Routing-aware design of indoor wireless sensor networks using an interactive tool

Routing Aware Design of Indoor Wireless Sensor Networks Using an Interactive Tool

Mohammad Mostafizur Rahman Mozumdar, *Member, IEEE,* Alberto Puggelli, *Student Member, IEEE,* Luciano Lavagno, *Member, IEEE,* Alberto L. Sangiovanni-Vincentelli , *Fellow, IEEE*

***Abstract*—In this paper, we present an interactive design tool that can assist 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 heuristics that returns close-to-optimal results in a shorter time. We analyze the performance of the designed networks by using OPNET simulation. Results show that our tool can assist designing sensor networks that have high throughput and consume power efficiently.**

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

I. Introduction

Applications for Wireless Sensor Networks (WSN) have been expanding rapidly in many fields such as factory automation, environmental monitoring, security systems and in a wide variety of commercial and military areas. 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 [15]. 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, and since it is flexible enough to accommodate changes in the building usage which are common over its life-cycle.

To reduce the energy consumed by the building stock, both new constructions and existing buildings must be equipped with control solutions that increase the building energy efficiency. From the BAS perspective, these solutions are combined with optimal architecture design (for the sensor-actuator network) and the use of advanced control algorithms that, based on measurements collected by sensors, compute an optimal control policy and send commands to actuators. Thus, the sensor-actuator network is a key element of building automation systems. The selection of an optimal network is driven by several metrics such as cost, network lifetime, throughput, and also the resiliency to sudden failures in the network architecture.

Designing efficient WSN-based solutions for building automation is a complex task, and we expect that multi-disciplinary teams will be involved in specifying, designing, implementing, deploying and maintaining them. These teams could involve architects as well as civil, electronics and telecommunication engineers, all with a common goal of sharing a unified representation of the network design, in order to optimize sensing, actuation, communication, power supply, maintenance access and so on.

In order to support the rapid design, prototyping and deployment of WSN for BAS applications, we aim to develop a design framework which provides rich interfaces to capture inputs from designers with different fields of expertise, and a tool chain that processes these inputs and guides towards robust solutions. Along this path, tools and methodologies have been developed for the modeling, simulation and automatic code generation of WSN applications [3], [4], [5].

In this paper, we present a graphical tool to support the design exploration and synthesis of network topology, i.e. the locations of nodes. An optimal topology should guarantee connectivity and support all functional requirements (e.g. latency, throughput, etc.), while optimizing several metrics such as cost, network lifetime, resiliency and others. The tool allows the designer to specify the location of the network end-devices (e.g. sensors, actuators and gateways) on a 2D schematic of the floor plan, and to enter quantitative parameters to capture the node behavior (e.g. bit rate, transmission power, etc.). Moreover, the tool provides interfaces to specify the characteristics of the network stack such as the behavior of the routing protocol. To take environment effects into account, our tool contains models of physical channels and obstacles (e.g. walls). After collecting these data, the tool can guide users towards an optimal placement of network relay-nodes (e.g. routers). Iterative refinements of the placement are also proposed when the users enter new data.

The tool facilitates users by reducing design time and by improving the quality of the network topology with respect to a simulation-based approach, in which designers have to simulate several different topologies and select the most performing one, with no guarantee of optimality. Based on user-defined requirements, our tool proposes an initial solution, and then users can tune it iteratively to increase network robustness.

Moreover, we identify common configuration parameters (e.g. propagation loss coefficients for an indoor scenarios) for wireless BAS design and make them available to users in such a way that both novice and advanced users can employ their expertise and knowledge to develop robust design solution, by either using our built-in library of models and protocols, or by tuning its parameters.

This paper extends our work described in [1]. In particular, we extend its features and present a detailed case study to design multi-hop network which is based on hierarchical routing protocol. We validate and analyze the designed sensor network by modeling and simulating it using a network simulator (OPNET[16]). Results confirm that the designed network has high throughput, consumes energy efficiently and meets all required specifications. We provide details on how on-site propagation loss measurements can be entered in our tool to increase its accuracy in modeling of the wireless medium and propose a modified mathematical formulation of the network synthesis problem, which removes possible ambiguous cases not considered in [1]. Moreover, we elaborate the whole design flow so that the reader can have a better understanding of the proposed solutions for network synthesis and robust design.

