}$ to differentiate among traffic patterns, and; 3) a desired number m of redundant replicas $\forall q \in Q$. **Compute** the set AR of additional routers and corresponding locations that minimizes network power consumption subject to guaranteeing the connectivity and QoS of m redundant paths $\forall q \in Q$.

In our implementation, the set Q is partitioned manually during the *Application Development* phase, as described in Section III.

In this section, we propose two algorithms to solve the above optimization problem. Both algorithms initially populate the floor plan with a set VR of virtual routers, i.e. potential locations for routers to be added to the network. In our implementation, VRs are uniformly distributed over the floor plan at discrete locations on a grid. Indeed, most non-pathological networks can be synthesized if $W = m \cdot \left(\frac{A}{A_c}\right)$ virtual routers are placed with this pattern, where A is the total area of the facility, and A_c is an estimate of the router connectivity area. The synthesis algorithms then select the set $AR \subseteq VR$ to optimize for power consumption, while satisfying all constraints.

The algorithms differentiate from one another because they trade-off the optimality of the solution with running time. In Section IV-A, we formulate the synthesis problem in terms of a MILP, which returns the globally optimal network topology. On the other hand, it is known that algorithms for the solution of MILPs are not polynomially bounded in running time, so solving them is not in general computationally efficient: high running time has been reported also for the synthesis of small networks (~ 30 end-devices) [6]. During the network design cycle (e.g. the *Site Survey* phase), a faster response time from the tool could be desired because new data may be available incrementally, and to try multiple different solutions (e.g. different communication protocols, which result in different bit rates). To address this problem, we propose in Section IV-B a polynomial-time heuristic that synthesizes the network in a shorter time, at the expense of returning a (possibly) sub-optimal solution.

The user can select the synthesis algorithm that is most suitable for the ongoing design stage. In the example presented in Section III, we first run the MILP-based synthesis to get a good starting point for the design (Figure 1(b)); we then run fast heuristic-based syntheses to locally tune the topology while adding information (Figure 1(c)). We concluded the design by running again a MILP-based synthesis to further improve performance (Figure 1(d)).

A. MILP-based Synthesis

The MILP representation is based on the one proposed in [6], but we modify it to model the power consumption of data traffic patterns, and to add redundancy to the network. A preprocessing step computes the connectivity matrix C of the network: nodes represent devices and the presence of an edge between two nodes is established based on the FS and the MW propagation models. The algorithm then enriches C with a set of virtual routers VRs, positioned on an equally-spaced grid. Each $vr \in VR$ is assigned a Boolean variable x_i, whose value represents whether the router is installed or not in the synthesized network. The network is now formed by nodes $n \in N = D \cup R \cup VR$. Each edge of C is assigned $m \cdot |Q|$ Boolean variables $y_{ij}^{q,k}$ for $k = 1$ to m, $\forall q \in Q$: $y_{ij}^{q,k}$ is true if the edge (n_i, n_j) is along the k^{th} replica of path $q \in Q$. Finally, for all variables y, we assign a variable $r_{ij}^{q,k}$, that models the bit rate of the transmission through the link (n_i, n_j) along the k^{th} replica of path q. A path $(s, d) \in Q$ is connected if there exist a solution to the equation $C\mathbf{y} = \mathbf{b}$, where $\mathbf{b}[s] = -1, \mathbf{b}[d] = 1, \mathbf{b}[j \neq s, d] = 0$.

$$\min_{x,y} \quad \mathrm{P} = \alpha \sum_i (p_i \cdot x_i) + \beta \sum_{q,k} \sum_{i,j} \left(y_{ij}^{q,k} \cdot r_{ij}^{q,k} \cdot \left(e_{ij}^{RX} + e_{ij}^{TX}\right)\right)$$

