M. Ahlberg, V. Vlassov, and Terumasa Yasui. Router Placement in Wireless Sensor Networks. KTH/ICT/ECS Technical Report, Royal Institute of Technology (KTH), Stockholm, Sweden, 2006.
Extended version of the paper presented at the 2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'06), pp.538-541, Oct. 2006; DOI: http://dx.doi.org/10.1109/MOBHOC.2006.278604

Router Placement in Wireless Sensor Networks

Michael Ahlberg
School of Information and Communication Technology
Royal Institute of Technology
Stockholm, Sweden
Email: michael@ahlberg.info

Vladimir Vlassov
School of Information and Communication Technology
Royal Institute of Technology
Stockholm, Sweden
Email: vladv@kth.se

Terumasa Yasui
Advanced Technology Research and Development Center
Mitsubishi Electric Corporation
Amagasaki, Japan
Email: Yasui.Terumasa@db.mitsubishielectric.co.jp

***Abstract*— Wireless sensor networks (sensor networks for short) have recently gained large popularity in both academy and industry. These networks of small, often battery powered, sensors can be placed where wired infrastructure is too expensive or impossible to deploy. A sensor network typically consists of a lot of nodes; some nodes might be more advanced in capabilities than others.**

In this paper we propose and evaluate algorithms for placement of routers in a sensor network. The proposed algorithms compute placement of routers in an efficient and reasonably fast way.

There are two major requirements on placement of router nodes. First, a placement must guarantee connectivity, i.e. every sensor node in the network must be able to communicate through routers with a predefined computer-connected gateway node. Second, a placement must provide robust communication in the case of router failures. That is, if one router breaks down or runs out of power, all sensor nodes must still be able to communicate with the gateway node. This can be achieved by placing redundant routers that increase the number of possible routes in the network. Both requirements should be met by placing as few routers as possible.

I. Introduction

Wireless sensor networks are a relatively new class of networks and have recently gained large popularity and interest in both academy and industry. These networks of small, often battery powered, sensors can be placed where traditional wired sensor infrastructure is too expensive, too difficult or even impossible to deploy. Currently the major use for these kinds of networks is expected to be monitoring and control of for example a home or a building.

Recent advances in techniques for wireless networking and electronics has made it possible to build cheap and power efficient wireless sensor nodes. Typically, a sensor network consists of a lot of nodes some of them might be more advanced in capabilities than others. One key feature in most of sensor networks is a self organizing ad-hoc network architecture. This means that a sensor network is capable to automatically add new units as they appear and likewise remove non operational units. The property of self-organization makes the network robust and fault tolerant, providing that there are enough redundant nodes available to keep the network operational and connected in the case of node failures.

In this article, we consider heterogeneous wireless sensor networks built of three types of nodes distinguished by their sensing and communication capabilities: (i) ordinary sensor nodes capable to communicate only with router nodes; (ii) router nodes which do not sense, are used for routing of messages, and can communicate with any other nodes; (iii) gateway nodes or computer-connected sink nodes which are capable to communicate with routers. Typically, a router node is a more expensive device that requires more memory and computing performance than a sensor node in order to implement the network protocol. Typically, a router node has a larger battery and consumes more energy than a sensor node.

To be operational a wireless sensor network must fulfill some major requirements that we briefly review below. Firstly, the network must provide *sensing coverage* defined as follows. Each sensor has a *sensing range* that defines an area which the node can sense. An area of interest (or the area of deployment of the network) is fully covered by the network if every location within the area of interest is within sensing range of at least one sensor [10], [5], [13]. For a homogeneous network[1], given the area of deployment A_d and the sensing range of a sensor A_s, one can calculate an approximate number (or a lower bound for the number) of nodes needed to cover the deployment area as $n = \frac{A_d}{A_s}$ assuming that the entire deployment area needs to be covered.

[1]In a homogeneous network all nodes have the same capabilities.

The sensing coverage property is a necessary, but not sufficient property of an ad-hoc sensor network to be operational because in most of cases the network must also provide *connectivity*, i.e. each sensor should be able to communicate (via routers) with a sink node. In other words, connectivity is the property whether messages can be transported between a pair of nodes. Each node of a wireless sensor network (a sensor, a router, or a gateway) has a *communication range* that the node can communicate within[2]. Even if a deployed network covers the entire area of interest it might be not fully connected. As shown in [13], given a communication range R_c and the sensing range R_s of the nodes in a homogeneous sensor network, a fully covered convex region is connected if $R_c \geq 2R_s$.

The third important requirement of a sensor network is *reliability* (or robustness) that depends on many factors such as node failures, node lifetime and communication problems. A general approach to improving network reliability in sensing coverage and connectivity is to deploy redundant sensors and routers. Redundancy increases the cost of deployment but on the other hand reduces the cost of maintenance as a failed node does not need to be replaced immediately. With a low level of redundancy, the network might fail if only one node fails, whereas with a high level of redundancy, the network might still operate as intended with a couple (several) failed nodes.

In this article we consider a problem of placing routers in a given wireless sensor network and propose algorithms of router placement with some redundancy that allows improving reliability of the network in terms of connectivity assuming that the sensor network provides full sensing coverage. In other words, given a deployed wireless sensor network, the problem is to find a redundant placement of as few routers as possible, while still fulfilling the following requirements:

- *Full connectivity.* All sensors must be able to communicate with a certain predefined computer-connected node (a sink node) or a gateway node.
- *Configurable redundancy for robustness*. There must be a configurable least number of possible routes from every sensor node to the gateway.
- *Placement constraints*. Router nodes cannot be placed anywhere, some placements are impossible.