The rest of the paper is organized as follows. We discuss related works on strategies for network synthesis for BAS in section II. In section III we give some background on WSN-oriented routing algorithms. Section IV describes the proposed tool and how it supports the design flow of a WSN. In section V we show details on the synthesis problem formulation, and we propose algorithms to solve it. In section VI, we show a complete case study where a network is synthesized using the proposed tool and its performance analyzed using OPNET. Final conclusions are drawn in section VII.

II. Related Work

The problem of network synthesis has already been addressed in the past. Contrarily to [6], which considers networks made of only 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) [7]. Our goal is to determine optimal locations for the routers.

In [8], [9], the authors present a design tool for the automated synthesis of WSNs satisfying connectivity and Quality of Service (QoS) constraints. The very general synthesis algorithm presented in [8] is based on a Mixed-Integer Linear Program (MILP). The framework proposed in [8] allows also the introduction of specific ad hoc algorithms for particular domains. A possible strategy to make the MILP approach scalable is to decompose the synthesis problem into an optimal number of local subproblems [9]. 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 differ from previous work, because our proposed algorithms can synthesize network structures that are much more general than the ones analyzed in [9], and in a much shorter time with respect to [8]. 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 [10], 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 [11]. Our tool increases network resiliency by augmenting the network 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 [11] 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, one of the core contributions 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, mobile nodes) [7]: 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 [12]. Since wireless transmission is the major source of power consumption in a WSN [13], 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 [11].

We conclude this section by pointing out that the proposed approach does not violate the layering principle, since it only models the behavior of RAs (path ranking and traffic patterns) without any assumption on their specific implementation.

III. Background

In order to motivate several choices that led to the final implementation of our tool, we now briefly describe the most commonly used traffic patterns that RAs for building applications should support [7], and some of the guidelines for setting path-ranking cost functions [12].

The large variety of BAS and the severe constraints on power consumption suggest the use of heterogeneous traffic

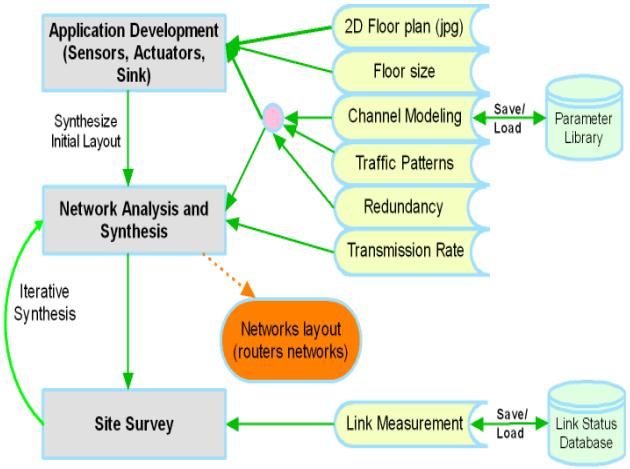

Fig. 1. Work-flow of the tool

patterns to route packets, so that each application can choose the one that results in the best performance. The basic traffic pattern to be supported is gateway/end-device unicast, since each device needs to communicate with the gateway during its lifetime. In principle, unicast is able to guarantee the functionality of most applications. On the other hand, a large amount of power and computation resources can be saved with the use of multicast and Peer-to-Peer (P2P) communication. 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 particularly suitable for applications in which local control is enough to guarantee the desired performance (e.g. HVAC, lighting control): P2P relaxes requirements on network delay, and it usually results in less power consumption, since fewer hops need to be traversed to process the data. Finally, RAs should also support mobile devices (e.g. remote controllers): this capability reduces the number of required end-devices, and it might be required for some applications (e.g. health monitoring).

Every RA ranks possible paths from source to destination according to some predefined cost function. As suggested in [12], RAs for WSNs should simultaneously 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 layers). In the following, the link quality is evaluated in terms of the estimated Propagation Loss (PL) between two devices: even if 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) [14]. Finally, nodes should be allowed to assert their willingness to route traffic: battery-powered devices might refuse to route packets if the traffic routed through them substantially reduces their lifetime.

IV. Design Flow

In this section, we present the design flow of our tool, developed using the Matlab GUI Development Environment [18]. The workflow of the tool is shown in Figure 1.

In the *Application Development* phase, 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 one to upload a 2D floor plan of the environment, where end-devices (sensors/actuators) and routers can be placed simply by clicking on the floor plan area. The users can specify or load from the *Parameter Library*: 1) the floor plan and size; 2) the channel modeling parameters (i.e. transmission and reception power/gain, radio frequency, wall loss); 3) the desired level of redundancy, and; 4) the traffic patterns (peer-to-peer, unicast, multi-cast) between different types of nodes. For example, peer-to-peer 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. Moreover, our tool can model mobile end devices for which users need to configure the trajectory by selecting multiple waypoints.