$$\begin{array}{lll}
\text{s.t.} & \textit{(Topological)} & \\
1) & C\mathbf{y_{q,k}} = \mathbf{b_q}, & \forall q \in Q, \forall k \\
2) & \sum_{k=1}^{m} \left(y_{ij}^{q,k}\right) - 1 \leq 0, & \forall i,j \in C,\ \forall q \in Q \\
3) & x_i + x_j - 2y_{ij}^{q,k} \geq 0, & \forall i,j \in C, \forall q \in Q, \forall k \\
 & \textit{(Power Accounting)} & \\
4) & r_{ij}^{q,k} = r^q, & \forall i,j \in C, \forall q \in Q \setminus Q_{multi}, \forall k \\
5) & \sum_{q \in Q_{multi}} r_{ij}^{q,k} = r^q, & \forall i,j \in C, \forall q \in Q_{multi}, \forall k \\
 & \textit{(QoS)} & \\
6) & \sum_{q,k} y_{ij}^{q,k} \cdot r_{ij}^{q,k} \leq BW_M, & \forall i,j \in C \\
7) & \sum_i e_{ij} \leq OUT_M, & \forall j \in C \\
8) & \sum_{ij} y_{ij}^{q,k} \cdot l_{ij} \leq L_M^q, & \forall q \in Q, \forall k \\
9) & \sum_{ij} y_{ij}^{q,k} \cdot \log(1 - b_{ij}) \leq \log(1 - BER_M^q), & \forall q \in Q, \forall k \\
10) & p_j + \sum_{q,k} \sum_i y_{ij}^{q,k} \cdot r_{ij}^{q,k} \cdot e_{ij}^{RX} \dots & \\
 & + \sum_{q,k} \sum_i y_{ji}^{q,k} \cdot r_{ij}^{q,k} \cdot e_{ij}^{TX} \leq PC_M, & \forall vr_j \in VR \\
11) & x_i,\ e_{ij},\ y_{ij}^{q,k} \in [0,1] & \forall i,j \in C, \forall q \in Q_{multi}, \forall k \\
12) & r_{ij}^{q,k} \geq 0 & \forall i,j \in C, \forall q \in Q_{multi}, \forall k
\end{array}$$

The *Topological* constraints enforce that m replicas $\forall q \in Q$ are connected (1); that the m replicas are all disjoint (2) (an edge can be picked at most once, when routing the m replicas of path $q \in Q$); and that routers are installed, if they are used (3). *Topological* constraints route all paths as if they were unicast paths. We add *Power Accounting* constraints to correctly differentiate among data traffic patterns. In (4), unicast, P2P and mobile paths are assigned an input bit rate: for the mobile paths, this assignment corresponds to a worst case scenario. Constraint (5) enforces the bit rate of a link to be constant even though multiple paths belonging to the same multicast group are routed through it: this models the sharing of the wireless medium. In order to synthesize a working WSN, we also need to guarantee some level of QoS in the network. Constraint (6) limits the sum of the bit rates to be transmitted across a link to the link bandwidth; (7) limits the maximum fan-out of a node; $(8-9)$ limit the maximum latency and the maximum Bit Error Rate (BER) of a path, where b_{ij} is the BER across the edge (n_i, n_j). Finally, constraint (10) limits the maximum average power consumption of a node: p is the fixed power consumption (standby and processing) of the router; e_{ij}^{TX} and e_{ij}^{RX} are the energy consumed to transmit and receive a bit over the link (n_i, n_j), respectively (e^{TX} and e^{RX} depend on the link quality and they are computed $\forall i, j$ in a preprocessing step). This constraint can be interpreted as the willingness of a router to route packets, and it sets a lower bound on the device lifetime. The cost function is made of two components. The first one represents the fixed power consumption of the routers; the second one represents the total power dissipated in transmission. The two components of the cost function are weighted by constants α and β $(\alpha + \beta = 1)$, in order to explore different regions of the optimization space. While fixed power consumption increases linearly with the number of routers, this penalty might be balanced by savings in power consumed in transmission, because more routers connect the network more effectively. Finally, we note that minimizing for power also enables the correct assignment of multicast paths, since multicast transmission is more power efficient than the unicast counterpart (constraint 9).

The algorithm returns the set $AR = \{vr_i \in VR \mid x_i = 1\}$.