When developing the placement algorithms, we focus on improving reliability of the network and do not consider the performance properties such as data throughput and/or data propagation delay (latency). As redundant router placement increases the number of possible routes in the network, we can expect that redundancy may affect the network performance. While still being of big importance for the QoS, considering network performance is outside of the scope of this paper.

[2]By analogy to the term *sensing coverage*, connectivity can be also termed as *communication coverage* that indicates whether a sensor is within communication range of a router.

A. Some Related Work

The issues of building sensor networks that provide full coverage and connectivity have arisen not only in the area of wireless sensor networks but in several other areas such as cellular networks and ad-hoc networks, as know as Mobile Ad Hoc Networks (MANETs). The difference between a wireless sensor network and an ordinary mobile ad-hoc network is that the latter is usually built upon "over deployment" of network nodes in order to achieve full and reliable coverage and connectivity, that is a typical MANET includes redundant nodes. An example of such an ad-hoc network is "The Grid Roofnet" presented in [1]. Some analysis of designs of redundant mesh networks is made in [4].

One of the major issues to be considered when building a wireless sensor network is to optimize energy consumption in the network by different approaches to power saving such as active/sleep mode, optimizing communication by, for example, using adaptive energy-efficient transmission rate and/or optimizing placement of relay/router nodes [7]. The problem of optimizing (redundant) router placement in wireless networks has been received much attention and is still of big interest in both academia and industry. It has been shown that finding optimal placement is terms of connectivity, energy consumption and reliability is an NP problem. Different heuristic approaches have been proposed such as the one based on, so called, a General Physical Layer Model, presented in [12] and a heuristic approach to relay placement proposed in [6]. Some previous related work has been done in studying (optimal) relay placement in energy-constrained wireless sensor networks in which sensors communicate collected data to a sink node via relay nodes. The typical problem studied is to find an (optimal) placement of relay nodes in a given sensor network to improve a given utility function, such as to maximize and to balance the amount of data gathered from sensors given node lifetime and communication cost. In [11], the author has specified a number of simplified relay placement problems and has shown that the problems of finding optimal solutions and even approximate solutions are NP-hard. Nevertheless, this theo-

retical observation does not prevent from using heuristic approaches to optimizing replay placement. A heuristic algorithm to optimizing relay placement as a solution to the problem of balanced data gathering in multi-hop wireless sensor networks is presented in [2]. The balanced data gathering problem is to obtain a routing solution that maximizes the amount of data received at a sink node and also guarantees that all sensors communicate a required minimum amount of data to the sink node. This problem is another formulation of the optimizing router placement problem to achieve full coverage and connectivity in energy-constrained wireless sensor networks. In this paper, we have proposed and studied a set of computationally efficient (polynomial) algorithms for both non-redundant and redundant router placement in wireless sensor networks. The major requirements to solutions obtained with these algorithms are full connectivity and a required level of redundancy.

II. Router Placement Algorithms

In this section we design and present a set of algorithms of placement of routers in a given sensor network that consists of sensors and gateway (sink) nodes. As already mentioned, we assume that a sensor can communicate only with routers or a (sink) gateway node, whereas a router can communicate with any other node. We also assume that each node (sensor, router, and gateway) is given its communication range. The problem is to place as few routers as possible to provide connectivity between each sensor node and its predefined gateway (sink) node. For each algorithm presented we assess its complexity and the number of routers it places.

A. Non-Redundant Router Placement

First, we present a set of non-redundant router placement algorithms that might still place unnecessary (redundant) nodes deployed for the sake of connectivity rather than for reliability. Next, we present a set of redundant router placement algorithms which introduce some redundancy to improve reliability of the network.

1) Trivial Router Placement: The trivial way of placing routers in a given sensor network is to place routers within communication range from each other on straight line from a sensor to its gateway for each and every sensor as shown in Figure 1. Such trivial placement provide exclusive route from each sensor to the gateway.

The *trivial placement algorithm* is shown in Algorithm 1. The algorithm is fairly straightforward: Given communication range of nodes, unless the sensor is already within reach of the gateway node, place router nodes along a straight line from the sensor to the gateway.

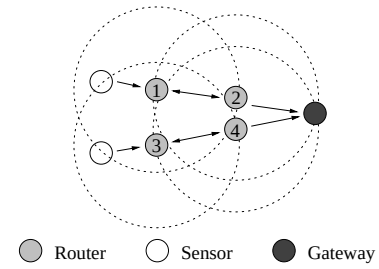


Fig. 1. Trivial placement of routers. (Communication range of routers is shown with dotted circles.)

The complexity of the trivial placement algorithm is $O(n_s d_m)$, where n_s is the number of sensors and d_m is the maximum distance from any sensor to the gateway. This expression is obtained based on the following observation: For each sensor ($O(n_s)$), the two coordinates of a router are computed and the router is deployed, until the router is within communication range of the gateway node ($O(d_m)$ iterations). In similar way, we can show that the number of routers placed is $O(n_s d_m)$.