After specifying the BAS requirements and the parameters related to wireless networks, the tool synthesizes a tentative layout of the network with the desired level of redundancy and QoS. At this step, errors may occur because the tool models the quality of the wireless link using Free-Space (FS) and Multi-Wall (MW) propagation models [19], and it assigns a default value of bit rate to nodes. Nevertheless, the designed topology represents a good starting point for the subsequent refinement steps, which will require more information from the designers.

In the *Network Analysis and Synthesis* phase, 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. 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.

Secondly, 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.

Finally, 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 [20].

At the network level, the field engineer can input in the tool accurate values for the parameters of the FS and MW propagation models, determined through measurements. At the single link level, the tool can store measured values of PL into a *Link Status 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.

More accurate models of the BAS (e.g. [22]) and of its environment 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 the new links might need to be assessed. However, we will show in Section V that the number of measurements needed is roughly linear in the network size, so data collection is simplified, and the database can be efficiently processed.

V. 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 and a base station D, and a set of pre-defined 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 *A*dditional Routers and their corresponding *l*ocations 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 IV. In particular, paths $p \in Q_{multi}$ are clustered in (possibly overlapping) Multicast Groups (MG), where a local Base Station (BS) sends each packet to more than one node. In an MG, the same messages are transmitted along all paths, so we set $r^q = r^{MG}\ \forall q \in MG$.

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. Other approaches have been proposed in the literature (e.g. [9]) to place VRs only at locations that are most promising for final deployment (e.g. close to walls). These approaches could be seamlessly integrated in our framework without changing the overall flow, should experimentation suggest it. The synthesis algorithms then select the set $AR \subseteq VR$ to optimize for power consumption, while satisfying all constraints.

The algorithms are different from one another because they trade-off the optimality of the solution with running time. In Section V-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 the execution time of algorithms for the solution of MILPs is not polynomially bounded, so solving them is not in general computationally efficient. High running times have been reported even for the synthesis of small networks ($\sim$ 30 end-devices) [8]. During the network design cycle, a faster response time from the tool could be desired because new data (e.g. from *Site Survey*) 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 V-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.

A. MILP-based Synthesis

The MILP representation is based on the one proposed in [8], 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 $q \in Q$. Moreover, each edge of C is also assigned a variable w_{ij}, which is set to true if any connection uses that link. In order to correctly compute the power consumed in transmission, we associate to each variable $y_{ij}^{q,k}$ a constant $r_{ij}^{q,k}$, which models the bit rate of the transmission through the link (n_i, n_j) along the k^{th} replica of path q. For a path $q \in Q \setminus Q_{multi}$, we set $r_{ij}^{q,k} = r^q$. On the other hand, for each Multicast Group (MG) $\in Q_{multi}$, we set:

$$\sum_{q \in MG} r_{ij}^{q,k} \cdot y_{ij}^{q,k} = r^{MG} \cdot z_{ij}^{MG,k} \tag{1}$$

where $z_{ij}^{MG,k} = \{0,1\}$. Constraint 1 sets the bit-rate to be either null, if the link is not used by any path in the MG, or to saturate to r^{MG}, no matter how many paths in the MG

use that link. The correct value of $z_{ij}^{MG,k}$ can be assigned by adding the constraint

$$\sum_{q \in MG} y_{ij}^{q,k} \leq B \cdot z_{ij}^{MG,k}$$

where B is a big number (e.g. $B = |MG|$), so that $z_{ij}^{MG,k} = 1$ only if the link is used at least by one path in MG. The Left-Hand Side (LHS) of Constraint 1 is not linear, so the constraint cannot be added to the MILP formulation. To overcome this problem, in the formulation below we substitute the LHS with the right-HS, which is linear.

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

A path $(s,d) \in Q$ is connected if there exist a solution to the equation $C\mathbf{y} = \mathbf{b}$, where $\mathbf{b}[s] = -1, \mathbf{b}[d] = 1, \mathbf{b}[j \neq s,d] = 0$. The *Topological* constraints enforce that:

- m replicas $\forall q \in Q$ are connected (1),
- 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
- routers are installed, if they are used (3).

