Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Christian Hess

Routing Protocol for a Wireless Fire Detection Network

Semester Thesis SA-2005-35
Summer Term 2005

Tutors:
Dr. Martin May
Vincent Lenders
Dr. August Kälin, Siemens

Supervisor:
Prof. Bernhard Plattner

21.10.2005

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Abstract

This paper describes a routing protocol for a wireless fire detection network. This kind of a wireless sensor network requires a high level of energy efficiency as well as link reliability. Two versions of a routing algorithm are presented in this work, both of them based on the concept of field-based routing, where potentials are assigned to each sensor node in order to determine a path to the central data sink. The two versions differ in the function that is used to calculate the potential of the nodes. One uses the distance to the sink in hops, the other calculates the potential based on RSSI (Receive Signal Strength Indication) values.

Simulations have shown that the RSSI based version creates a bigger amount of overhead traffic, but also brings important advantages like much better link qualities. The RSSI based algorithm therefore seems to be better suitable for the application of wireless fire sensors.

Inhaltsangabe

Diese Arbeit beschreibt ein Routing-Protokoll für ein drahtloses Netzwerk von Feuersensoren. Diese Art eines drahtlosen Sensornetzwerks erfordert sowohl einen tiefen Energieverbrauch als auch zuverlässige Funkverbindungen. Es werden in dieser Arbeit zwei Versionen eines Routing-Algorithmus vorgestellt. Beide basieren auf Field-based Routing, einem Konzept, bei dem jedem Sensorknoten ein Potenzial zugewiesen wird, um damit einen Pfad zur zentralen Einrichtung (Sink) zu finden. Die beiden Versionen unterscheiden sich in der Funktion, mit der sie das Potenzial der Knoten berechnen. Im einen Fall wird die Anzahl der Hops zum Sink verwendet, im andern Fall sind es RSSI-Werte (Receive Signal Strength Indication).

In Simulationen hat sich gezeigt, dass die RSSI-basierte Version zwar grösseren Datenverkehr verursacht, jedoch auch deutliche Vorteile wie zum Beispiel zuverlässigere Funkverbindungen mit sich bringt. Die RSSI-basierte Version scheint daher für die Anwendung bei drahtlosen Feuersensoren besser geeignet zu sein.

Table of Contents

1 Introduction

A wireless sensor network is a system that consists of multiple nodes that are collecting information. These nodes are not connected by any fixed infrastructure but communicate with each other using a wireless radio link. They may either exchange information with other nodes or forward collected data towards one or many central devices (sinks) that are responsible for further processing of these data.

Generally, each sensor node has its own power source (e.g. batteries) of limited capacity, energy efficiency is therefore one of the most important issues in wireless sensor networks. It is usually the transceiver that is responsible for a considerable part of the power consumption of such a sensor node. To keep this power consumption at a reasonable level, the transmission power and hence the transmission range are limited. These constrictions normally exclude the possibility of all nodes communicating with the sink directly. Instead, a multi-hop network structure has to be built.

This is the point where a routing algorithm is needed. It is responsible to create some kind of a routing tree so that every node finds a path to communicate with the sink. This routing algorithm could run as a centralized application that computes an optimal routing scheme and tells every node what to do. The other possibility is a decentralized algorithm where the nodes in the network autonomously create a routing tree by exchanging information with their neighbors. In either way, the number of overhead messages that are used by the routing algorithm to create a routing tree has to be kept as low as possible for the sake of energy efficiency.

The goal of this semester thesis was to develop and evaluate a decentralized routing algorithm for a wireless fire detector network (WFDN). In this area, reliability is another crucial point that comes up. A node failing to communicate with the central sink for some time leads to a false alarm, which cannot be tolerated. Obviously, a fire detection network also calls for lower delays than many other sensor applications. A good routing algorithm for a WFDN should therefore respect many different aspects like energy consumption, link reliability or end-to-end delays.

Chapter 2 of this thesis gives an overview on the underlying WFDN project and describes the protocols used therein. The basic concept that is used for the routing algorithm is described in chapter 3. This concept is adapted to the specific requirements, and a very simple and a bit more complex version of a routing algorithm are presented. Chapter 4 contains the evaluation of these and discusses advantages and drawbacks on the basis of simulation results. Finally, Chapter 5 concludes this work.

2 The Wireless Fire Detection Network (WFDN) Project

WFDN is a project of Siemens with the goal to develop a fire detection network that consists of battery-powered sensor nodes without any fixed connection infrastructure. Currently, a small test network of programmable nodes exists to gain insight into their real-world performance.

The objective of this thesis was to develop a routing algorithm that is more suitable to the needs of this project than the current one.

2.1 Protocol Layering

As a communication system, the WFDN project comes with a layered structure. Table 2.1 shows the protocols that are currently used in the different layers.

Layer	Protocol
Application	WFDN
Network / Routing	Routing algorithm based on flood propagation
MAC	WiseMac
Radio/Physical	WiseNet Transceiver

Table 2.1. Protocols in the WFDN Project

Because both application- and MAC-layer are of particular importance for a routing protocol, these are shortly presented in the next sections.

2.2 Mac Layer

This sensor network uses WiseMAC [1] as medium access control protocol. WiseMAC is a CSMA-based MAC protocol using the preamble sampling technique to minimize energy consumption due to idle listening.

