CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	A Cooja-based tool for coverage and fifetime evaluation in an in-building sensor network.
Authors	Sitanayah, Lanny;Sreenan, Cormac J.;Fedor, Szymon
Publication date	42419
Original Citation	Sitanayah, L., Sreenan, C. J. and Fedor, S. (2016) 'A Cooja-Based Tool for Coverage and Lifetime Evaluation in an In-Building Sensor Network', Journal of Sensor and Actuator Networks, 5(1), 4 (22 pp). doi: 10.3390/jsan5010004
Type of publication	Article (peer-reviewed)
Link to publisher's version	https://www.mdpi.com/2224-2708/5/1/4 - 10.3390/jsan5010004
Rights	© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons by Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/). - https://creativecommons.org/licenses/by/4.0/
Download date	2026-09-16 19:02:35
Item downloaded from	https://hdl.handle.net/10468/8976

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Journal of
Sensor and Actuator Networks

MDPI

Article

A Cooja-Based Tool for Coverage and Lifetime Evaluation in an In-Building Sensor Network†

Lanny Sitanayah [1,*], Cormac J. Sreenan [1] and Szymon Fedor [2]

[1] Department of Computer Science, University College Cork, Cork T12 YN60, Ireland; cjs@cs.ucc.ie
[2] United Technologies Research Center (UTRC) Ireland Ltd, 4th Floor Penrose Business Centre, Penrose Wharf, Cork, Ireland; szymon.fedor@gmail.com
* Correspondence: ls3@cs.ucc.ie; Tel.: +1-980-265-3722

† This paper is an extended version of our paper published in Sitanayah, L.; Sreenan, C.J.; Fedor, S. Demo Abstract: A Cooja-Based Tool for Maintaining Sensor Network Coverage Requirements in a Building. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13), Rome, Italy, November 2013.

Academic Editor: Adnan Al-Anbuky
Received: 22 November 2015; Accepted: 7 February 2016; Published: 19 February 2016

Abstract: Contiki's Cooja is a very popular wireless sensor network (WSN) simulator, but it lacks support for modelling sensing coverage, focusing instead on network connectivity and protocol performance. However, in practice, it is the ability of a sensor network to provide a satisfactory level of coverage that defines its ultimate utility for end-users. We introduce WSN-Maintain, a Cooja-based tool for coverage and network lifetime evaluation in an in-building WSN. To extend the network lifetime, but still maintain the required quality of coverage, the tool finds coverage redundant nodes, puts them to sleep and automatically turns them on when active nodes fail and coverage quality decreases. WSN-Maintain together with Cooja allow us to evaluate different approaches to maintain coverage. As use cases to the tool, we implement two redundant node algorithms: greedy-maintain, a centralised algorithm, and local-maintain, a localised algorithm to configure the initial network and to turn on redundant nodes. Using data from five real deployments, we show that our tool with simple redundant node algorithms and reading correlation can improve energy efficiency by putting more nodes to sleep.

Keywords: sensor network; simulation tool; coverage maintenance; lifetime evaluation

1. Introduction

Wireless sensor network (WSN) technologies that are available today allow easy network installation. However, the true cost of WSNs has shifted into the area of the maintenance costs that are inherent to networks with a large number of nodes and a limited battery life. Since the majority of the sensors used are still battery powered, these batteries require regular changing and/or recharging. Moreover, when used for long-term monitoring, a network evolves: new nodes may be added to the system; failed nodes are removed from the system; and user requirements and system functionality change over time. In addition, for an in-building WSN, the building itself may evolve: physical obstacles, such as walls and doors, that affect sensing coverage and communications, are often removed or added when a refurbishment is carried out. In long-term deployments, it is these evolutionary changes that can be highly disruptive to the network, affecting its ability to operate as expected.

Contiki's Cooja [1] is a very popular WSN simulator. Similar to other WSN simulators, such as OMNeT [2], TOSSIM [3], OPNET [4], Ns-2 [5] and Ns-3 [6], it lacks support for modelling sensing coverage, focusing instead on network connectivity and protocol performance. However, in practice, it

is the ability of a sensor network to provide a satisfactory level of coverage that defines its ultimate utility for end-users. In this paper, we introduce WSN-Maintain, a Cooja-based tool for coverage and network lifetime evaluation in an in-building WSN. It takes the floor plan of a building specifying the locations of obstacles, such as walls and doors, the coverage requirement of each region and the locations of sensor nodes. With WSN-Maintain, users can specify different degrees of coverage to different regions. For example, a server room and an electrical room require that each point in the area is covered by at least two sensors, and the rooms must be 100% covered, while it is enough for an office space to be 75% covered, and each point is covered by one sensor. WSN-Maintain runs in parallel with the collect-view tool of Contiki, which was integrated into the Cooja simulator and allows an interactive view of a data-gathering application.

Application developers can use WSN-Maintain to implement and evaluate different techniques/algorithms to maintain coverage, where the tool finds coverage redundant nodes, puts them to sleep and turns them on when active nodes fail and coverage quality decreases. As our use cases, we implement two redundant node algorithms: greedy-maintain, a centralised algorithm, and local-maintain, a localised algorithm to configure the initial network and to turn on redundant nodes. We also incorporate the concept of reading correlation using the inverse distance weighting interpolation method. Sensors whose reading can be predicted using the reading of nearby active sensors can be considered as redundant and can be put to sleep to conserve energy. WSN-Maintain and Cooja simulation results using data from five real deployments show that utilising simple redundant node algorithms with reading correlation can improve energy efficiency by putting more nodes to sleep.

2. Problem Definition

In this work, we assume a pre-deployed WSN in a building that has a connected finite set of sensor nodes and one sink, which are static. While we assume a network with one sink, one can implement a supersink that has connection to many sinks for a multiple sink scenario. The values for parameters used in our tool are based on Tmote Sky hardware [7] and to consider the heterogeneity of device specifications, nodes can have different power levels, sensing ranges and initial energy. For other types of hardware, one can change the values of the parameters accordingly.

Inside a building, we assume obstacles that can limit communication and sensing capabilities are walls and doors. However, for simplicity, we currently do not consider the types of construction materials used. As manual wall classification is labour intensive and requires architectural information that may not be readily available to application developers or network managers, one could implement an automatic wall classification based on link quality measurement, such as in [8].

This work assumes that sensor nodes cannot sense through walls and doors. Furthermore, users may require different coverage degrees for special rooms, such as a server room and an electrical room. To do this, users can select a room and specify the coverage requirement. The total coverage percentage is the ratio of the total area satisfying the coverage requirements to the total interior area of the building.

3. WSN-Maintain

Cooja [1] is a very popular network simulator within the WSN research community, but it does not model sensor coverage, focusing instead on network connectivity and protocol performance. However, in practice, it is the ability of a sensor network to provide a satisfactory level of coverage that defines its ultimate utility for end-users. WSN-Maintain is designed to allow users to evaluate different techniques to turn on redundant nodes to maintain coverage requirements. This tool is run in parallel with Cooja. The relationship between WSN-Maintain and Cooja is depicted in Figure 1.

Draw building floor plan and deploy sensor nodes: With WSN-Maintain, users can import the floor plan of a building from an XML file or manually draw one by clicking and dragging walls and doors on its canvas. While we use walls and doors as indoor obstacles that can limit communication

and sensing capabilities, for simplicity, we currently do not consider the types of construction materials used. In addition to the floor plan, users can click on the canvas to place sensor nodes, click on a room to automatically deploy all nodes for 100% coverage or load their locations from a file.

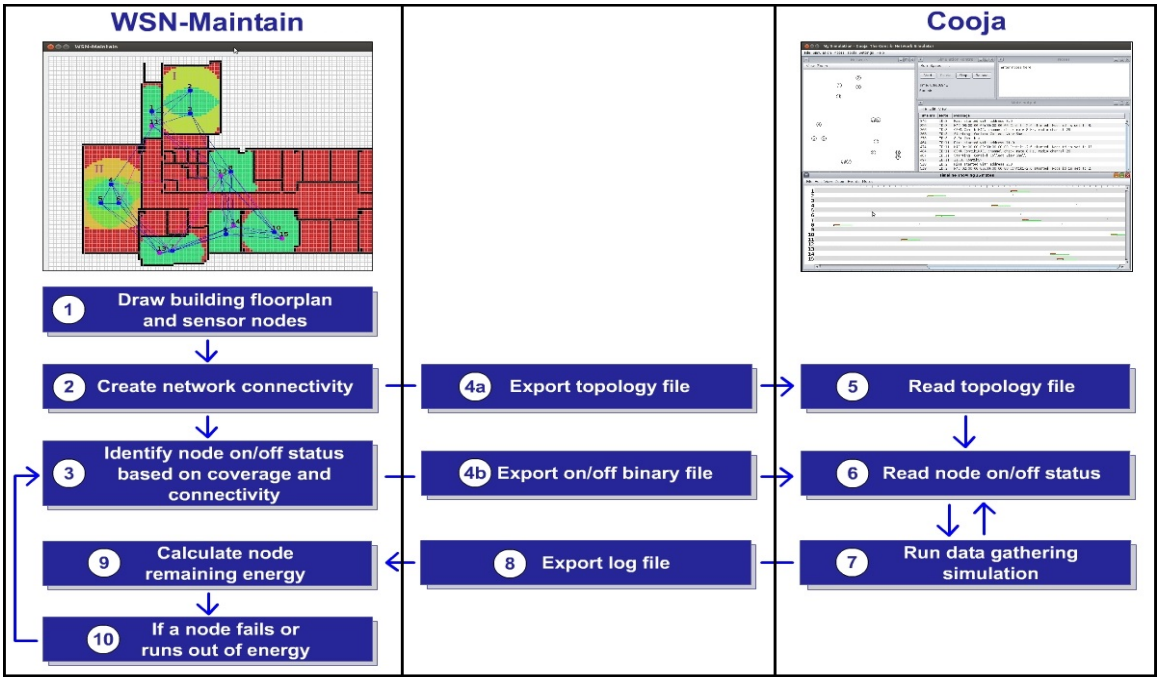


Figure 1. WSN-Maintain and Cooja framework.