The *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) sets the value of $z_{ij}^{MG,k}$ for each link, so that the bit-rate through the link is bounded by a constant even though multiple paths belonging to the same MG are routed through it: this constraint 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; $(7-8)$ limit the maximum fan-in of a node; $(9-10)$ 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 (11) limits the maximum average power consumption of a node, where: p is the fixed power consumption (standby and processing) of the router; e_{ij}^{TX} and e_{ij}^{RX} is 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 [9]. 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.

We assume in the following that matrix C is sparse, due to the limited connectivity range of wireless devices. Hence in most practical cases 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 IV. 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 III. Moreover, edges entering user-defined routers ($r \in R$) are heuristically counted as a half hop, so they are the preferred choice to route paths, with respect to VRs.

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 $O\left(m \cdot |Q| \cdot |N| \cdot \log\left(|N|\right)\right)$.

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 VRs 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 [23]. 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 one, as it has been presented so far, always returns the same network topology. Consequently, when switching from one synthesis algorithm to the other, the heuristics 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 heuristics algorithm to emulate the tuning of the MILP parameters α and β.

1) A synthesis strategy equivalent to $\beta > \alpha$ is obtained by populating the network with all W candidate VRs. The resulted 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 VRs among different paths.
2) In a second synthesis strategy, the network is populated with few VRs at the beginning (lower effective W, $W_{eff} < W$), and the number of VRs (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.
3) Instead of adding fewer VRs to C before synthesizing the network, the algorithm can produce results consistent with the ones obtained by setting $\beta < \alpha$ in the MILP also by disconnecting as many VRs as possible after the synthesis, and checking that all constraints are still satisfied. In this third strategy, VRs 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.
4) Finally, a fourth possible strategy combines the in-

TABLE I
PERFORMANCE OF THE SYNTHESIS ALGORITHMS

Input	MILP ($\alpha = 0$)				First Strategy				MILP ($\alpha = 0.6$)				Second Strategy				
$\|D\|$	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

cremental addition of VRs 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.

C. Algorithm Benchmarking

In order to benchmark the implemented algorithms, we synthesized several different instances of the network presented in Section VI, while varying the number of end-devices (D). 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 performance. We solved the MILP-based synthesis problems using the Matlab API functions to LPsolve [24]. The values for p, e^{RX}, e^{TX} were taken from measurements reported in [25]. 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 last row summarizes the average performance in bold. 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 5% (8%) fewer hops and 11% (10%) 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 one hour Time-Out (TO) that we set, while all polynomial time synthesis strategies finish within tens of seconds.

Overall, experiments 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 [8] for networks of comparable input size (30 end-devices); on the other hand, it is slower than the algorithm in [11], even though its complexity is lower. We think that the reasons for poorer performance are: 1) the algorithm in [11] 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.

VI. CASE STUDY OF DESIGNING INDOOR SENSOR NETWORKS

To verify our tool and design methodology, we designed a sensor network for our office. In the following sub-sections we first provide a brief overview of the routing algorithm that we modeled to guide the network synthesis; we then describe how we designed the sensor network using our tool; finally, we present simulation results obtained with OPNET, which confirm the robustness of the synthesized network to node failures.

A. Hierarchical WSN Architecture for Building Automation and Control Systems

The modeled routing algorithm is inspired to the one presented in [2]: the algorithm implements gradient based routing for collecting data and a label switching table for disseminating configuration commands, thereby supporting upstream and downstream data flows across the network. The algorithm assumes that the network architecture is organized into a hierarchy of components that include *end devices*, *access points* and a *base station*. An end device is a sensor or an actuator. Each device has a *floor* and a *room* ID and is able to join the network through any access point on the same floor (in a star configuration), which also ensures in-network load balancing. The requirement to join nodes on the same floor is due to the fact that most office buildings are divided into floors that may be occupied by different companies, whose heating and air conditioning is managed independently. A sensor node could be configured for periodic or threshold crossing reporting depending on the quantity to be measured (e.g temperature, light and occupancy).