B. Heuristic-based Synthesis

In this section, we propose a polynomial time algorithm whose output result satisfies the same constraints enforced in the MILP. Moreover, the returned solution is close-to-optimal if the network has mostly local interconnections, as it commonly happens in BAS applications [7]. The connectivity matrix C allows us to represent the network as a graph: paths among nodes can now be computed using shortest path algorithms. In fact, RAs use shortest path algorithms to route packets: we emulate their behavior, as if they were to be run in a network populated also by VRs. Moreover, shortest paths minimize the number of hops and maximize the quality of the transmission, so less power is consumed in transmission. After all paths are routed, all the VRs that appear along at least one of the paths are collected in the set AR, and the resulting network satisfies all constraints.

Matrix C is sparse due to the limited connectivity range of wireless devices, so C has $O(|N|)$ non-zero entries. This confirms that only $O(|N|)$ measurements need to be taken during the *Site Survey* to characterize it, as argued in Section III. Edges are assigned a weight, in the range $[1-4]$, to represent their Link Quality (LQ) (a low value represents high LQ). The weights are computed by estimating the link path loss using FS and MW models. We then use a modified version of the Dijkstra algorithm to route paths. The cost function $C = f(\#H, LQ)$ ranks paths according to the number of hops (#H) to the destination, and the LQ of each hop, following the indications in Section II.

Algorithm 1 shows how paths are calculated in our implementation. As far as routing is concerned, unicast, P2P and mobile traffic patterns are treated in the same way (lines $4-10$). For mobile nodes, the area A_m in which they can be moved is divided into $s = \left(\frac{A_m}{A_c}\right)$ sections, and a path is routed from the center of each section to the destination. The algorithm processes one path at a time: first, it disconnects from C all edges entering end-devices (apart from the destination), Algorithm 1. Synthesis of Power-optimized WSNs

```
1: **Given** Sets of end-devices $D$, routers $R$, virtual routers $VR$
2: **Input** Connectivity matrix $C$, set $Q$ of pairs $(s, d)$, redundancy $m$
3: **Output** Set of synthesized paths $P$
4: //Process paths with unicast/P2P/mobile traffic pattern.
5: for $k = 1$ to $|Q_{uni}| + |Q_{p2p}| + |Q_{mob}|$ do
6:     $C_k \leftarrow$ disconnect_end_devices($C, q_k$)
7:     for $j = 1$ to $m$ do
8:         $\left[p_k^j, conn\right] \leftarrow$ Dijkstra($C_k, s_k, d_k$)
9:         $P \leftarrow P \cup p_k^j$
10:        $C_k \leftarrow$ disconnect_path$\left(C_k, p_k^{j-1}\right)$
11: //Process paths with multicast traffic pattern.
12: for $l = 1$ to $\#MG$ do
13:     for $j = 1$ to $m$ do
14:         $\left[P_{multi}^j, conn\right] \leftarrow$ Dijkstra($C, BS_l, MG_l$)
15:         $P_{multi}^j \leftarrow$ sort_paths$\left(P_{multi}^j\right)$
16:         for $k = 2$ to $|MG_l|$ do
17:             $C \leftarrow$ set_path_cost_to_0$\left(C, P_{multi}^j[k-1]\right)$
18:             $P_{multi}^j[k] \leftarrow$ Dijkstra($C, s_k, BS$)
19:         $P \leftarrow P \cup P_{multi}^j$
20:         $C \leftarrow$ disconnect_paths$\left(C, P_{multi}^j\right)$
21: $[BW, PC, OUT] \leftarrow$ path_accounting($P$)
22: $P \leftarrow$ reroute_shortest_paths($C, P, BW, PC, OUT$)
23: return($P$)
```

since no paths can be routed through them; second, it traverses the graph from source to destination. Since we aim at routing m independent replicas $\forall q \in Q$, at each iteration of the inner loop (line 7) the algorithm disconnects from the graph the path that has just been computed (line 10). The following iteration will thus find a path that is completely independent from the previous ones. The complexity of this part of the algorithm is $O\left(m \cdot |Q| \cdot |N| \cdot \log\left(|N|\right)\right)$.