Algorithm "Trivial Router Placement Algorithm"
Input: List of Sensor Nodes S, Gateway g
Output: List of Router Nodes R

for each sensor s in S
  if $distance(s, g) > s.range$
    deploy router r at distance $s.range$ from s towards g
    add r to R
    if $distance(r, g) < r.range$
      continue
    while $distance(r, g) > r.range$
      deploy router r_{tmp} at distance $r.range$ from r towards g
      $r \leftarrow r_{tmp}$
      add r to R
return R

Algorithm 1: Trivial placement algorithm.

2) Trivial Placement Reusing Routers: In it easy to see that the trivial algorithm places large number of redundant routers. For example, in the placement shown in Fig.1 either the router 2 or the router 4 can be removed and the resulting configuration still satisfies the connectivity requirement. The algorithm can be enhanced to reduce the number of unnecessary routers by reusing already deployed routers when making a placement decision. The idea is to place routers that connect a sensor node to the nearest already deployed router connected to the gateway as illustrated in Fig.2.

The *trivial reuse algorithm* based on reuse of already deployed routers is shown in Algorithm 2. First, the algorithm sorts sensors according to their distance to the

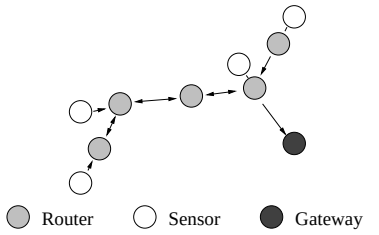


Fig. 2. Enhanced trivial placement of routers in which sensor nodes are connected to the closest already connected router node.

gateway node. Next, it connects a sensor that is closest to the gateway with a straight line of as few routers as possible. Then, the next closest sensor is connected to the closest already connected router or to the gateway if the latter is closer to the target sensor; and so on.

Algorithm "Place Routers Using Deployed Routers"
Input: List of Sensor Nodes S, Gateway g
Output: List of Router Nodes R

for each sensor s in S
  if $distance(s, g) > s.range$
    for each router $r_{deployed}$ in R
      if $distance(r_{deployed}, s) < distance(r_{selected}, s)$
        $r_{selected} \leftarrow r_{deployed}$
    if $distance(s, r_{selected}) > s.range$
      deploy router r at distance $s.range$ from s towards $r_{selected}$
      add r to R
      if $distance(r, r_{selected}) < r.range$
        continue
      while $distance(r, r_{selected}) > r.range$
        deploy router r_{tmp} at distance $r.range$ from r towards $r_{selected}$
        $r \leftarrow r_{tmp}$
        add r to R
return R

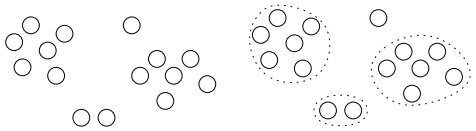
Algorithm 2: Trivial reuse algorithm.

The trivial reuse algorithm is slower than the trivial algorithm, as the former requires searching through all already placed routers for every sensor to connect. Nevertheless, we can expect that in a network with many sensor nodes, the trivil reuse algorithm places less number of routers than the trivial algorithm.

For a small number of sensors the number of routers placed by the trivial reuse algorithm is $O(n_s {d_m}^2)$, where d_m is the maximum distance from the gateway node to any sensor and n_s is the number of sensors; however with increasing number of (already connected) sensors the number of placed routers asymptotically becomes $O({d_m}^2)$ when the area of deployment is saturated with already placed routers. The area is saturated with routers when for each sensor to be connected there is always at least one route to the gateway through already deployed routers. Thus, the trivial reuse algorithm, which deploys in the best case $O({d_m}^2)$ routers, is better scalable than the trivial placement algorithm, which deploys $O(n_s d_m)$ routers: If the number of sensors increases on the fixed deployment area, the trivial reuse algorithm places less number of routers than the trivial algorithm.

The complexity of the trivial reuse algorithm is $O(n_s n_r)$, where n_s is the number of sensors and n_r is the number of routers. Using that n_r is $O(n_s {d_m}^2)$ or $O({d_m}^2)$, the complexity of the trivial reuse algorithm can be expressed as $O({n_s}^2 {d_m}^2)$ and $O(n_s {d_m}^2)$ when the area of deployment is saturated with routers.

3) Cluster Router Placement: The complexity of the router placement algorithm can be further improved by clustering sensors and connecting clusters as described below. The idea is to find groups (clusters) of closely positioned sensors and cover groups by routers instead of covering each sensor separately. Sensors in a cluster are within the communication range of a router and can be covered by (connected to) the same router. The covered clusters can be then connected with each other to provide full connectivity of the network. Fig. 3 illustrates clustering of sensors and covering the clusters by routers.



(a) Sensor placement (b) Clustered sensors

Fig. 3. Finding Clusters of Sensors

The router placement algorithm based on clustering (further we call it the *cluster algorithm*) consists of two steps: (1) clustering of sensors based on communication ranges and covering each cluster by a router(s) so that all sensors are within communication range of the router; (2) connecting the routers using the trivial reuse algorithm (Algorithm 2) applied to the routers placed on the first step rather than to sensors. The first step is performed as follows: For every sensor s, find three closest neighbors and add them to the cluster, to which s belongs, if the neighbors are within a predefined distance from the sensor; if a sensor node belongs to more than one cluster, the clusters are merged into one cluster. The cluster algorithm is shown in Algorithm 3 (Note that the cluster algorithm uses the *save tree algorithm* shown in Algorithm 4). The cluster algorithm is a modification of the algorithm described in [8].