To automatically deploy sensor nodes in a room, the boundary of the room must be identified. To identify the boundary, we create and scan rectangular grids, starting from the location on the canvas that is clicked by a user. The rectangular grids augment until they hit the obstacles. After the room is identified, sensor nodes are added one by one until the room is 100% covered. The first sensor node is deployed at the user's clicked location, and the rest are greedily added to maximise the coverage of the room.

Create network connectivity: To calculate network connectivity, WSN-Maintain currently supports two types of radio mediums, *i.e.*, the unit disk graph model (UDGM) and directed graph radio medium (DGRM). With UDGM, the topology file contains node locations, transmission range, interference range, transmission success ratio and receive success ratio. As we follow the specified format of files from Cooja, we can only create a network with homogeneous communication capabilities using UDGM. That is, all sensor nodes have the same range and receive success ratio. To create a heterogeneous network, users may opt for DGRM. With DGRM, WSN-Maintain can either automatically calculate received signal strength and receive success ratio for every pair of sensor nodes using the formula specified in [9,10] or take real link quality traces as a topology file.

The received signal strength (RSS) at a distance d ($P_r(d)$) is:

$$P_r(d) = P_t - PL(d) \tag{1}$$

where P_t is the output power of the transmitter and $PL(d)$ is the path loss. Table 1 shows the Tmote Sky output power configuration for CC2420 [7], an IEEE 802.15.4-compatible radio.

The in-building radio propagation model that takes into account the effects of obstacles (walls) between the transmitter and the receiver on the same floor is given by [10]:

$$PL(d) = PL(d_0) + 10n \log_{10}(\frac{d}{d_0}) + \sum WAF \tag{2}$$

where d is the transmitter-receiver distance, d_0 a reference distance, n the path loss exponent (the rate at which the signal decays), and WAF is the *wall attenuation factor*. Based on the measurement in [11], WAF is chosen to be 3.1 dBm. Table 2 shows the real signal propagation measurements conducted using Tmote Sky with the maximum power level (P_t = 0 dBm) in line-of-sight (LOS) and non-line-of-sight (NLOS) conditions reported in [12].

Table 1. Tmote Sky output power configuration for the CC2420 [7].

Power Level	Output Power (dBm)	Current Consumption (mA)
3	−25	8.5
7	−15	9.9
11	−10	11.2
15	−7	12.5
19	−5	13.9
23	−3	15.2
27	−1	16.5
31	0	17.4

Table 2. Tmote Sky signal propagation measurements [12].

Parameters	LOS Value	NLOS Value
Reference distance (d_0)	1.5 m	10 m
Path loss at d_0($PL(d_0)$)	40 dB	83 dB
Path loss exponent (n)	1.9	6.1
Wall attenuation factor (WAF)	-	3.1 dBm

The packet reception rate (PRR) at a distance d ($p(d)$) is given by:

$$p(d) = (1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}})^{8f} \tag{3}$$

where f is the frame size, which is 128 bytes for Tmote Sky's CC2420. $\gamma(d)$ is the signal-to-noise ratio (SNR) at a distance d and is given by:

$$\gamma(d) = P_t - PL(d) - P_n \tag{4}$$

P_n is the noise floor, which is given by [10]:

$$P_n = (F+1)kT_0B \tag{5}$$

F is the noise figure, k the Boltzmann's constant, T_0 the ambient temperature and B the equivalent noise bandwidth. The values for these parameters for Tmote Sky's CC2420 are given in Table 3 [13]. One can change all of these parameters according to the type of hardware used.

Table 3. CC2420 noise floor parameters [13].

Parameters	Default Value
Noise figure (F)	15.3 dB
Boltzmann's constant (k)	1.38×10^{-23}
Ambient temperature (T_0)	290 K
Equivalent bandwidth (B)	2×10^6 Hz

Identify node on/off status based on coverage and connectivity: To create a coverage map of the building, WSN-Maintain currently supports two types of coverage models, *i.e.*, the binary detection

model and Elfes's model [14]. These models can be used to approximate, for example, passive infrared (PIR) sensors. While at the moment, only these two models are implemented in WSN-Maintain, one can add other models based on their own requirements.

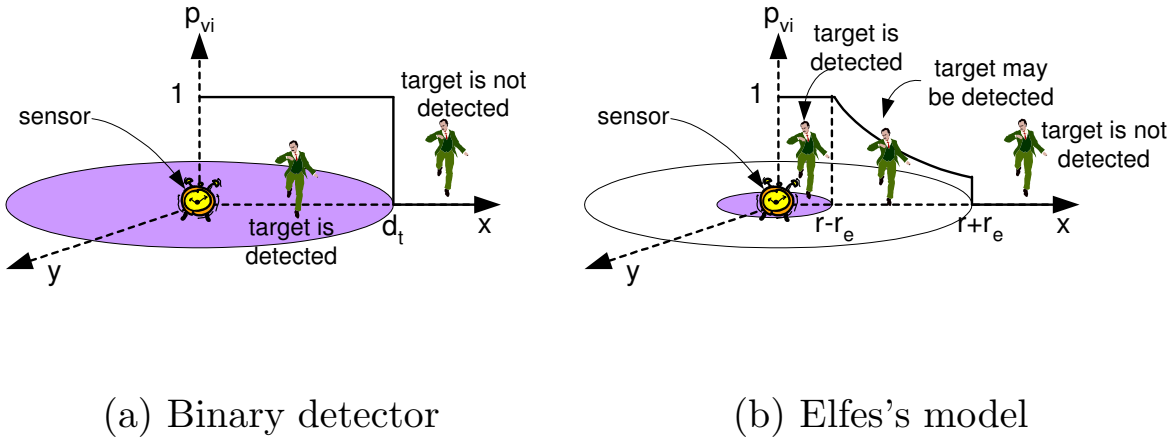
With the binary model, the probability p_{vi} that a sensor node i senses a point v in a floor plan is one if the distance d_{vi} between v and i is less than or equal to a threshold distance d_t, *i.e.*, the node's sensing range (Figure 2a).

$$p_{vi} = \begin{cases} 1 & \text{if } d_{vi} \leq d_t \\ 0 & \text{if } d_{vi} > d_t \end{cases} \tag{6}$$

In Elfes's model, as depicted in Figure 2b, the probability that sensor i detects a target on grid point v is:

$$p_{vi} = \begin{cases} 1 & \text{if } r - r_e \geq d_{vi} \\ e^{-\lambda a^{\beta}} & \text{if } r_e > |r - d_{vi}| \\ 0 & \text{if } d_{vi} \geq r + r_e \end{cases} \tag{7}$$

where r is the sensing range, r_e ($r_e < r$) is a measure of uncertainty in sensor detection, λ and β are parameters that model different sensor characteristics and $a = d_{vi} - r + r_e$. The parameters r, r_e, λ and β are adjusted according to the physical properties of the sensor. In particular, r and r_e affect the threshold distances of target detection. In [15], $\lambda = 0.2$, and $\beta = 0.6$. We approximate $r_e = 0.25 \times r$ and set the probability threshold to 0.9, *i.e.*, a point cannot be sensed if its probability is less than this threshold.



(a) Binary detector (b) Elfes's model

Figure 2. Sensor detection models [15]. (**a**) Binary detector. (**b**) Elfes's model.

In this work, we assume that sensor nodes cannot sense through walls and doors. We also assume that users may require different coverage degrees for special rooms, such as a server room and an electrical room. WSN-Maintain allows users to do this by clicking a point inside a room and specifying the coverage requirement. The total coverage percentage is calculated as the ratio of the total area satisfying the coverage requirements to the total interior area of the building.

Export topology file, export on/off binary file, read topology file and read node on/off status: WSN-Maintain communicates with Cooja by exporting a topology file specifying network connectivity and a binary file specifying the on/off status of nodes. Note that with UDGM, the topology file specifies transmission range and interference range in meters, as well as the transmission success ratio and the receive success ratio in the range of [0, 1]. With DGRM, the topology file contains received signal strength in dB and received success ratio in the range of [0, 1] for every pair of sensor nodes.

Run data gathering simulation: WSN-Maintain is run in parallel with the collect-view tool of Contiki, which was integrated into the Cooja simulator, to gather data from the WSN.