For multicast traffic, we assume that devices are clustered in (possibly overlapping) Multicast Groups (MG), where a local Base Station (BS) sends packets to several other nodes. The algorithm presented above would not generate acceptable results when modeling multicast traffic, since it synthesizes a set of unicast paths from the BS to all the nodes of the MG, which results in a waste of power. Since in multicast several nodes can be reached with a single packet transmission, no more power is dissipated if we connect an end-node to a router that has already been selected. We thus aim at determining the smallest set of *VR*s that is capable of connecting the MG to the BS: since each router only transmits once, the overall power consumption is minimized. Lines $12-20$ in Algorithm 1 present an $O\left(\#MG \cdot m \cdot |N|^2 \cdot \log\left(|N|\right)\right)$ approach to achieve this goal. MGs are processed one at a time. P_{multi} is the set of paths from the BS to each node in the MG (line 14). At line 15, the elements of P_{multi} are sorted according to the cost to get to the BS. Paths are then processed in increasing order of cost: the path from the BS to the node with the least cost is taken, since that is the shortest path to get to the MG. The key point of the algorithm is that the cost of the first path can now be set to 0: if any other node chooses that path, no more power is consumed due to multicast propagation. The second least costly path of the set is then rerouted, and the newly obtained path replaces the old one in P_{multi}, since its cost is less or equal (line 18). This process is then iterated for each path of the set. Finally, the routine is iterated m times (line 13), in order to generate the desired level of redundancy.

While computing paths, the algorithm also checks whether the solution satisfies path-related QoS constraints, which are a function of the path cost (maximum latency and BER are passed to the Dijkstra routine as parameters). If any path does not satisfy all constraints, the algorithm returns an empty set of paths. When all paths are computed, the algorithm also checks constraints on link maximum bandwidth, and device maximum power consumption and out-degree (line 21). Even if some of these constraints are not fulfilled, an acceptable solution may still be obtained simply by ripping-up and rerouting some of the paths in excess through other nodes in the graph [18]. In particular, the algorithm selects the paths to be rerouted by sorting them in terms of cost, and by rerouting the ones with the lowest cost (line 22), since those are more likely to fulfill all constraints also after rerouting. Constraints are checked once again after rerouting: if the network still does not satisfy them, the algorithm returns failure.

At the beginning of the section, we argued that the user can use the two synthesis algorithms interchangeably, depending on the ongoing design phase. However, the MILP-based algorithm is more flexible, since it is able to explore different regions of the design space by tuning parameters α and β, while the heuristic, as it has been presented so far, always returns the same network topology. Consequently, when switching from one synthesis algorithm to the other, the heuristic could return an unnecessarily perturbed network topology, if it is not able to emulate the MILP parameter tuning. To partially overcome these problems, we present in the following paragraph four synthesis strategies that can be pursued with the heuristic algorithm to emulate the tuning of the MILP parameters α and β.

The first synthesis strategy is equivalent to $\beta > \alpha$, and it is obtained by populating the network with all W *VR*s from the beginning: the returned network has lower transmission power consumption, since a tailored path is synthesized for each end-device, but higher fixed power consumption, since there is less sharing of *VR*s among different paths.

In a second synthesis strategy, the network is populated with few *VR*s at the beginning (lower effective W, $W_{eff} < W$), and the number of *VR*s (W_{eff}) is incrementally increased every time Algorithm 1 returns with a failure. In order to limit the number of iterations, W_{eff} is increased exponentially, for a total of $O\left(\log\left(W\right)\right)$ iterations. This strategy is equivalent to $\beta < \alpha$, since the algorithm finds a solution with fewer routers than the previous strategy, but more power is spent in transmission, since paths are made of more hops with worse link quality.

Instead of adding fewer *VR*s to C before synthesizing the network, the algorithm produces results consistent to the ones obtained by setting $\beta < \alpha$ in the MILP also by disconnecting as many *VR*s as possible after the synthesis, and checking that all constraints are still satisfied. In this third strategy, *VR*s are sorted in terms of transmission power consumption after running Algorithm 1; the routine then tries to disconnect them using a binary search, starting with the least used. The binary search results in a logarithmic number of iterations, so it makes the strategy computationally practical.