It's easy to see that the complexity of the clustering algorithm is $O({n_s}^2)$ (where n_s is the number of sensor nodes) because each sensor is inspected for clustering with all other sensors. If we then use the algorithm in

Algorithm "Find Clusters of Nodes"
Input: List of Sensor Nodes S
Output: List of Clusters C

for each sensor s in S
  for each sensor n in S
    if $distance(s, n) < s.range$
      add n to $s.neighbors$ if closer than any other neighbor
        (replace the furthest neighbor if more than 3 neighbors)
  for each neighbor n in $s.neighbors$
    add s to $n.otherneighbors$

$clusteridid \leftarrow 0$ **for** each sensor s in S
  if $s.clusterid = 0$
    $clusterid \leftarrow clusterid + 1$
    $c \leftarrow savetree(s, clusterid)$ (see Algorithm 4)
    add c to C
return C

Algorithm 3: Find sensor clusters

Algorithm "Save Tree"
Input: Node n
Output: List of Nodes Belonging to the Same Tree l

if not $n.marked$
  set $n.marked$
  add n to l
  for each neighbor s in $n.neighbors$
    add $savetree(s)$ to l
  for each neighbor s in $n.otherneighbors$
    add $savetree(s)$ to l
  return l

Algorithm 4: Saving of neighbor trees.

Algorithm 2 for router placement, the complexity of the second step is $O(n_c {d_m}^2)$, where n_c is the number of clusters found on the first step. From this we can see that the total complexity depends on how the sensor nodes are deployed. This assessment shows that the cluster algorithm is faster than the trivial router reuse algorithm.

It is difficult to give an analytical assessment of the number of routers that can be placed by this algorithm because the number of routers depends on how the sensors are deployed. We leave this assessment to our future work. In the worst case the number of clusters is equal to the number of sensors and the cluster algorithm will place the same number of routers as the trivial router reuse algorithm used on the second step of the cluster algorithm. We have evaluated both the time complexity and the number of placed routers by implementing algorithms (see Section III).

B. Redundant Routes

Solutions obtained by non-redundant router placement algorithms presented above have full connectivity as long as all routers are functional. But if one or more routers fail, large part of the network might fail as shown, for example, in Figure 4 where a failure of only one router causes the entire network to break down.

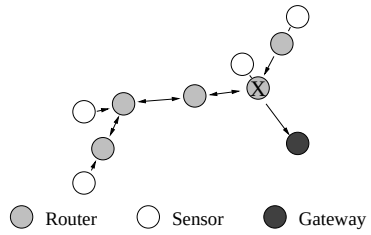


Fig. 4. One failed router (marked "X") causes all sensors to be unable to communicate with the gateway node.

A general approach to improve reliability of a connected sensor network is to add redundant routers so that each sensor has more than one route to communicate with the gateway node. We call *mutually exclusive* any routes or paths through the connection graph if the routes do not share any routers. We can define the *level of redundancy* of a sensor as the number of mutually exclusive routes from the sensor to a predefined gateway node. The level of redundancy of the entire network can be estimated as the minimum level of redundancy in the network. We call a router placement solution *k-redundant* is its level of redundency is k, i.e. any sensor has at least k exclusive (alternative) routes to its gateway node.

The simplest way to achieve a k-redundant solution is to deploy k routers at every position designated by a non-redundant router placement algorithm. This approach is simple, fast and straightforward, but in most cases it is possible to achieve the same level of redundancy with less number of routers. Figure 5 shows a simple example in which 5 routers are used in a non-redundant solution whereas 10 routers are needed for the 2-redundant solution obtained by placing 2 routers in each of 5 positions. A better solution that uses 9 routers to achieve the same level of redundancy in this network is shown in Figure 6. This solution complies well with the observation reported in [9] that optimal 2-way redundant routes in connection graphs tend to form rings. Even though only one router is saved in this example, we can expect that more routers can be saved in this way in a larger network.

C. Non-trivial Redundant Router Placement

In this section, we propose a redundant router placement algorithm that takes as an input a connected network with routers placed by one of the non-redundant router placement algorithms presented in Section II-A and places a number of redundant routers (if necessary) to achieve a given level of redundancy specified as the minimum number of exclusive routes between any sensor and a gateway node.

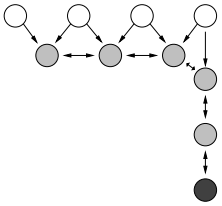


Fig. 5. Non-optimal redundant solution: all router nodes are multiplied.

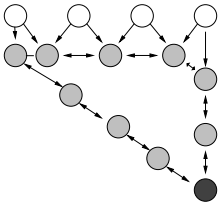


Fig. 6. Better redundant solution that utilizes other already placed routers.

The algorithm is comprised of two stages. At the first stage, it determines and counts the number of mutually exclusive routes in the connection graph that represents the given network in order to estimate the level of redundancy in the given network, and whether more routers need to be added to increase redundancy to the required level. As a side effect of computing mutually exclusive routes, the algorithm also checks whether the network is fully connected (i.e. fully covered by routers) and adds missing connections if necessary. At the second stage, if the level of redundancy estimated on the first stage is lower than required, the algorithm places redundant routers to achieve the required level of redundancy.