Export log file and calculate node remaining energy: Cooja communicates with WSN-Maintain through its log file. WSN-Maintain reads the log file to calculate the nodes' remaining energy. In [16], a linear model is used to estimate sensor node energy consumption. The total energy consumption E is defined as:

$$\frac{E}{V} = I_m t_m + I_l t_l + I_t t_t + I_r t_r + \sum_i I_{c_i} t_{c_i} \quad (8)$$

where V is the supply voltage, I_m the current draw of the microprocessor when running, t_m the time in which the microprocessor has been running, I_l and t_l the current draw and the time of the microprocessor in low power mode, I_t and t_t the current draw and the time of the communication device in transmit mode, I_r and t_r the current draw and time of the communication device in receive mode and I_{c_i} and t_{c_i} the current draw and time of other components, such as sensors and LEDs. Table 4 shows the energy consumption parameters used. The current draw of the communication device in transmit mode (I_t) for different power level has been shown in Table 1. Time measurement is implemented using the on-chip timers of the MSP430. The 32,768-Hz clock divided by eight is used, producing 4096 clock ticks per second.

Table 4. Energy consumption parameters [1,7].

Parameters	Default Value
Voltage (V)	3 V
Current draw of microprocessor when running (I_m)	1.8 mA
Current draw of microprocessor in low power mode (I_l)	0.0545 mA
Current draw of communication device in receive mode (I_r)	19.7 mA

If a node fails or runs out of energy: When a node dies, the coverage percentage decreases or the network disconnects. If the coverage percentage is below a threshold or the connectivity is insufficient, WSN-Maintain will either automatically find and turn on non-overlapping redundant nodes to maintain the coverage and connectivity requirement or report failures that require physical maintenance. Users can interactively use WSN-Maintain to simulate node failures by killing nodes or just wait until a node runs out of energy to see the results of WSN restoration.

4. Implementation

To avoid costly manual interventions, we try to extend the network lifetime as long as possible by putting coverage redundant nodes to sleep to conserve energy. These nodes will then be automatically turned on to cover some part of an uncovered area if active nodes that sense this area fail or run out of energy. In this section, we will show an example of how WSN-Maintain runs with Cooja to collect data, and we will discuss two use cases where we implement centralised and localised algorithms to find coverage redundant nodes. To further improve the energy efficiency, we incorporate the concept of reading correlation by putting more nodes to sleep if they have high reading correlations. We will make the tool available via our website at publication time [17].

4.1. An Example

WSN-Maintain is implemented using the Java programming language. Figure 3 shows a snapshot of the implementation of WSN-Maintain and Cooja that simulates a data-gathering application. In the floor plan shown, 15 sensor nodes are placed arbitrarily in seven rooms, where Node Numbers 11 to 15 are coverage redundant nodes. All rooms have a coverage requirement equal to one, except Rooms I and II have requirements equal to two and three, respectively. WSN-Maintain visualises the coverage map with colours ranging from red to green. The red area (darker colour if the figure is in grayscale) means the coverage requirement is not satisfied at all, while green (lighter colour in grayscale) means the requirement is satisfied. For example, when a user turns Node 8 off, WSN-Maintain will turn on redundant Node 12 to maintain coverage. These changes will be informed to Cooja through a file that

contains the on/off status of all nodes. Cooja will act accordingly by turning off Node 8 and turning on Node 12.

4.2. Use Cases

We implemented two redundant node algorithms in WSN-Maintain with the aim to extend the network lifetime while still preserving coverage and connectivity. The two algorithms, *i.e.*, greedy-maintain, a centralised greedy algorithm, and local-maintain, a localised algorithm, identify coverage redundant nodes. A sensor node is coverage redundant if, when turned off, the quality of coverage does not fall below the coverage threshold and the network is not disconnected. Coverage redundant nodes can be put to sleep to save energy and will be turned on to cover some part of an uncovered area if active nodes that sense this area fail or run out of energy. The on/off status of nodes to turn on redundant nodes and the status of the remaining energy to inform other nodes when active nodes run out of energy are piggybacked on control messages. Both algorithms, with a slight modification, can be used to configure the initial network, as well as to turn on redundant nodes. To differentiate the two functionalities, we call the algorithms to configure the initial network the greedy initial network configuration and the local initial network configuration.

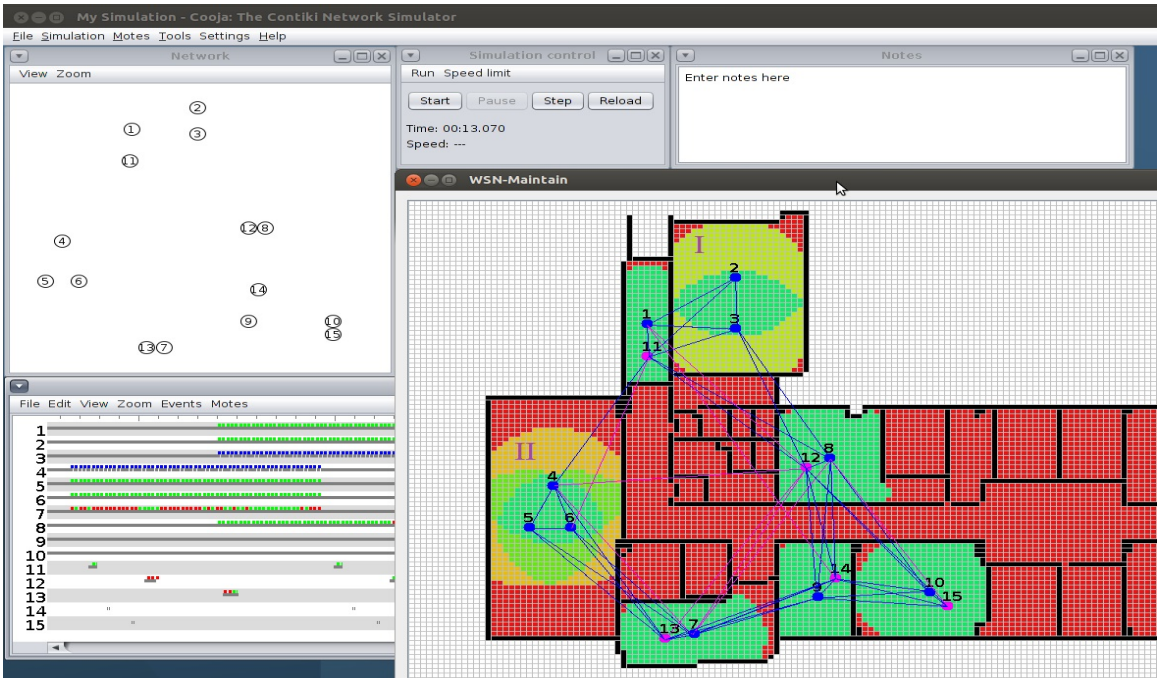


Figure 3. WSN-Maintain runs in parallel with Cooja.

4.2.1. Use Case I: Greedy-Maintain

In the centralised greedy algorithm, the sink/base station has global information about the network, the layout of the building and the obstacles, such as walls and doors. At the moment, we assume a single sink scenario. However, implementing multiple sink cases would not be a big challenge. One approach is by choosing a supersink that has a connection to the other sinks. By doing this, we reduce the problem of multiple sinks to the problem of a single sink.

In the greedy initial network configuration, the computation is performed centrally, room by room, starting from a room that has the closest node to the sink/base station. The algorithm identifies nodes that are required to be active in order to satisfy the coverage and connectivity requirements for each room, while the rest are deemed as redundant. The pseudocode for the greedy initial network configuration is given in Algorithm 1. It takes as input the set V of sensor nodes that are initially off

and the local coverage threshold, *i.e.*, a room's required coverage. In WSN-Maintain, V is implemented as a list of objects. Each $v \in V$ stores the following information: ID, position_x, position_y, status, room_ID, power_level, sense_range, initial_energy, remaining_energy and a set of neighbours. These variables are self-explanatory.

The procedure is repeated until all rooms are processed. In each iteration, to select a room to be processed, we firstly find a node v whose status is off and has the smallest distance to an active node so as to guarantee network connectivity. Initially, only the sink is active, so the first room to be processed must be the room where the sink is or the nearest room if the sink's room has no other nodes. The room that needs to be processed is the room where v is. We then need to activate nodes in that room until the room's local coverage satisfies the local coverage threshold. For these purposes, we greedily find an inactive node in that room that has the largest contribution to the room's local coverage, has the highest remaining energy and needs the smallest number of active relay nodes. The room's local coverage (*current_local_coverage*) is calculated as the percentage of area in the room that is covered by active sensor nodes. Relay nodes are sensor nodes that need to be turned on for network connectivity; otherwise, the active network is disconnected. To find how many relay nodes are needed by each inactive node, we build a spanning tree using breadth first search. The complexity of this algorithm with breadth first search ($O(|E|)$) is $O(|V||E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of links.

Algorithm 1 Greedy initial network configuration.

Require: V, *local_coverage_threshold*
Ensure: *status* for all $v \in V$
  while all rooms are not processed **do**
    Find $v \in V$ that satisfies:
    (1) $v.status =$ off
    (2) v has the smallest distance to an active node
    $i \leftarrow v.room_ID$
    while $i.current_local_coverage < local_coverage_threshold$ **do**
      Find $v \in V$ that satisfies:
      (1) $v.room_ID = i$
      (2) $v.status =$ off
      (3) v has the largest contribution to $i.current_local_coverage$
      (4) v has the highest *remaining_energy*
      (5) v needs the smallest number of active relays
      $v.status \leftarrow$ on
      Calculate $i.current_local_coverage$
    end while
    Turn on all necessary relays for connectivity
  end while

The pseudocode for greedy-maintain is given in Algorithm 2. It takes as input the set V of sensor nodes, a dead node x and the local coverage threshold. V is implemented as a list of objects, where an object $x \in V$ has ID, position_x, position_y, status, room_ID, power_level, sense_range, initial_energy, remaining_energy and a set of neighbours.