An access point is part of the backbone network that is used for data collection and command routing. These devices can be always on and capable of low power listening to minimize energy consumption. Each access point has a *floor* ID and a *network-wide unique* ID. These devices permit end-devices to join the network and to send data to collection

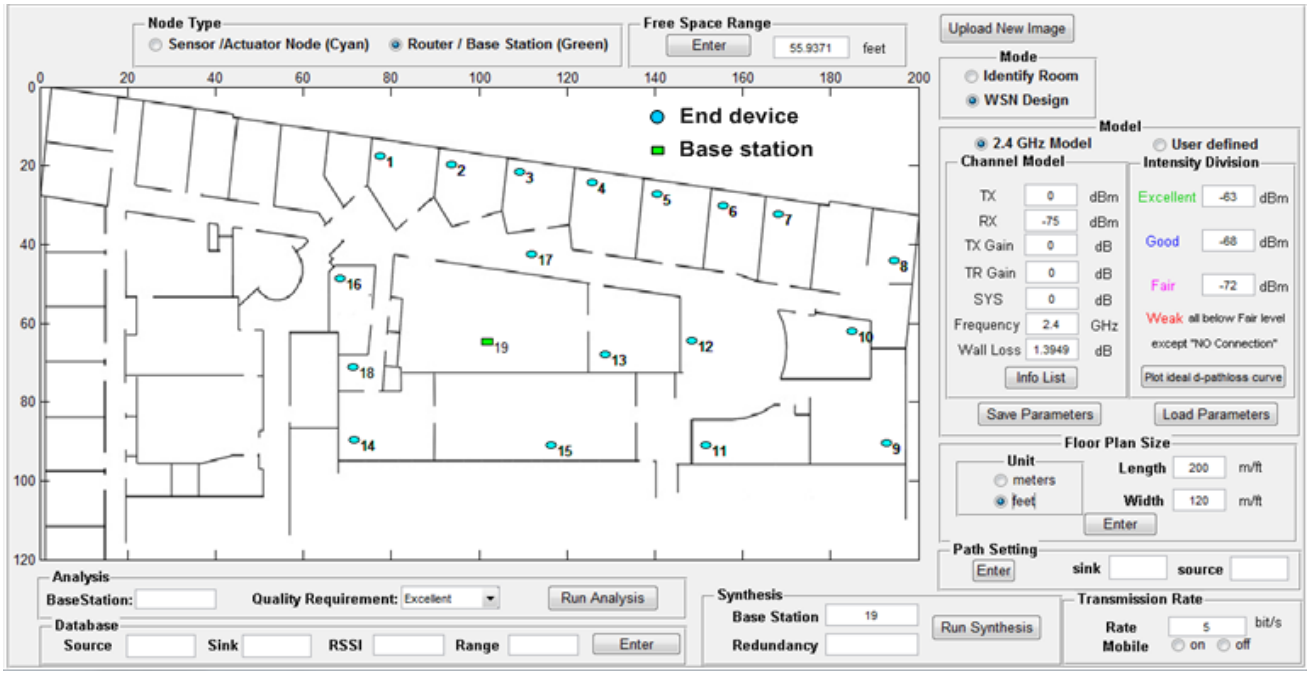

Fig. 2. The figure shows the GUI interface of the tool and the loaded floor plan, where end-devices and the base station have been placed.

points, to construct aggregated packets and to route them to the base station. The base station uses access points to configure sensors and actuators.

A base station has wireless connections with access points and an Ethernet connection for LAN access. A base station works as a master and initiates the formation of the backbone network. It collects the sensor data and logs them into the database, which can be analyzed by a suitable control algorithm. There could be one or more base stations for the whole system, depending on the network size.

The network is formed and activated by following a series of phases. First, the backbone network is formed. The base station and the access points participate in this phase. Then, end devices join with access points. Each access point sends a report of the connected end devices to the base station. The base station sends configuration commands to activate/deactivate sensors and actuators. After activation, end-device sensors periodically report to the associated access point node, which aggregates sensor data to construct a single packet that is routed to the base station.

The main purpose of the backbone network is to support routing of messages in the network. We use gradient-based routing to form the backbone. The formation of the backbone network is initiated by the base station which constructs and broadcasts the Beacon Packet (BP). Access points that are in the radio range of the base station receive the BP and set their level to 1, and base station as their parent. An access point always broadcasts the BP after updating its level and/or parent value. When broadcasting, an access point node modifies the BP by incrementing the level by one, and by setting *senderID* to its own ID. While broadcasting the BP, the node waits for a random time and uses a simple CSMA/CA protocol at the MAC layer to reduce collisions. This process of controlled flooding continues until the backbone network is formed.