The sub-algorithm that computes mutually exclusive routes passes two steps. On the first step it transforms the bidirectional connection graph of the given network to a unidirectional connection graph in order to simplify counting of mutually exclusive routes. On the second step, it computes the maximum flow from each sensor to the gateway node in order to define the number of mutually exclusive routes: the maximum flow from a sensor to its gateway is equal to the number of mutually exclusive routes from the sensor to the gateway.

For example, Figure 7 shows the 2-reduntant router placement obtained by our algorithm applied to the connected network shown in Figure Figure 4. The initial network was built using the algorithm (Algorithm 2).

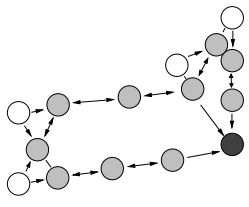
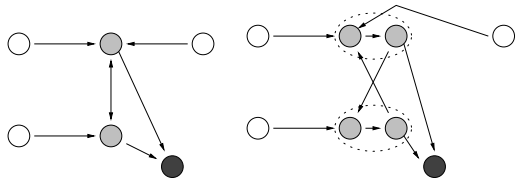


Fig. 7. 2-redundant network. Every sensor node has two mutually exclusive routes to the gateway node.

Next subsections describe the two stages of the redundant router placement algorithm.

1) Stage 1: Computing Maximum Flow in the Connection Graph: To simplify counting of mutually exclusive routes the bidirectional connection graph that represents a given network is transformed to a unidirectional graph. This transformation is performed by substituting two routers connected with a unidirectional edge for every router in the graph so that all incoming edges go to one of the two routers whereas all outgoing edges start from the other router as illustrated in Figure 8.



(a) Initial bidirectional graph. (b) Resultsing unidirectional graph.

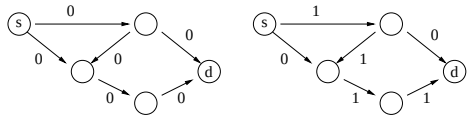
Fig. 8. Transformation of bidirectional graph to unidirectional graph

Note that every bidirectional connection graph has one and only one transformation that is unique and does not depend on the order of traversal.

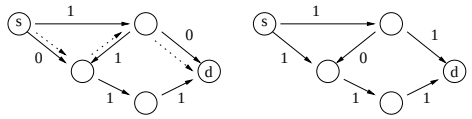
To compute maximum flow in the unidirectional graph obtained on the previous step, we use a simplified Ford-Fulkerson algorithm described in [3], Section 8.2.2. The Ford-Fulkerson algorithm computes the maximum flow from a source to a sink in a weighted directed graph. To apply the algorithm to a unidirectional graph representing a connected sensor network, the maximum flow through every connection is set to 1 to indicate that every connection can accommodate one and only one route. The Ford-Fulkerson algorithm is based on finding paths where more flow can be pushed through, so called augmenting paths. Forward edges using less than full capacity and backward edges with flow more than 0 can be used as a path. The graph is traversed in a breadth first manner, searching for unused paths in the graph. Since the maximum flow through any edge in our case is 1 and it is either used or not, the algorithm can be simplified as follows. The flow between two nodes (e.g. a sensor and a gateway) is computed as follows.

1 Find a usable path through the graph either by using an unused path or by using a used edge in the reverse direction.
2 Mark path as used, whereas edges used in the reverse direction are marked “unused”.
3 Repeat Step 1 and 2 until no new paths found.
4 Sum the flows in the edges that start at the source to get the total flow through the graph from the source to the destination.

Figure 9 shows an example of finding of augmenting paths between the source s and the destination d. A path using flow 1 is found and marked as used (Fig.9(b). Then a new path is found by using one of the edges backwards (Fig.9(c). When the path is marked as used, the weight of the edge used backwards is reduced to 0 (Fig.9(d).



(a) Graph with no flow (b) One route is used



(c) A new route using a backward edge, indicated with dashed lines (d) The final result with a total flow of 2

Fig. 9. Finding Augmenting Paths in a Graph

The algorithm to find augmenting paths in a weighted unidirectional graph, based on the algorithm in [3], Section 8.2.1, is shown in Algorithm 5. The simplified Ford-Fulkerson algorithm is shown in Algorithm 6.

Algorithm Find Augmenting Path
Input: weighted directed graph G, source s, destination d
Output: path π

for each node n in G
  $n.visited \leftarrow false$
create FIFO-queue Q
put s into Q
while Q is not empty
  get node n from Q
  if $n = d$
    build path π recursively starting at n.route
    return path π
  $n.visited \leftarrow true$
  for each edge e to and from n
    if not $e.destination.visited$ **and** e is forward edge **and** $e.flow = 0$
      $e.destination.route \leftarrow e$
      put $e.destination$ into Q
    if not $e.source.visited$ **and** e is backward edge **and** $e.flow = 1$
      $e.source.route \leftarrow e$
      put $e.source$ into Q

Algorithm 5: Algorithm for finding augmenting paths

2) Stage 2: Redundant Router Placement: After computing the number of mutually exclusive routes, we can estimate whether the number of routes between a sensor and the gateway is sufficient to provide the required level of redundancy, i.e. whether the number of routes is at least the required level of redundancy. If not, new exclusive routes are created with the algorithm shown in Algorithm 7.