When x dies, the room's local coverage may fall below the local coverage threshold, and the network may be disconnected. Therefore, in order to satisfy the coverage and connectivity requirements for the room where x is located, greedy-maintain finds nodes that are required to be active. The complexity of this algorithm with breadth first search to build a spanning tree for connectivity is $O(|V| + |E|)$.

Algorithm 2 Greedy-maintain.

Require: V, dead_node x, *local_coverage_threshold*
Ensure: *status* for all $v \in V$
  while $x.room_ID.current_local_coverage < local_coverage_threshold$ **do**
    Find $v \in V$ that satisfies:
    (1) $v.room_ID = x.room_ID$
    (2) $v.status =$ off
    (3) $d(x,v) < x.sense_range + v.sense_range$
    (4) v has the largest contribution to $x.room_ID.current_local_coverage$
    (5) v has the highest *remaining_energy*
    $v.status \leftarrow$ on
    Calculate $x.room_ID.current_local_coverage$
  end while
  Turn on all necessary relays for connectivity

4.2.2. Use Case II: Local-Maintain

In the localised algorithm, each node only knows the locations of its direct neighbours, their sensing range and whether they are in the same room or not. Nodes do not have knowledge regarding the layout of the building and obstacles (walls and doors). Each node decides locally to be active or redundant based on its priority. The complexity of the localised algorithm is $O(|V|)$.

The local computation starts from a node that has the highest priority. Priority is computed locally. In the local initial network configuration (Algorithm 3), it is based on the node's remaining energy and the distance to a sink, while in local-maintain (Algorithm 4), it is based on the node's remaining energy and the distance to a dead node. In the priority calculation, α and β are weights that are assigned to the residual energy and the distance. In this paper, we set $\alpha = 0.5$ and $\beta = 0.5$.

Algorithm 3 Local initial network configuration.

Require: V, *local_coverage_threshold*
Ensure: *status* for all $v \in V$
  for all $v \in V$ **do**
    $v.priority \leftarrow 1 / \left[1 - \left(\alpha \times \frac{v.remaining_energy}{v.initial_energy}\right) - \left(\beta \times \frac{1}{d(sink,v)}\right)\right]$
  end for
  for all $v \in V$ in decreasing order of *priority* **do**
    Calculate $v.overlap_coverage \leftarrow$ percentage of v's coverage area that is overlap with every w's coverage area for all $w \in v.neighbour$ that satisfies:
    (1) $w.room_ID = v.room_ID$
    (2) $d(v,w) < v.sense_range + w.sense_range$
    if $v.overlap_coverage < local_coverage_threshold$ **then**
      $v.status \leftarrow$ on
    end if
  end for

A node v locally decides to be active if the overlap coverage is below the local coverage threshold. v's overlap coverage is calculated as the percentage of v's coverage area that is overlapped with its neighbours' coverage area. To preserve connectivity, a node also needs to check if it is connected to any other active nodes or not. To compute connectivity locally, we utilise the Span method [18]. Span is a decentralised algorithm to activate extra nodes for one-connectivity. With Span, a node decides to become active if two of its neighbours cannot reach each other, either directly or via one or two active nodes.

In both Algorithms 3 and 4, V is implemented as a list of objects, where an object $v \in V$ or $x \in V$ has ID, position_x, position_y, status, room_ID, power_level, sense_range, initial_energy, remaining_energy, a set of neighbours, priority and overlap_coverage. The set of neighbours is implemented the same as V.

Algorithm 4 Local-maintain.

Require: V, dead_node x, $local_coverage_threshold$
Ensure: $status$ for all $v \in x.neighbour$
  $A \leftarrow \emptyset$
  for all $v \in x.neighbour$ that satisfies:
  (1) $v.room_ID = x.room_ID$
  (2) $d(x,v) < x.sense_range + v.sense_range$ **do**
    $v.priority \leftarrow 1/[1 - (\alpha \times \frac{v.remaining_energy}{v.initial_energy}) - (\beta \times \frac{x.sense_range + v.sense_range - d(x,v)}{2 \times x.sense_range})]$
    $A \leftarrow A \cup \{v\}$
  end for
  for all $v \in A$ in decreasing order of $priority$ **do**
    Calculate $v.overlap_coverage \leftarrow$ percentage of v's coverage area that is overlap with every w's coverage area for all $w \in v.neighbour$ that satisfies:
    (1) $w.room_ID = v.room_ID$
    (2) $d(x,w) < x.sense_range + w.sense_range$
    if $v.overlap_coverage < local_coverage_threshold$ **then**
      $v.status \leftarrow$ on
    end if
  end for

The four algorithms try to achieve the desired coverage by turning on some nodes in the network. If no nodes satisfy the coverage and connectivity requirements when the network operates, the quality of coverage decreases over time, as will be shown in the simulation results in Section 6. WSN-Maintain then notifies users by reporting failures that require physical maintenance. If no nodes satisfy the requirements at all during the network set-up, users will also be notified, as no nodes are active and no data are gathered.

4.2.3. Correlation

In WSNs, sensor nodes are often densely deployed to cover a monitoring area. This redundant deployment causes spatial correlation among the readings of nearby nodes. In this paper, we define reading correlation as spatial correlation, where the reading of a node may be predicted from the readings of its nearby neighbours with high confidence.

We incorporate the concept of reading correlation using the inverse distance weighting interpolation method, such as used in [19]. The inverse distance weighting interpolation method is based on the close similarity principles: the shorter the distance between two things is, the more similar they are. In a building scenario, two nearby nodes may not have similar readings if they are in two different rooms. Therefore, to calculate the reading correlation of a sensor node in a room, we only take into account nearby nodes in the same room.

By using the inverse distance weighting interpolation method, the unknown measure at a position can be interpolated by means of a weighted average of the measures associated at the known points. It is achieved by the following function:

$$\hat{V}_i^t = \frac{\sum_{j=1, j\neq i}^{n} \omega(i,j) V_j^t}{\sum_{j=1, j\neq i}^{n} \omega(i,j)} \tag{9}$$

$\hat{V}_i^t$ stands for the interpolated value of node i at time t; node j belongs to the set Z of adjacent nodes; $\omega(i,j)$ is the distance weight between node i and j. If $d(i,j)$ is the distance between node i and j, the weight can be expressed as follows:

$$\omega(i,j) = \frac{1}{d(i,j)^m} \tag{10}$$

where m is a positive real number, called the power parameter. The absolute difference between the predicted value $\hat{V}_i^t$ and real V_i^t is the error ϵ called the redundancy of a node:

$$\epsilon = |\hat{V}_i^t - V_i^t| \tag{11}$$

By using the correlation method, sensors that have a high reading correlation can be considered as redundant and can be put to sleep. Nodes decide whether they are redundant or not based on ϵ. Sensors that have high reading correlations are sensors whose data reading can be predicted using the reading of nearby active sensors in the same room. With this method, we expect to improve the energy efficiency by putting more nodes to sleep.

5. Simulation Set-Up

In the WSN-Maintain simulation with Cooja, we use datasets from five sensor network deployments: a data centre room at the Informatics Institute at Federal Fluminense University in 2013 (TMON) [20], the ground floor and first floor of the Nimbus building at the Cork Institute of Technology in 2013 (CIT-1 and CIT-G) and the second floor and third floor of the Strata Conference building in New York that was held in 2012 (NY-2 and NY-3) [21]. We chose these deployments as we require dense networks for our simulation, where some rooms must have multiple sensor nodes, as to be expected in large buildings. Unless otherwise specified, the five datasets that are used in our simulation consist of temperature sensor reading for 24-hour periods. The floor plan of the buildings and the sensor node locations that are used in the simulation are depicted in Figures 4–8.

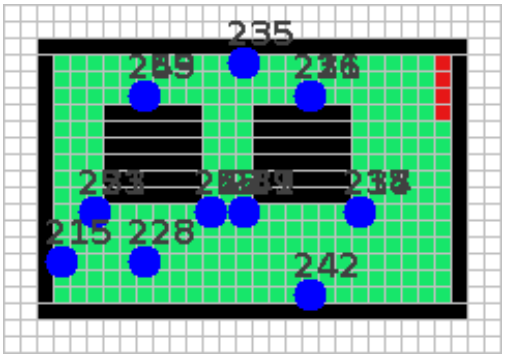


Figure 4. WSN deployment in a data centre room at the Informatics Institute at Federal Fluminense University (TMON).

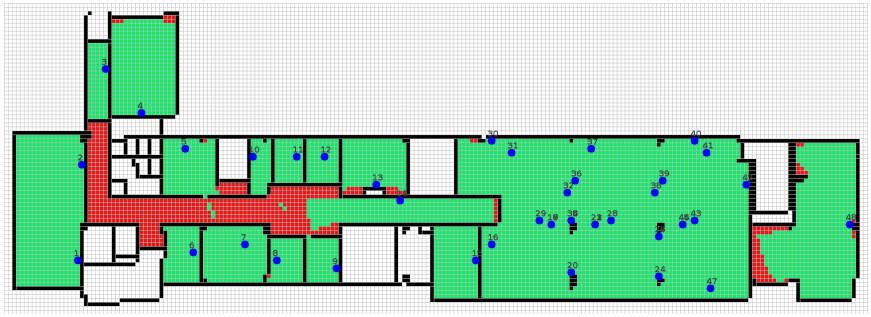


Figure 5. WSN deployment on the first floor of the Cork Institute of Technology (CIT-1).