A tributary network is then formed between access points and end devices. After power-up, each end-device constructs and broadcasts the EJRR (End-device Join Request Response) packet, which contains *floorID, nodeID, packet type* and *gradient-level* information. Any access point node after receiving the EJRR packet checks the floorID. If the end-device joining request is coming from other floors it is ignored, otherwise it modifies the EJRR packet with its information and then rebroadcasts it. An end-device might get multiple responses from different access points, but it chooses the access point with the lowest gradient. Ties are broken choosing the best link quality. After network formation, the base station configures the end-devices to activate the desired sensors and their reporting interval. After receiving configuration commands, the end-devices start sending sensing data to the base station based on these settings. The base station logs and analyzes sensor reports and, if necessary, it sends commands to the actuators based on application requirements. Details of the routing, network formation and maintenance are described in [2].

B. Network Design

We identified 18 locations to collect report for temperature and light. We allocated temperature sensors for some rooms and light sensors for the corridor and the lounge (12 temperature sensors, 6 light sensors). After identifying sensor locations, we marked their positions on the 2D diagram shown in Figure 2. We then identified the location of the base station. Temperature and light sensors communicate to the base station via unicast, which is the default traffic pattern. We set the propagation model to use the 2.4GHz frequency and used default values to rank link quality on a scale from 1 to 4 (as

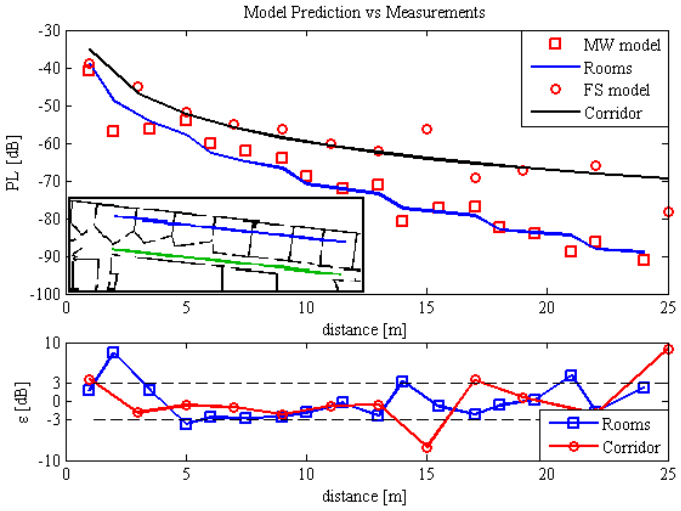

Fig. 3. The figure shows the measured and estimated values of PL for the two scenarios (top); the location in which we took the measurements (inset); the absolute error in the model (bottom).

explained in Section V-B). We also set the redundancy value $m = 1$, and $R = \emptyset$, i.e. no initial routers preferences. Using the MILP formulation, the tool synthesizes a first layout of the network with the desired level of redundancy, and QoS requirements, based on the information entered up to this point. At this step, some choices may need to be corrected because the tool modeled the quality of the wireless link with the free-space and multi-wall propagation models, and it assigned a default value of bit rate to nodes. Nevertheless, the output topology represented a good starting point for the subsequent refinement steps, which required more information from the designer. In the first synthesis, the tool added 14 routers to meet the specifications.

As a first design refinement step, we added the actual bit rate for each node, by taking into consideration the MAC layer and considering packet headers. These data can be retrieved for each type of packet (e.g. beacon, EJRR) in [2]. In our application, temperature and light sensor report every 2 minutes. A new synthesis can be run after adding this information. Based on the result of the previous step, the synthesis algorithm at first recalculates the load at each router and in case a router is overloaded, it splits the load by adding more routers. In our case, 3 extra routers were added to cope with these additional constraints.

As mentioned earlier, our tool has the capability to integrate data collected from the site survey to adjust the propagation model. To further refine the design, we took measurements of the PL between two nodes, as a function of the distance between them, using TelosB nodes by Crossbow [21]. We considered both a corridor scenario and a multi-room scenario. For the FS and MW models, we used the standard formulas [19].

$\text{PL}_{FS} = L_0 + 10 \cdot n \cdot \log(d) + \Omega_{shadowing}$

$PL_{MW} = L_{FS} + \#W \cdot L_W + L_{W,0} + \Omega_{shadowing}$

Figure 3 shows the measured (markers) and modeled (lines) values of PL, obtained from fitting ($L_0 = 37.6dB, n = 2.2, L_W = 3.2dB, L_{W,0} = 1.2dB$, $\Omega \sim \mathcal{N}(0, 2.25dB)$), while keeping the transmission power constant to $P_{TX} = 0dBm$. The inset shows the part of the floor where we collected data