Algorithm Maximum Flow
Input: weighted directed graph G, source s, destination d
Output: maximum flow $maxFlow$ from s to d

for each edge e in G
  $e.flow \leftarrow 0$
$maxFlow \leftarrow 0$
repeat
  traverse G starting at s to find a augmenting path π to d, see Algorithm 5
  if a path π exists **then**
    $maxFlow \leftarrow maxFlow + 1$
    for each edge e in π **do**
      if e is forward edge **then**
        $e.flow \leftarrow 1$
      else
        $e.flow \leftarrow 0$
  else
    stop

Algorithm 6: Simplified Ford-Fulkerson Algorithm

Algorithm Create Redundant Route from s to d
Input: connection graph G, source s, destination d, gateway g
Output: connection graph G

$R \leftarrow$ router nodes in G
$S \leftarrow$ sensor nodes in G
sort S by decreasing distance to g
for ($i = 0$; $i <$ desired level of redundancy; $i++$)
  for each sensor s in S
    compute number of routes n from s to g in G using the "Maximum Flow" algorithm in Algorithm 6
    if $n \leq$ desired redundancy
      for each router r in R
        if r is not on same branch in the graph as s
          build route from s to r, this can be done in the same way as in any of the non-redundant algorithms
          continue

Algorithm 7: Creating of Redundant Routes

First, the algorithm checks whether the sensors already have enough connections to satisfy the redundancy required. This check is performed using the "Maximum Flow" algorithm (Algorithm 6). Next, if the level of redundancy must be increased, new connections are added as follows. The sensor nodes are examined starting with the node furthest away from the gateway node: if a node does not have enough connections, a new mutual exclusive connection is created by to the closest router that is on the other branch in the graph than the sensor itself. The actual placement of routers is then done in the same way as in the non-redundant placement algorithms described in Section II-A. These steps, checking the number of exclusive routes (redundancy level) and adding redundant nodes if needed, are repeated for every sensor node and every level of redundancy required.

3) Analysis of the Algorithm: Based on the rigorous analysis in [3], Section 8.3, the complexity of the algorithm for finding augmenting paths (Algorithm 5) is

$O(n_e)$ where n_e is the number of edges in the graph. The complexity of the simplified Ford-Fulkerson algorithm (Algorithm 6) depends on the total flow f and on the number of routers n_r because, in the worst case, the algorithm marks all routers when computing flows. Based on these observations, we can conclude that the complexity of the algorithm for computing maximum flow (Algorithm 6) is $O(fn_en_r)$. The branch computation has complexity $O({n_h}^2)$, with n_h the maximum number of hops in the graph. Since the number of edges n_e and number of routers n_r always is greater or equal to the maximum hop count n_h the flow computation is the dominant part. The placement of redundant routers, using a similar algorithm as the trivial one in Section II-A.1 needs $O(d)$ steps, where d is the maximum distance between any two nodes in the graph. Since $O(fn_en_r)$ is a larger factor than $O(d)$, the $O(d)$ factor can be ignored.

Combining the above results yields a total complexity of the redundant router placement algorithm of $O(rn_s(fn_en_r)^r)$, where r is the required level of redundancy. Thus, the time needed by the algorithm increases steeply as the level of redundancy increases.

We leave the analytical assessment of the number of routers placed by the algorithm to our future work. Nevertheless we have evaluate the algorithms by implementing them (see Section III).

D. Optimization

A solution obtained by using the router placement algorithms presented above might be not optional as some of deployed routes might be not needed to achieve full connectivity and required level of redundancy. Below, we shortly present two optimization procedures that remove unnecessary routers still preserving full connectivity and required level of redundancy in an optimized solution.

In a solution with non-redundant connectivity, some routes might be longer than necessary and some routers might be not needed to have a connected and covered network. To optimize the non-redundant solution, all sensor nodes are reconnected to a reachable router node that has as short route to the gateway node as possible. This procedure is performed for every router. The routers with only connection to another router node are removed. This is repeated until no more routers can be removed.

When the routes are created using the redundant algorithm in Section II-C, more routers than necessary might be deployed. To find routers that can be removed, all routers one by one are temporarily removed from the network. For every removed router, the number of routes is computed by using the algorithm described in Section II-C. If all sensors still have enough redundancy, the router node is permanently removed. The optimization algorithm is shown in Algorithm 8.

Algorithm Optimize Connection Graph
Input: connection graph G, source s, destination d, gateway g
Output: connection graph G

$R \leftarrow$ router nodes in G
$S \leftarrow$ sensor nodes in G
sort S by decreasing distance to g
for each sensor s in S
  for each router r in R
    mark r as failed
    compute number of routes n from s to g in G using the "Maximum Flow" algorithm from Algorithm 6.
    if $n \leq$ desired redundancy
      mark r as non failed
    else
      permanently remove r

Algorithm 8: Optimization of the Connection Graph

III. Evaluation

Table I summarizes assessments of complexity and the number of routers placed for different algorithms presented in the previous section.