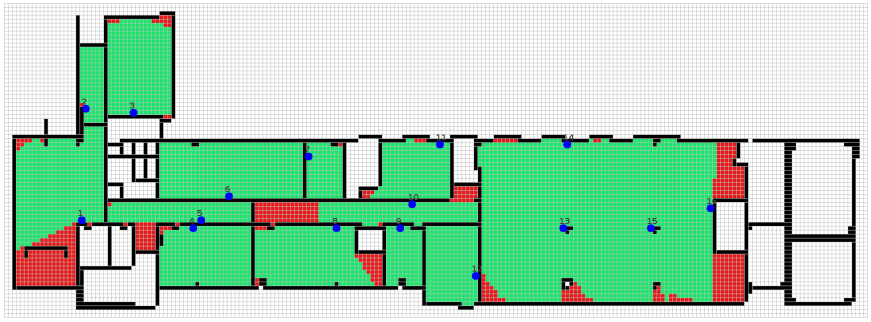


Figure 6. WSN deployment on the ground floor of the Cork Institute of Technology (CIT-G).

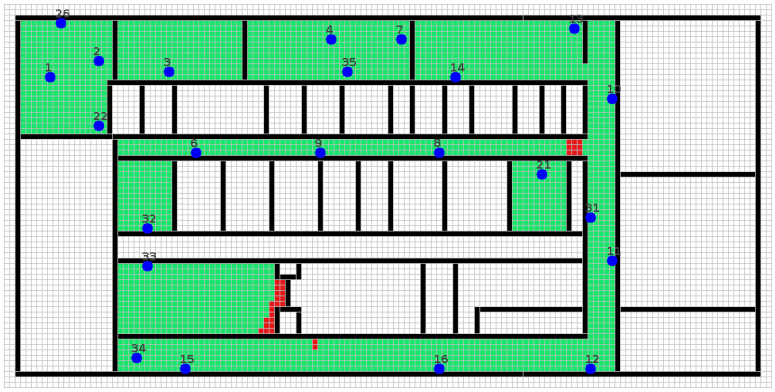


Figure 7. WSN deployment on the second floor of the Strata Conference in New York (NY-2).

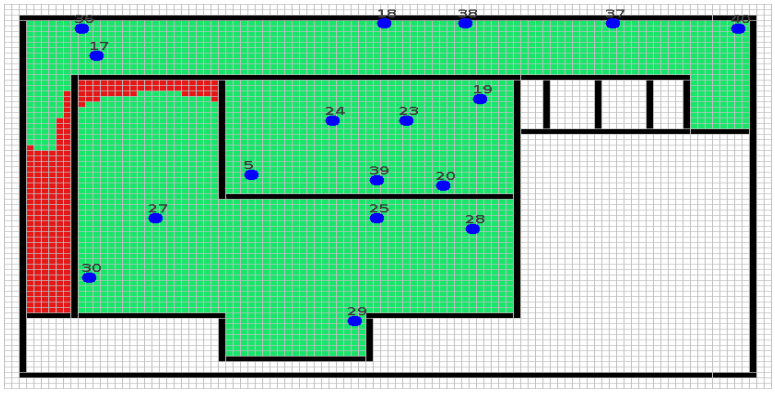


Figure 8. WSN deployment on the third floor of the Strata Conference in New York (NY-3).

The information for the five WSN deployments is summarised in Table 5. These deployments have different numbers of sensor nodes. Note that some rooms do not have sensor nodes, and so, they are not considered in the performance calculation. The simulation parameters are given in Table 6. We assume all sensor nodes communicate using the highest power level. For Tmote Sky's CC2420 radio, the output power is 0 dBm (Table 1). To have sufficient sensing coverage, we use 7.5 m as the sensing range for all sensor nodes in the CIT-1, CIT-G, NY-2 and NY-3 deployments. However, for the one-room TMON deployment, a 2.5-metre sensing range is sufficient to have the room covered. TMON needs higher accuracy because of the nature of the application, *i.e.*, data centre monitoring. We also assume that all sensor nodes have the same initial energy. We choose a small amount of energy, *i.e.*, one Joule for TMON, five Joules for CIT-1 and CIT-G and four Joules for NY-2 and NY-3. This is a pragmatic choice that allows us to run experiments that can produce results within practical time periods, since in reality, sensor networks are expected to have sufficient power for operation over several months.

Table 5. WSN deployment information.

WSN Deployment	Number of Nodes	Number of Rooms	Sink Identifier
TMON	22	1	215
CIT-1	48	14	32
CIT-G	16	13	13
NY-2	23	7	1
NY-3	17	3	30

For the network connectivity, we use directed graph radio medium because for an in-building scenario, we take into account the effects of obstacles (walls and doors) between the transmitter and the receiver. For the sake of simplicity, we use the binary detection model, and all rooms have the same coverage requirements, *i.e.*, $k = 1$. To have sufficient coverage redundancies, we use 75% local coverage threshold, as the deployments are not dense enough, except the TMON deployment, where we use 95%. For the reading correlation, a node is redundant if the difference between the predicted and its actual temperature readings is below 0.1 °C. This value is in the range of Tmote Sky's temperature sensor accuracy [22]. For two dimensions, we use the correlation's power parameter $m = 2$. Our simulation utilises the collection tree protocol (CTP) [23] implemented in Contiki and the duty-cycling Contiki MAC protocol [24].

Table 6. WSN-Maintain simulation parameters.

Simulation Parameters	Default Value
Transmit power (P_t)	0 dBm
Sensing range (r)	
TMON, CIT-1, CIT-G, NY-2, NY-3	2.5 m, 7.5 m, 7.5 m, 7.5 m, 7.5 m
Initial energy	
TMON, CIT-1, CIT-G, NY-2, NY-3	1 J, 5 J, 5 J, 4 J, 4 J
Connectivity	Directed graph radio medium (DGRM)
Coverage	Binary detection model
Coverage requirement (k)	1
Local coverage threshold	
TMON, CIT-1, CIT-G, NY-2, NY-3	95%, 75%, 75%, 75%, 75%
Correlation's redundancy error (ϵ)	0.1 °C temperature
Correlation's power parameter (m)	2
Sensing and reporting interval	1 time unit
Routing protocol	Collection tree protocol (CTP) [23]
MAC protocol	Contiki MAC [24]

6. Performance Evaluation

In this section, we show how our tool maintains coverage and improves network lifetime. We use the following metrics in the evaluation:

1. The global quality of coverage (QoC) over time shows the overall percentage of area in a floor that is covered by active sensor nodes. In this case, we only take into account rooms that have sensors.
2. The increased lifetime of a network is the ratio of the lifetime with a redundant node algorithm to when all nodes are active. The lifetime is the time until the global QoC cannot satisfy the local coverage threshold.
3. The percentage of active nodes over time and/or when the simulation finished.
4. The percentage of dead nodes over time and/or when the simulation finished.
5. The percentage of disconnected nodes when the simulation finished.

The results presented in this paper are based on one simulation run each, as we use trace files from the datasets, and so, the sensor samples do not change. The topologies of the networks are also fixed, based on the connectivity calculation using UDGM and DGRM in Section 3.

We will firstly present the simulation results for the one-room deployment, *i.e.*, the data centre room at the Informatics Institute at Federal Fluminense University [20]. The TMON deployment is very dense, and we expect to get a very good result in lifetime improvement. Secondly, we will present the results for multiple-room deployments, *i.e.*, the ground floor and first floor of the Nimbus building at Cork Institute of Technology and the second floor and third floor of the Strata Conference building in New York [21]. In these multiple-room deployments, even though the coverage redundancy among the deployed sensor nodes is smaller, we still expect to get improvement in network lifetime as some rooms have more than one sensor node. Finally, we will show the results of lifetime prediction if we use the real energy storage in AA batteries.

6.1. One Room

In the TMON deployment, there are 22 sensor nodes in one room. Figure 9 shows TMON's global quality of coverage. When all nodes are active, the network can only maintain the desired QoC with a 95% local coverage threshold for 150 time units. With greedy-maintain and local-maintain, the lifetime can be increased up to around eight times as their percentages of QoC do not fall below the threshold before 1195 time units. The increased lifetime is calculated by dividing the lifetime of the network with an algorithm by the lifetime when all nodes are active. The details for the lifetime improvement are summarised in Table 7, where greedy-maintain and local-maintain achieve 8.39 and 7.99 times lifetime improvement, respectively. Greedy-maintain achieves a longer lifetime compared to local-maintain, as it requires less nodes to be active to cover the room (see Figure 10). In cases where the percentage of active nodes increases in this figure, that is due to redundant nodes being turned on to repair the coverage when a node dies.

Using the dataset from TMON, there is only a slight additional improvement in the lifetime of the network when the reading correlation is utilised, *i.e.*, 8.43 times for greedy-maintain with correlation and 8.04 times for local-maintain with correlation. This happens as the temperature variations in the data centre room is very high. Figure 11 shows the percentage of dead nodes over time, where the localised algorithm has more dead nodes compared to the greedy one. In the TMON deployment, all nodes are within a one-hop distance from the sink. Therefore, when a node dies, no other nodes will be disconnected from the network. The percentages of active and dead nodes when the simulation finished are summarised in Table 7. Greedy-maintain with reading correlation has 45.45% of nodes still active when the simulation finished, but can only support 77.43% QoC. Local-maintain with reading correlation has 13.64% active nodes for 54.51% QoC.