Finally, a fourth possible strategy combines the incremental addition of *VR*s and the post-processing of the synthesized network: this is equivalent to setting $\beta << \alpha$.

All strategies have been implemented, and the designer can use the above guidelines to choose among them, depending on the desired result. In our example, we used $\alpha = 0.6$ and the second strategy: this choice minimizes the number of components and installation dollar cost of the network, at the expense of higher maintenance costs, since each device will have the radio turned on more often and hence have a shorter battery life.

C. Experimental Results

To better benchmark the implemented algorithms, we synthesized reduced and extended versions of the example presented in Section III, while varying the number of end-devices (D). Results related to the network designed in Section III are reported on the third row. In all examples, we first run the MILP-based synthesis to get a starting point for the design; after adding measurement results, we run both the MILP-based and the heuristic-based syntheses to compare their

TABLE I
PERFORMANCE OF THE SYNTHESIS ALGORITHMS

Input	MILP ($\alpha = 0$)				First Strategy				MILP ($\alpha = 0.6$)				Second Strategy				
$\lvert D \rvert$	T[s]	Final	#H	LQ	T[s]	Final	#H	LQ	T[s]	Final	#H	LQ	T[s]	Final	W_{eff}	#H	LQ
25	1980	28	2	2.47	6.8	29	2.16	2.75	1960	19	2.3	2.85	0.82	20	64	2.4	3.2
30	2590	37	1.9	2.43	9.4	37	2.1	2.65	2630	26	2.37	2.7	1.4	26	64	2.5	3
50	3582	53	2.23	2.38	20	55	2.38	2.7	3512	40	2.62	2.8	2.83	39	64	2.76	3.11
Average	**2717**	**39**	**2.04**	**2.42**	**12**	**40**	**2.21**	**2.7**	**2700**	**28**	**2.43**	**2.8**	**1.68**	**28**	**64**	**2.55**	**3.1**
75	TO	-	-	-	28	128	2.6	2.7	TO	-	-	-	52	128	256	2.6	2.7
100	TO	-	-	-	95	77	2.6	3	TO	-	-	-	25	39	128	3	3.2

performance. We solved the MILP-based synthesis problems using the Matlab API functions to LPsolve [19]. The values for p, e^{RX}, e^{TX} were taken from measurements reported in [20]. We also synthesized the network using all four strategies of the heuristic-based algorithm. The first, second and third strategies returned topologies similar to the ones obtained after solving the MILP for $\alpha = 0/0.6/0.45$, respectively. These values of parameter α show that the strategies are tailored to the synthesis of networks where transmission power is higher than standby and computation power. The fourth strategy returns results similar to the second one because the fewer added routers are all necessary to guarantee connectivity, so it will not be further considered in the following.

Table I summarizes the results in terms of computation time (T) (on an Intel T7300 2GHz, 2GB of RAM), number of final routers, average number of hops per path (H), and average link quality in the synthesized network (LQ). The value of W_{eff} is also reported for the second strategy, which incrementally increases it. The fourth row summarizes the performance averages taken only on the networks for which all algorithms terminated within the Time Out (TO), set to one hour. We only report results obtained for $\alpha = 0$ ($\alpha = 0.6$) to be compared with the first (second) strategy, due to space limitations. The MILP-based synthesis outputs a network with 7.7% (4.7%) fewer hops and 10.3% (9.6%) better LQ, on average, with respect to the first (second) strategy. On the other hand, it is able to synthesize networks only up to 50 devices within the TO, while all polynomial time synthesis strategies terminate within tens of seconds.

Overall, experimentation results show that the heuristic-based approach is able to return close-to-optimal results in a short time, thus enabling an interactive usage of the tool. Enhanced performance can then be obtained by running a final MILP-based synthesis.

While a rigorous comparison with other tools is not possible, since neither the code nor the used test benches are publicly available, we mention that the running time of the heuristic-based algorithm is more than two orders of magnitude faster than the one reported in [6] for networks of comparable input size (30 end-devices); on the other hand, it is slower than the algorithm in [9], even though its complexity is lower. We think the reasons for poorer performance are: 1) the algorithm in [9] does not take QoS constraints into account, so it performs fewer checks and it finds more quickly what it considers an acceptable solution, and; 2) Matlab code, which is used in our implementation, is not compiled but interpreted.