Algorithm	Router Nodes	Time
Non-Redundant		
Trivial	$O(n_sd_m)$	$O(n_sd_m)$
Trivial Reuse (few sensors)	$O(n_sd_m^2)$	$O(n_s^2d_m^2)$
Trivial Reuse (many sensors)	$O(d_m^2)$	$O(n_sd_m^2)$
Cluster	n.a.	$O(n_cd_m^2)$
Redundant		
Trivial	n.a.	$O(n_cd_m^2)$
Non-Trivial	n.a.	$O(rn_s(rn_en_r)^r)$

TABLE I
Time complexity and the number of placed routers

To verify and evaluate the proposed algorithms, each algorithm has been implemented as an application. The varied input parameters in evaluation experiments are the number of sensors randomly placed on the deployment area and the required level of redundancy. The measured parameters are the number of routers deployed and the execution time. In all evaluation experiments, the router-to-router communication range is 30 meters; the sensor-to-router communication range is 15 meters; the area of deployment is a square of 400 meters. With these values of input parameters, the area of deployment is $A_d = 160000\ m^2$, and a router to sensor communication range is $A_c = 15^2\pi\ m^2$. As the above values fulfill the requirement that the router-to-router communication range is twice the router-to-sensor communication range,

we can use the formula mentioned in Section I to compute the minimum number of routers that should be placed to provide connectivity of sensors. The computed theoretical minimum is $\frac{A_d}{A_c} \approx 226$ routers.

A. Evaluation of Non-Redundant Placement Algorithms

Figure 10, Table II and Table III show result of evaluation of non-redundant placement algorithms presented in II-A. The plots depict the number of routers and the computation time as functions of the number of sensors. We have observed that the results of evaluation of algorithms without optimization coincide with the analytical assessments shown in Table I. The results in Table II show that the cluster algorithm without optimization (Algorithm 3 and Algorithm 4) gives the number of deployed routers which is closest to the theoretical minimum of 226 routers for the given input parameters, whereas the trivial placement algorithm (Algorithm 1) and the trivial router reuse algorithm (Algorithm 2) deploy much more routers than the theoretical minimum. Figure 10(c) shows that the number of routers has an asymptotic behavior as the number of sensors increases. This means that there is a saturation point in increasing the number of sensors when for a new sensor added beyond this point there is always at least one route of already deployed routers connecting the new sensor to the gateway.

Figure 10(d) and Table III show results of evaluation of non-redundant algorithms with optimization described in Section II-D. The results show that optimization allows to remove quite a few routers and that the "Cluster" algorithm is faster than all others. Thus, the cluster algorithm with optimization is the best choice for non-redundant optimized placement of routers.

	200 Sensors		500 Sensors	
Algorithm	Routers	Time/s	Routers	Time/s
Trivial	1091	0.069	2713	0.12
Trivial Reuse	248	4.3	458	73
Cluster	199	0.30	237	0.61

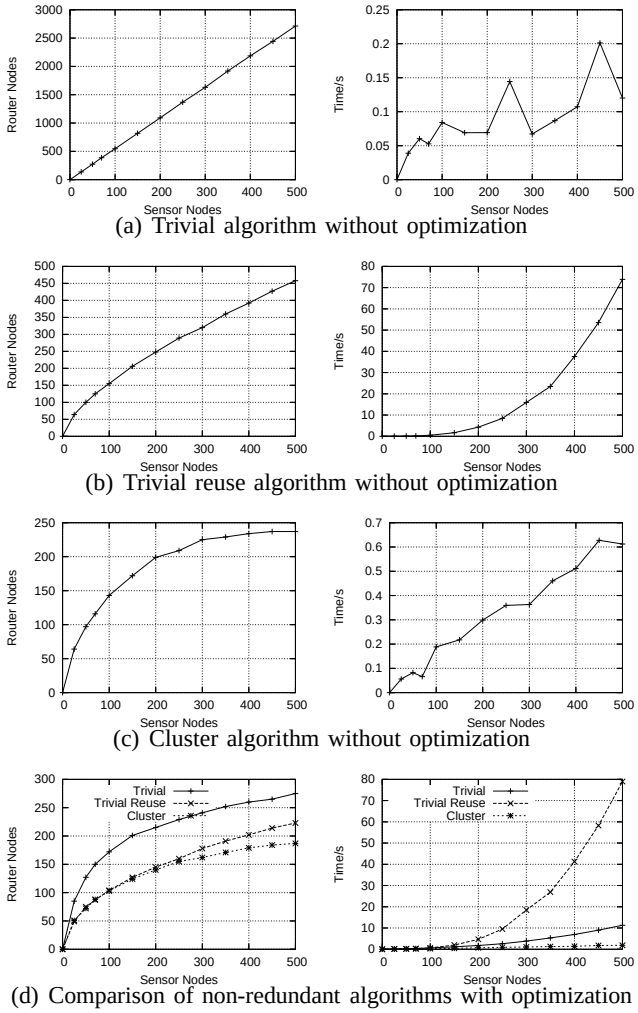
TABLE II

COMPARISON OF NON-REDUNDANT NON-OPTIMIZED SOLUTIONS

	200 Sensors		500 Sensors	
Algorithm	Routers	Time/s	Routers	Time/s
Trivial	215	1.8	275	11
Trivial Reuse	145	4.7	223	79
Cluster	140	0.69	187	1.9

TABLE III

COMPARISON OF NON-REDUNDANT OPTIMIZED SOLUTIONS



(a) Trivial algorithm without optimization

(b) Trivial reuse algorithm without optimization

(c) Cluster algorithm without optimization

(d) Comparison of non-redundant algorithms with optimization

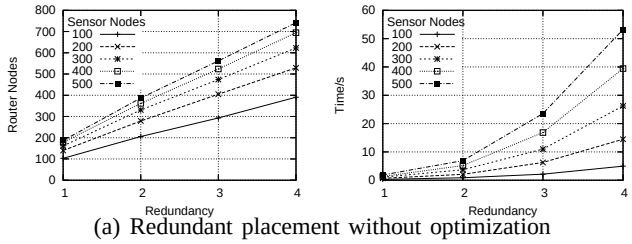
Fig. 10. Results of Evaluation of Non-Redundant Placement Algorithms (Plots show the number of placed routers and the computation time as functions of the number of sensors)