The average execution time for each node with greedy-maintain, local-maintain, greedy-maintain with correlation and local-maintain with correlation are 0.0164, 0.0151, 0.0203 and 0.0197 time units, respectively.

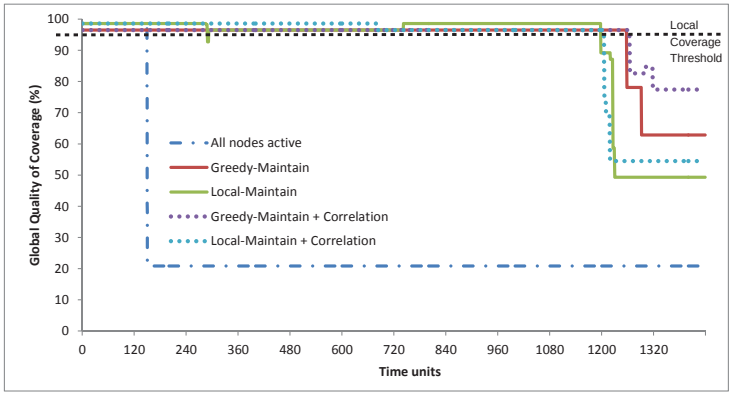


Figure 9. TMON's global quality of coverage.

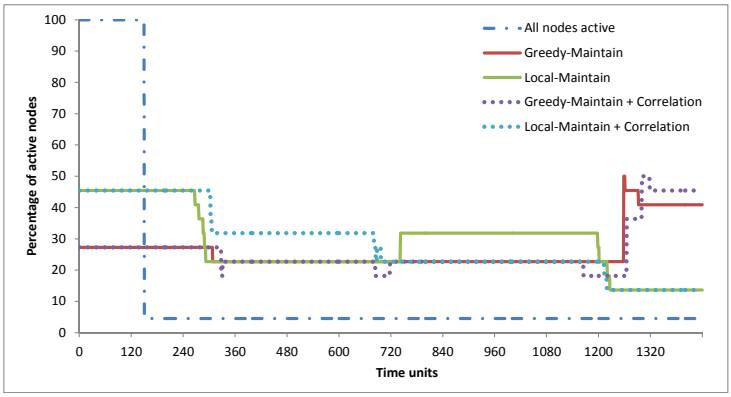


Figure 10. TMON's percentage of active nodes over time.

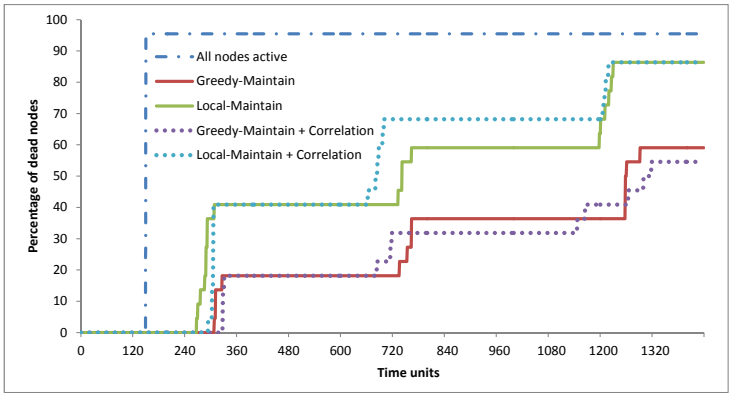


Figure 11. TMON's percentage of dead nodes over time.

Table 7. TMON's simulation results.

Algorithm	Lifetime (time units)	Increased Lifetime	Active Node	Dead Node	QoC
All nodes active	150	-	4.55%	95.45%	20.83%
Greedy-maintain	1258	8.39×	40.91%	59.09%	62.85%
Local-maintain	1198	7.99×	13.64%	86.36%	49.31%
Greedy-maintain + correlation	1265	8.43×	45.45%	54.55%	77.43%
Local-maintain + correlation	1206	8.04×	13.64%	86.36%	54.51%

6.2. Multiple Rooms

In the multiple-room evaluation, we use the datasets from the other four deployments, *i.e.*, the ground floor and first floor of the Nimbus building at the Cork Institute of Technology and the second floor and third floor of the Strata Conference building in New York that was held in 2012 [21]. In this paper, we only show the detailed simulation results for CIT-1, as the results from the other deployments support the conclusions we draw. Figures 12–14 show CIT-1's global QoC, the percentage of active nodes over time and the percentage of dead nodes over time, respectively. These results have similar trends with the TMON deployment in Section 6.1, where the greedy algorithm performs better than the localised algorithm and reading correlation improves the results. The summary of CIT-1's simulation results is presented in Table 8. When the simulation finished, nodes that are not active, dead or disconnected are redundant nodes, *i.e.*, nodes that are inactive throughout the simulation.

Table 8. CIT-1's simulation results.

Algorithm	Lifetime (time units)	Increased Lifetime	Active Node	Dead Node	Disconnected Node	QoC
All nodes active	597	-	2.08%	68.75%	29.17%	0%
Greedy-maintain	919	1.54×	4.17%	10.42%	22.92%	34.67%
Local-maintain	881	1.48×	8.33%	31.25%	25%	36.93%
Greedy-maintain + correlation	935	1.57×	6.25%	10.42%	22.92%	35.75%
Local-maintain + correlation	928	1.55×	10.42%	31.25%	25%	35.83%

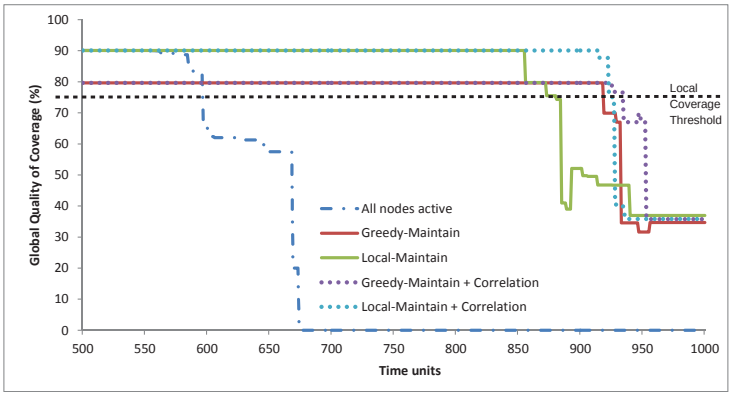


Figure 12. CIT-1's global quality of coverage.

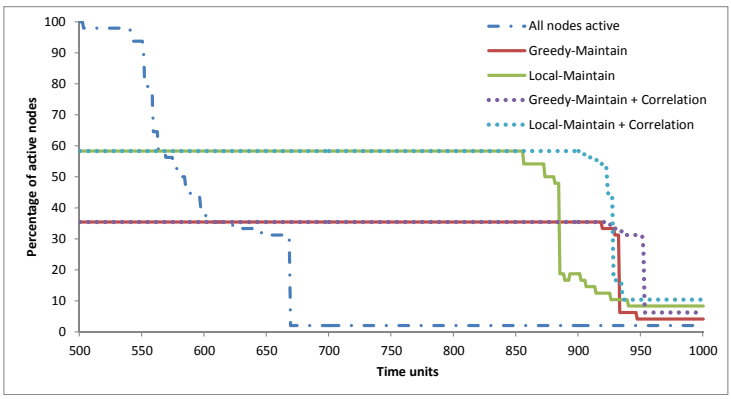


Figure 13. CIT-1's percentage of active nodes over time.

The average execution time for each node with greedy-maintain, local-maintain, greedy-maintain with correlation and local-maintain with correlation are 0.0377, 0.0064, 0.0452 and 0.0080 time units, respectively.

For the other deployments, we only present the lifetime improvement in Figure 15, where the text next to each bar shows how much the lifetime was extended in time units. This figure shows that the lifetime of the network increases in all cases. When the reading correlation method is used, the lifetime improves, except in NY-2 with greedy-maintain. This happens because some nodes near the sink are turned off due to reading correlation, and so, active nodes near the sink need to forward more packets from their children to the sink. This depletes their battery quickly and disconnects the children from the sink. If more nodes near the sink are active, the workload to forward packets to the sink can be shared so as to balance the traffic and reduce the energy consumption of some nodes. The lesson learned from this experiment is that in selecting nodes to be inactive, one needs to be cognisant of the impact on message routing. We do not discuss this issue in this paper. However, one can refer to [25] for a possible solution for identifying critical nodes for network connectivity.

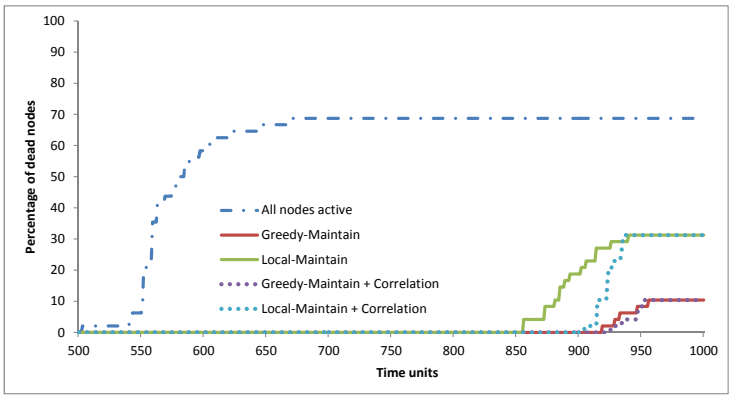


Figure 14. CIT-1's percentage of dead nodes over time.