V. CONCLUSION

In this paper, we presented a tool to assist the design flow of WSNs for building applications. The tool optimizes network resiliency and power consumption by emulating the behavior of routing algorithms. We cast the synthesis problem into an optimization problem, and we proposed a MILP-based algorithm that returns an exact solution, and a polynomial-time heuristic that returns close-to-optimal results in a shorter time. Users can use the exact algorithm to generate a tentative initial topology and to further improve network performance at the end of the design; the heuristic is most suitable during intermediate design steps when new information is incrementally added by designers with different field of expertise.

As future work, we plan to validate the proposed framework by deploying the network whose design was used as an example in the paper. Collected measurements on network resiliency and lifetime will allow us to further tune the synthesis strategies.

VI. ACKNOWLEDGMENTS

The authors thank Z. Hongye for help in developing the first prototype of the graphical interface. The authors acknowledge the support of the Multiscale Systems Center, one of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.

REFERENCES

[1] "Energy Future: Think Efficiency", *The American Physical Society*, September 2008.

[2] M. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri, "A Framework for Modeling, Simulation and Automatic Code Generation of Sensor Network Application", *Proc. of SECON '08*, pp. 515–522.

[3] http://nsnam.isi.edu/nsnam/index.php/Main_Page

[4] Y. Wang, C. Hu, and Y. Tseng, "Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks", *Proc. of WICON '05*, pp. 114–121.

[5] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", June 2010.

[6] A. Pinto, M. D'Angelo, C. Fischione, E. Scholte, and A. Sangiovanni-Vincentelli, "Synthesis of Embedded Networks for Building Automation and Control", *Proc. of ACC '08*, pp. 920–925.

[7] A. Guinard, A. Mc Gibney, and D. Pesch, "A Wireless Sensor Network Design Tool to Support Building Energy Management", *Proc. of BuildSys '09*, pp. 25–30.

[8] H. Kim, T. Kwon, and P. Mah, "Multiple Sink Positioning and Routing to Maximize the Lifetime of Sensor Networks", *IEICE Trans. Commun.*, vol. E91-B, no. 11, November 2008.

[9] M. Ahlberg, V. Vlassov, and T. Yasui, "Router Placement in Wireless Sensor Networks", *Proc. of MASS '06*, pp. 538–541.

[10] P. Levis, A. Tavakoli, and S. Dawson-Haggerty, "Overview of Existing Routing Protocols for Low Power and Lossy Networks", April 2009.

[11] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-Aware Wireless Microsensor Networks", *IEEE Signal Processing Magazine*, no. 3, pp. 40–50, March 2002.

[12] M. Lu, P. Steenkiste, and T. Chen, "Design, Implementation and Evaluation of an Efficient Opportunistic Retransmission Protocol", *Proc. of MobiCom '09*, pp. 73–84.

[13] The MathWorks - Matlab and Simulink for Technical Computing. http://www.mathworks.com

[14] G. L. Stüber, "Principles of Mobile Communication", *Kluwer Academic Publishers*, 1996.

[15] S. Zvanovec, P. Pechac, and M. Klepal, "Wireless LAN Networks Design: Site Survey or Propagation Modeling?", *Radioengineering*, vol. 12, no. 4, December '03, pp. 42–49.

[16] http://www.willow.co.uk/TelosB_Datasheet.pdf

[17] P. Pechac and M. Klepal, "Effective Indoor Propagation Predictions", *Proc. of VTC '01*, pp. 1247-1250.

[18] W. Dees and P. Karger, "Automated Rip-Up and Reroute Techniques", *Proc. of DAC '82*, pp. 432–439.

[19] http://lpsolve.sourceforge.net/5.5/

[20] G. de Meulenaer, F. Gosset, F. Standaert, and O. Pereira, "On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks", *Proc. of WIMOB '08*, pp. 580–585.