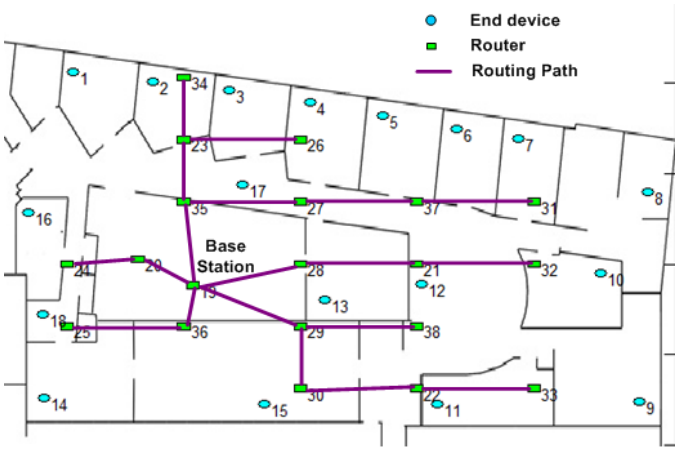

Fig. 4. Non-Redundant network ($redundancy = 1$) - each end device has at least one path towards the base station.

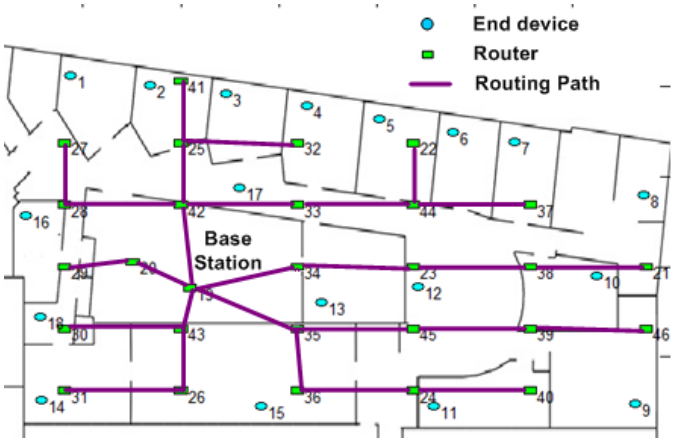

Fig. 5. Redundant network ($redundancy = 2$) - each end device has at least two disjoint paths towards the base station.

using TelosB nodes. While most PL values are predicted by the model with an error (ϵ) within $\pm 2\sigma_\Omega = \pm 3dB$ of the shadowing noise, a few values are very far ($|\epsilon_{max}| = 9dB$). Analyzing the data, we found that larger errors occur when the environmental conditions present discontinuities (e.g. the presence of the hall at the end of the corridor in the inset of Figure 3). In order to correct these errors, we also took measurements of the PL of all the links selected in the previous synthesis step, and stored these data in the database.

Figure 4 shows the final layout of the network, obtained after two iterations of measurements. The algorithm added 2 more routers (for a total of 19 routers), and it changed the location of three routers after taking into consideration the measurements of the link quality. In the network synthesis phase, we used $\alpha = 0.6$ and the second strategy (described in sub-section V-B): 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 switch the radio on more often and thus consume more battery power.

Following the same steps, we also designed a redundant network, with $m = 2$, for the same number of end devices. Now by setting redundancy equals to two, we enforce the

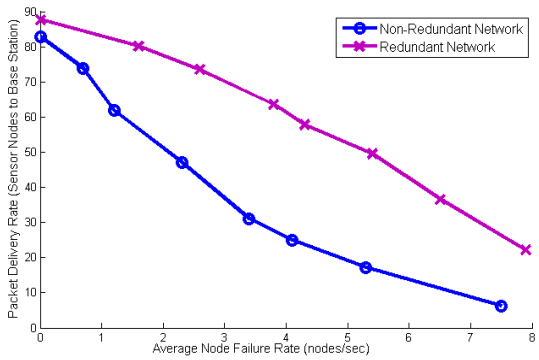

Fig. 6. Packet delivery rate as a function of the average node failure rate.

constraint that for each end-device node there should be at least two disjoint paths towards the base station. Figure 5 shows the designed redundant network, which includes 27 router nodes.