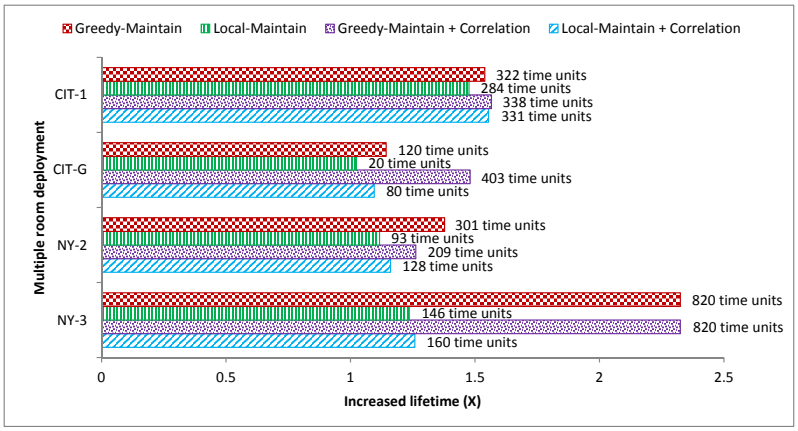


Figure 15. Increased lifetime for deployments with multiple rooms.

6.3. Behaviour with Higher Initial Energy

Due to the excessive simulation time when using a realistic battery budget and network lifetime that increases almost linearly [26], we predict the lifetime improvement for TMON and CIT-1 using the linear interpolation method. For TMON, we simulate using 0.5 and 1 Joules of initial energy. For CIT-1, we use 2.5 and 5 Joules. Based on the simulation results with these small amount of energy, we

predict the lifetime of the network using the real energy storage, *i.e.*, around 10,000 Joules for two AA batteries.

The increased lifetime for the two deployments is illustrated in Figure 16. The text next to each bar in Figure 16 shows how much the lifetime was extended in time units. While the one-hop TMON deployment achieves up to around 9× for greedy-maintain with reading correlation, the CIT-1 deployment only results in around a 1.5× lifetime improvement. This happens because CIT-1 has smaller coverage redundancy among the nodes, and there are two bridge nodes, *i.e.*, Nodes 13 and 14 in Figure 5, that can disconnect the left part of the network when they fail. The lesson learned from this experiment is that it is necessary to identify connectivity critical nodes by taking message paths into account. After identifying the critical nodes, such as Nodes 13 and 14 in Figure 5, one can deploy additional nodes as backups. Therefore, when critical nodes die, the network is still connected [25]. WSN-Maintain can be extended to incorporate this issue. It can then be used to re-deploy sensor nodes to achieve better coverage and better lifetime, given a coverage threshold and an expected lifetime.

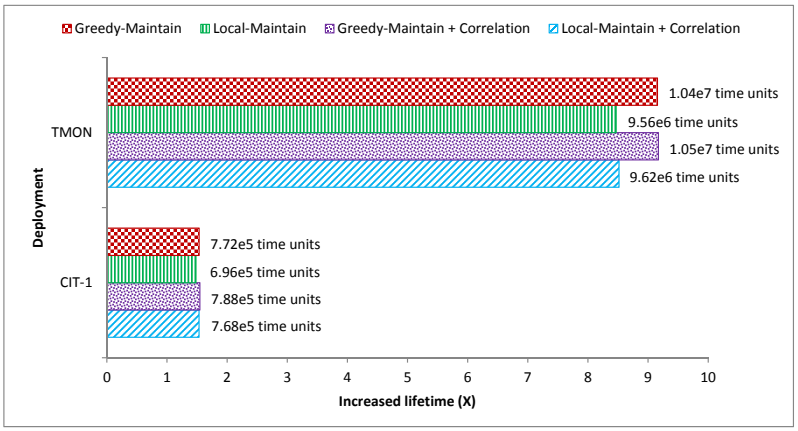


Figure 16. Increased lifetime for TMON and CIT-1 deployments using real energy storage in AA batteries.

7. Related Work

Currently in the literature, there does not exist a tool to manage and maintain coverage in a deployed sensor network. However, in this section, we try to review publications that relate to maintenance efficiency in wireless sensor networks.

Wireless sensor networks are prone to node failures, for instance caused by battery depletion, damaged hardware and buggy software. To be able to maintain efficient operations and to prolong the lifetime of the network, a sensor network should be resilient to these dynamics and able to automatically reconfigure itself. In some applications, physical node access costs are assumed to be dominant over *in situ* costs during WSN maintenance. This means the maintenance cost for replacing one sensor (or its batteries) will be approximately the same as the maintenance cost for replacing all sensors in a region. In this case, one may try to balance the energy consumption of the whole network, so as to extend the time intervals between replacement of one or more nodes/batteries in the field. Barroso *et al.* [27] modify the Greedy Perimeter Stateless Routing (GPSR) protocol for these purposes. In the modification of GPSR, a message is not necessarily forwarded to a neighbour that is the closest one to the destination. Instead, the message is randomly delivered to any node closer to the destination. This scheme proves to achieve better maintenance efficiency, which depends on the number of messages sent. However, using this scheme, messages travel longer paths, which result in increased latency.

A failed node can create coverage and connectivity holes in a sensor network. In order to maintain network effectiveness, failed nodes should be replaced. There are three policies for failed

node replacement by repairing network holes created by failed nodes [28]. The first policy is the directed furthest node first (DFNF), where an active node repairs the hole using one of its deactivated neighbours that has the longest distance, but in the same direction to the hole. In the second policy, weighted directed furthest node first (WDFNF) selects replacements by considering both the distance and direction of a node. The third one is best fit node (BFN), where all active nodes adjacent to a coverage hole participate in the replacement procedure. These three policies need strong assumptions regarding the network properties: for the unit disk graph model, the transmission range is twice the sensing range, and nodes can measure distances and angles to other nodes. It has been shown that DFNF and WDFNF can achieve similar lifetime to BFN when the network is extremely dense. Extremely dense here means an active node has around 20 inactive neighbours, which is not realistic in a real deployment.

Under the condition that the communication range $\geq 2 \times$ the sensing range, a sensor network only needs to be configured to guarantee coverage in order to satisfy both k-covered and k-connected network, where $k \geq 1$ [29]. A decentralised algorithm called the Coverage Configuration Protocol (CCP) has been proposed to evaluate a node's eligibility to become active if one of the intersection points of its neighbours' sensing circles are not k-covered. This algorithm requires the information about locations of all sensing neighbours, which are gathered using beacons (*i.e.*, HELLO messages) received from the neighbours. When the ratio of the communication range to the sensing range <2, CCP guarantees k-coverage, but nodes might not communicate with each other. In this case, Span [18] can be utilised. Span is a decentralised algorithm to activate extra nodes for one-connectivity. With Span, a node decides to become active if two of its neighbours cannot reach each other either directly or via one or two active nodes.

An in-field maintenance framework for VigilNet has been proposed [30]. VigilNet is a surveillance application. In this framework, users are allowed to define the coherency requirements of the states of protocols, specify the maintenance/repair policy (*i.e.*, when to self-heal the system and who invokes the maintenance service) and define which set of protocols need maintenance services in order to satisfy a system performance requirement and the dependency constraints among the services, according to the system specification and constraints. For example, time synchronisation repair is performed once per day; sensing coverage repair is performed locally if the coverage $<100\%$; while communication repair is performed at the base station if the connectivity $<100\%$. During the communication repair, the base station computes the connectivity graph based on the connectivity information reported from the nodes. Different from the original VigilNet system [31], where all of the protocols are reinitialised once per day without considering if a protocol needs the reinitialisation, the proposed framework allows different maintenance periods for different maintenance services and, thus, saves up to 5% of a node's battery capacity.

Li *et al.* [32] propose a dependency constraint-directed self-healing framework to allow users to compose self-healing services. The framework includes three components: health monitoring, self-healing policy and self-healing engine. The health monitor is implemented to collect the health state information at system execution time, such as battery level, link quality and end-to-end delay. The self-healing policy, which is usually specified by the application designer, is not discussed in this paper. The self-healing engine includes a set of run-time invocation routines embedded in the main loop of the application.

When an automated reconfiguration cannot heal the network, physical maintenance is required. In [33], Parikh *et al.* propose node-replacement policies to maintain a minimum threshold coverage and maximise the network lifetime using a fixed amount of replacement nodes. The policies determine if a failed node in the network is important enough to be replaced by calculating the weight of the failed node based on a specific parameter. If the weight of a failed node is greater than the policy threshold, the failed node is replaced. Otherwise, it is ignored. The parameters are the cumulative reduction in area of sensing coverage, the energy increase per node, the local reduction in area of the sensing coverage and the hybrid of the three parameters with different weights. With up to 10% coverage

degradation, the results show that the hybrid policy improves the network lifetime by up to 90% compared to the no-replacement policy and up to 23% compared to the simple first-fail-first-replace policy.