B. Evaluation of the Redundant Placement Algorithm

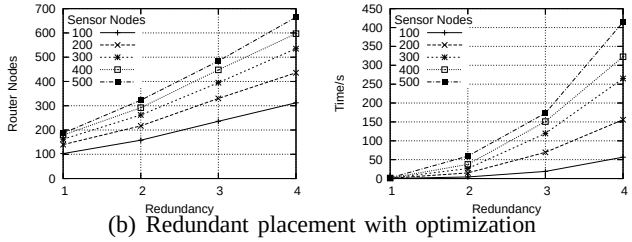
Figure 11(a) and Table IV show results of evaluation of the redundant placement algorithm described in Section II-C for different level of redundancy and different number of sensors. Figure 11(a) indicates that our redundant placement algorithm scales better than a trivial redundant placement algorithm in turms of the number of placed routers. Figure 11(a) shows also that the computation time increases steeply as the level of redundancy increases as predicted in Section II-C.3.

Figure 11(b) and Table V show results of evaluation of the redundant placement algorithm with optimization that reduces the number of routers still preserving connectivity and required level of redundancy. We can see from Table V that after optimization the number of routers is lower than in the non-optimized redun-

dant solution (compare with Table IV) for the cost of increased computation time. Figure 11(b) also depicts performance of the redundant router placement algorithm with optimization. The computation time of the algorithm increases steeply as the redundancy level increases.



(a) Redundant placement without optimization



(b) Redundant placement with optimization

Fig. 11. Results of Evaluation of the Redundant Placement Algorithm.

	200 Sensors		500 Sensors	
Redundancy	Routers	Time/s	Routers	Time/s
1	140	0.69	187	1.9
2	279	2.1	387	6.9
3	405	6.3	562	24
4	529	14	742	53

TABLE IV

COMPARISON OF REDUNDANT NON-OPTIMIZED SOLUTIONS

	200 Sensor Nodes		500 Sensor Nodes	
Redundancy	Router Nodes	Time/s	Router Nodes	Time/s
1	140	0.69	187	1.9
2	217	14.9	323	59
3	330	69	484	170
4	436	160	665	410

TABLE V

COMPARISON OF REDUNDANT OPTIMIZED SOLUTIONS

IV. CONCLUSIONS

We have presented algorithms for non-redundant and redundant router placement in wireless sensor networks. The non-redundant placement algorithms deploy routers to achieve full connectivity of the network, whereas the redundant placement algorithm adds redundant routers to a non-redundant solution in order to achieve required level of redundancy for reliability. We have proposed two optimizing procedures that allow reducing the number of routers preserving the full connectivity and the required level of redundancy. We have shown the complexity of the algorithms and the number of deployed routers (Table I). We have verifyed and evaluated the algorithms. Evaluation results show that the cluster algorithm with optimization performs better than other non-redundant placement algorithms. The cluster algorithm should be used to obtain initial router placement for the redundant router placement algorithm.

Our future work includes extending the proposed algorithms to the 3-dimensional case; taking in to account other parameters such as the network latency (or propagation delay), constraints on router placement.

REFERENCES

[1] Benjamin A. Chambers. The grid roofnet: a rooftop ad hoc wireless network. Master's thesis, MIT, 2002.

[2] P. Floréen, P. Kaski, J. Kohonen, and P. Orponen. Exact and approximate balanced data gathering in energy-constrained sensor networks. *Theoretical Computer Science*, Volume 344, Issue 1, No. 11, pp. 30–46, 2005.

[3] M- T. Goodrich and R. Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples.* John Wiley & Sons, 2002.

[4] P.-H. Hsiao and H. T. Kung. Layout design for multiple collocated wireless mesh networks. In *Proc. of IEEE VTC*, 2004.

[5] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *Proc. of the 2nd ACM Int Conf on Wireless sensor networks and applications*, pp. 115–121, 2003.

[6] Y. T. Hou, Yi Shi, and Ha. D. Sherali. On Energy Provisioning and Relay Node Placement for Wireless Sensor Networks. In *IEEE Trans. on Wireless Comm.*, (4)5:2579–2590, Sep. 2005.

[7] Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks. ed. J. Wu. Auerhach Publications, 2006.

[8] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, Aug 1999.

[9] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager. Redundant trees for preplanned recovery in arbitary vertex-redundant or edge-redundant graphs. *IEE/ACM Trans on Networking*, 7(5):641–652, Oct 1999.

[10] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pp. 1380–1387, 2001.

[11] J. Suomela. Computational Complexity of Relay Placement in Sensor Networks. In *SOFSEM 2006*, LNCS 3831, pp. 521–529. Springer-Verlag, 2006.

[12] S. Toumpis and G. A. Gupta. Optimal Placement of Nodes in Large Sensor Networks Under a General Physical Layer Model. In *Proc of IEEE SECON*, September 2005.

[13] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proc of the 1-st Int Conf on Embedded networked sensor systems*, pp. 28–39. ACM Press, 2003.