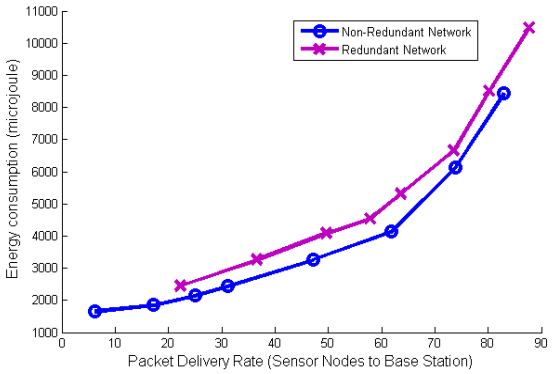

Fig. 7. Energy consumption at different packet delivery rate. We assumed that for each packet transmission and reception a node consumes $e^{TX} = 0.6\mu J$ and $e^{RX} = 0.7\mu J$ respectively [25].

C. Network Verification by Simulation

To verify the functionality and QoS of the designed sensor network, we modeled the nodes (end device, router, and base station) and routing protocol in OPNET. Overall, the node models contain around 1.4k lines of code. OPNET has three hierarchical component levels: the network level creates the topology of the network, the node level defines the behavior of the node and controls the flow of data between different functional elements inside the node, and the process level describes the underlying protocols by using finite state machines (FSMs). Based on this node library, we designed two network scenarios both for the *redundant* and the *non-redundant* cases. While constructing these scenarios in OPNET, we enforced connectivity and link status between nodes to be the same as in the network synthesized by the tool. We also configured the radio parameters in OPNET, so that they match the parameter settings used during network synthesis.

Simulations confirmed network functionality and QoS when all nodes were operating properly. In the following, we only present the results of a sensitivity analysis of network robustness with respect to node failures, obtained by running several simulations while varying the average node failure rates for router nodes. In particular, we collected simulation results in terms of: 1) Packet Delivery Rate (PDR), defined as the ratio between the number of packets received by the base station and the number of packets sent by the end-devices, which we used as a synthetic estimation of network QoS, and; 2) energy consumption. Figure 6 depicts the PDR for both the redundant and non-redundant networks at different average node failure rates. With no router failures, the PDR for the Redundant (R) and Non-Redundant (NR) networks is 87.65% and 82.94% respectively (packet drops are due only to collisions in accessing the wireless channel). If we inject random failures in the routers (by deactivating/activating them for random periods of time), the non-redundant setup drops a much larger number of packets compared with the redundant one. This confirms that the redundancy introduced by our tool is capable of increasing network robustness, as desired. Moreover, Figure 7 shows that the increase in robustness is obtained with a moderate increase in the network power consumption (on average $\Delta E^R_{tot} = +22.75\% E^{NR}_{tot}$).

VII. 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 both an MILP-based algorithm that returns an exact solution, and a polynomial-time heuristics that returns close-to-optimal results in a shorter time. Designers can use the exact algorithm to generate a tentative initial topology and to further improve network performance at the end of the design; the heuristics on the other hand is most suitable during intermediate design steps when new information is incrementally added by designers based e.g. on field measurements.

We validated the proposed tool by designing a sensor network for our office, and verified its functional correctness and robustness using simulations in OPNET.

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.

References

[1] A. Puggelli, M. Mozumdar, L. Lavagno, and Alberto L. Sangiovanni-Vincentelli, "A Routing-Algorithm-Aware Design Tool for Indoor Wireless Sensor Networks", *Proc. of International Conference on Computing, Networking and Communications '12*, pp. 964 - 969

[2] M. Mozumdar, A. Puggelli, A. Pinto, L. Lavagno, and Alberto L. Sangiovanni-Vincentelli, "A Hierarchical Wireless Network Architecture for Building Automation and Control Systems", *Proc. of ICNS '11*, pp. 178–183.

[3] 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.

[4] M.M.R. Mozumdar, , L. Lavagno, L. Vanzago, and Alberto L. Sangiovanni-Vincentelli. HILAC: A framework for Hardware In the Loop simulation and multi-platform Automatic Code Generation of WSN Applications. In *Proc. of SIES*, pages 88-97, Italy, 2010.

[5] Z. Song, M.M.R. Mozumdar, M. Tranchero,L. Lavagno, R. Tomasi, and S. Olivieri. Hy-Sim: model based hybrid simulation framework for WSN application development. In *Proc. of SIMUTools '10*, pages 1-8, Spain, 2010.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

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

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

[16] Application and Network Performance with OPNET http://www.opnet.com/

[17] The Network Simulator - NS-2 http://isi.edu/nsnam/ns/

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

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

[20] 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.

[21] http://www.willow.co.uk/TelosB$_Datasheet.pdf$

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

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

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

[25] 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.