The betweenness centrality of a node is the sum of the probability that the node falls on a randomly-selected shortest path between all pairs of nodes in a network. According to Kas *et al.* in [34], when nodes with high betweenness centrality die, the number of hops between any two nodes in a network becomes greater. This happens because the high betweenness nodes are on a high fraction of shortest paths among other nodes in the network. The concept of betweenness centrality can be used for routing purposes, where routing paths can bypass nodes with high centrality to preserve their energy. For the maintenance point of view, the locations of nodes with high centrality can be used to deploy redundant nodes as backups. Therefore, when nodes with high centrality die, the redundant nodes can be activated to replace the dead nodes. The use of centrality to analyse network robustness and deploy relay nodes has been proposed in [25].

8. Conclusion

This paper presents WSN-Maintain, a Cooja-based tool to maintain coverage requirements and evaluate network lifetime. As use cases for this tool, we implement greedy-maintain and local-maintain to configure the initial network, as well as to turn on redundant nodes. We evaluate our tool using the two redundant node algorithms with datasets that consist of the temperature sensor reading from five real deployments. Our simulation results with Cooja show that we can extend the network lifetime as many as 9× for the one-room one-hop TMON deployment and 1.5× for the multiple-room multiple-hop CIT-1 deployment.

From the experiments, we learned that disconnected networks cause a shorter lifetime. Therefore, it is necessary to identify connectivity-critical nodes by taking message paths into account. This can later be utilised to deploy some additional nodes as backups. Therefore, when the critical nodes fail, the network is still connected. Our future work will include techniques to re-deploy sensor nodes to achieve better coverage and better lifetime, given a coverage threshold and an expected network lifetime. We will also investigate techniques to schedule, predict and defer the physical maintenance to reduce the maintenance cost.

Acknowledgements: This research is fully funded by the Irish Research Council and United Technologies Research Center Ireland Ltd. under the Enterprise Partnership Scheme. The authors thank the NIMBUS Centre at Cork Institute of Technology (CIT) Ireland, the Laboratório Tempo Sistemas de Tempo Real e Embarcados at Universidade Federal Fluminense (UFF) Brasil and the Data Sensing Lab for the datasets used in this paper.

Author Contributions: The work reported in this paper was done by Lanny Sitanayah when she was a post-doctoral researcher at University College Cork Ireland under the supervision of Cormac J. Sreenan and Szymon Fedor.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Contiki. Available online: http://www.contiki-os.org/ (accessed on 12 January 2016).
2. OMNeT++. Available online: http://www.omnetpp.org/ (accessed on 12 January 2016).
3. TOSSIM. Available online: http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM (accessed on 12 January 2016).
4. OPNET. Available online: http://www.riverbed.com/products-solutions/products/opnet.html (accessed on 12 January 2016).
5. Ns-2. Available online: http://www.isi.edu/nsnam/ns/ (accessed on 12 January 2016).
6. Ns-3. Available online: https://www.nsnam.org/ (accessed on 12 January 2016).
7. Tmote Sky Datasheet. Available online: http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf (accessed on 12 January 2016).

8. Chipara, O.; Hackmann, G.; Lu, C.; Smart, W.D.; Roman, G.C. Practical Modeling and Prediction of Radio Coverage of Indoor Sensor Networks. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10), Stockholm, Sweden, 12–15 April 2010.
9. Zuniga, M.; Krishnamachari, B. Analyzing the Transitional Region in Low Power Wireless Links. In Proceedings of the 1st IEEE International Conference on Sensor and *Ad Hoc* Communications and Networks (SECON '04), Santa Clara, CA, USA, 4–7 October 2004 .
10. Rappaport, T.S. *Wireless Communications: Principles and Practice*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
11. Bahl, P.; Padmanabhan, V.N. RADAR: An In-Building RF-Based User Location and Tracking System. In Proceedings of the 19th IEEE International Conference on Computer Communications (INFOCOM '00), Tel Aviv, Israel, 26–30 March 2000.
12. Azevedo, J.A.R.; Santos, F.E. *Signal Propagation Measurement with Wireless Sensor Nodes*; Project INTERREG III B (05/MAC/2.3/C16)—FEDER, 2005–2008; University of Madeira: Funchal, Portugal, 2007.
13. Rousselot, J.; El-Hoiydi, A.; Decotignie, J.D. Trade-off Analysis of Communication Protocols for Wireless Sensor Networks. In Proceedings of the Workshop Adaptive and Reconfigurable Embedded Systems (APRES '08), St Louis, MO, USA, 21 April 2008.
14. Onur, E.; Ersoy, C.; Deliç, H.; Akarun, L. Surveillance Wireless Sensor Networks: Deployment Quality Analysis. *IEEE Netw.* **2007**, *21*, 48–53.
15. Onur, E. Deployment Quality Measures in Sureillance Wireless Sensor Networks. Ph.D. Thesis, Boğaziçi University, Istanbu, Turkey, 2007.
16. Dunkels, A.; Osterlind, F.; Tsiftes, N.; He, Z. Software-Based On-Line Energy Estimation for Sensor Nodes. In Proceedings of the 4th Workshop Embedded Networked Sensors (EmNets '07), Cork, Ireland, 25–26 June 2007; pp. 28–32.
17. Mobile and Internet Systems Laboratory. Available online: www.cs.ucc.ie/misl (accessed on 12 January 2016).
18. Chen, B.; Jamieson, K.; Balakrishnan, H.; Morris, R. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *Wirel. Netw.* **2002**, *8*, 481–494.
19. Wang, J.; Wang, Z.; Cheng, Y.; Zhu, Y. An Energy-aware Iterative Sampling Framework for Data Gathering in Wireless Sensor Networks. *Int. J. Grid Distrib. Comput.* **2013**, *6*, 9–18.
20. Zanatta, G.; Bottari, G.D.; Guerra, R.; Leite, J. Building a WSN Infrastructure with COTS Components for the Thermal Monitoring of Datacenters. In Proceedings of the 29th Symposium on Applied Computing (SAC '14), Gyeongju, Korea, 24–28 March 2014.
21. Data Sensing Lab. Available online: http://datasensinglab.com/google-io-2013/ (accessed on 12 January 2016).
22. Datasheet SHT1x (SHT10, SHT11, SHT15) Humidity and Temperature Sensor. Available online: https://www.sparkfun.com/datasheets/Sensors/SHT1x_datasheet.pdf (accessed on 12 January 2016).
23. Gnawali, O.; Fonseca, R.; Jamieson, K.; Moss, D.; Levis, P. Collection Tree Protocol. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), Berkeley, CA, USA, 4–6 November 2009.
24. Dunkels, A. *The ContikiMAC Radio Duty Cycling Protocol*; SICS Technical Report T2011:13; SICS: Kista, Sweden, 2011.
25. Sitanayah, L.; Brown, K.N.; Sreenan, C.J. Fault-Tolerant Relay Deployment Based on Length-Constrained Connectivity and Rerouting Centrality in Wireless Sensor Networks. In Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN '12), Trento, Italy, 15–17 February 2012; pp. 115–130.
26. Fung, C.J.; Liu, Y.E. Lifetime Estimation of Large IEEE 802.15.4 Compliant Wireless Sensor Networks. In Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems (MASCOTS '08), Baltimore, MD, USA, 8–10 September 2008; pp. 1–4.
27. Barroso, A.; Roedig, U.; Sreenan, C.J. Maintenance Awareness in Wireless Sensor Networks. In Proceedings of the 1st European Conference on Wireless Sensor Networks (EWSN '04) (Short Papers), Berlin, Germany, 19–21 January 2004.
28. Sakib, K.; Tari, Z.; Bertok, P. Failed Node Replacement Policies for Maximising Sensor Network Lifetime. In Proceedings of the 6th International Symposium on Wireless and Pervasive Computing (ISWPC '11), Hong Kong, China, 23–25 February 2011; pp. 1–6.

29. Wang, X.; Xing, G.; Zhang, Y.; Lu, C.; Pless, R.; Gill, C. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In Proceedings of the 1st ACM Confernce on Embedded Networked Sensor Systems (SenSys '03), Los Angeles, CA, USA, 5 –7 November 2003; pp. 28–39.
30. Cao, Q.; Stankovic, J.A. An In-Field-Maintenance Framework for Wireless Sensor Networks. In Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'08), Santorini Island, Greece, 11–14 June 2008; pp. 457–468.
31. He, T.; Krishnamurthy, S.; Luo, L.; Yan, T.; Krogh, B.; Gu, L.; Stoleru, R.; Zhou, G.; Cao, Q.; Vicaire, P.; *et al*. Vigilnet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Trans. Sens. Netw.* **2006**, *2*, 1–38.
32. Li, J.; Wu, Y.; Stankovic, J.A.; Son, S.H.; Zhong, Z.; He, T.; Kim, B.W.; Joo, S.S. Predictive Dependency Constraint Directed Self-Healing for Wireless Sensor Networks. In Proceedings of the 7th International Conference on Networked Sensing Systems (INSS '10), Kassel, Germany, 15–18 June 2010; pp. 22–29.
33. Parikh, S.; Vokkarane, V.M.; Xing, L.; Kasilingam, D. Node-Replacement Policies to Maintain Threshold-Coverage in Wireless Sensor Networks. In Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN'07), Honolulu, HI, USA, 13–16 August 2007; pp. 760–765.
34. Kas, M.; Appala, S.; Wang, C.; Carley, K.M.; Carley, L.R.; Tonguz, O.K. What if Wireless Routers Were Social? Approaching Wireless Mesh Networks from a Social Networks Perspective. *IEEE Wirel. Commun.* **2012**, *19*, 36–43.

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons by Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).