To check for traffic, nodes listen to the radio channel every sampling period for a short duration and then turn off their transceiver for the rest of the sampling period. If a node wants to send data, it first has to send a wake-up preamble with a length equal to the sampling period to be sure the receiving node wakes up during this preamble.

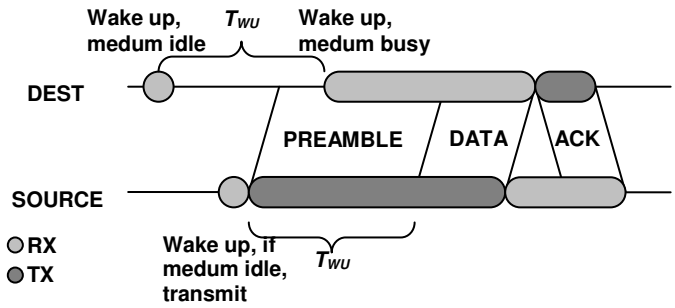

Figure 2.2: Preamble Sampling. Source: [1]

WiseMAC further decreases energy consumption by minimizing the size of these preambles. To do so, nodes keep track of the wake-up schedules of their neighbors. Because

the transceiver can not switch instantly from receiving to transmitting mode, collisions may occur if two nodes sense the medium at the same time and then start a transmission.

If multiple nodes want to communicate with the same node, collisions become more and more likely because of the synchronization of the wake-up schedules. To reduce probability of collisions, a medium reservation preamble of random length is being used.

To address the hidden node problem, WiseMAC works with a transmission range that is smaller than its actual sensitivity range. Further details about this MAC protocol can be found in [1].

2.3 Routing Layer

The routing system employed so far is a very simple one. It is based on a flood propagation algorithm where the sink initiates a flood wave. After a small time window, each node that received the message from the sink inserts its own hop count into the messages and forwards the flood wave. Using this procedure, each node learns its distance from the sink and its parent.

Nodes keep a list of possible parents where they insert all received broadcast messages with a smaller hop count. These "alternative parents" can be used in case of a broken link to the favorite parent. The routing algorithm described in the next chapter uses the same approach.

2.4 Application Layer

In normal operation, all nodes regularly have to send some status messages to the sink. To meet the requirements on a fire detection network, a maximum delay of 30 seconds is tolerated, i.e. a status message has to arrive at the sink not later than 30 seconds after it has been sent by the sensor node. These status messages basically can consist of a single bit indicating that everything is ok. The approach done here to minimize the number of transmissions is to schedule the transmission of status messages according to a node's hop distance (h) from the sink. Each node gathers messages from its children in the period $[T-(h+1)\Delta_m; T-h\Delta_m]$ where T = k * 30s. The node puts messages from its children together with its own and starts to transmit this at $T-h\Delta_m$. Using this procedure, every node has to send exactly one message within each 30s under normal operation.

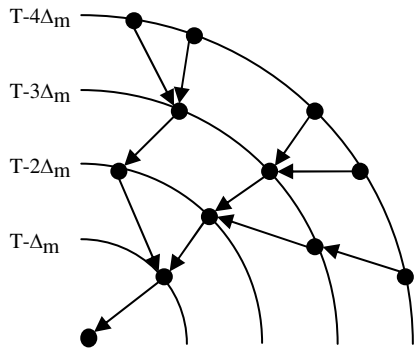

Figure 2.3: Alive status collection.
Source: [2]

The size of the collection window Δ_m is a trade-off between giving the nodes enough time to successfully transmit their message and keeping the delay for nodes with large hop counts low. In fact, Δ_m limits the maximal hop distance a node can have. Currently, Δ_m is set to 3s, resulting in a maximum hop distance of 10.

If a node fails to transmit its message within the right time window, it can tell its parent to forward this message immediately to the sink.

The concepts described above bring up some requirements that have to be met by other layers:

- Each node has to know its next hop on the route towards the sink
- Each node has to know its exact hop distance to the sink
- All nodes have to be synchronized within a certain accuracy

When building a routing algorithm, one has to keep in mind that it is a task of the routing protocol to fulfill these conditions.

In the case of a fire alarm, things change noticeably. Now, energy consumption is not important anymore and all nodes just forward the alarm message as quick as they can. But as alarms shouldn't appear in everyday life, all protocols are optimized for the standard forwarding of status messages.

There are concepts to control the network and collect statistics from nodes on application layer. These concepts were not considered in this work as they don't have any major impact on the routing protocol. Details can be found in [2].

3 Routing Algorithms

3.1 General Concepts

3.1.1 Field-based Routing

The concept of field-based Routing [3] was chosen to be used as a basis for the routing algorithm developed in this work. Field-based Routing is a service discovery mechanism for ad hoc networks. In analogy of an electrostatic field, where negative test charges are attracted by positive ones, nodes get a positive potential value, and packets are seen as negative charges. Packets are then just sent to the neighbor node with the highest potential. This implies that the function that assigns a potential to a node has to be monotonically increasing from each node towards the sink. If the potential wasn't monotonically increasing, packets wouldn't reach the sink.

Using this technique, the occurrence of loops in the network can be prevented. The simplest example for a monotonically increasing potential function is a function that is inversely proportional to the distance (number of hops) a node has to the sink. Other metrics such as signal strength or remaining energy could be included in the potential function, as long as the function is shown to be monotonically increasing.

3.1.2 Parent Tables

If the nodes keep a list with the potential values of their neighbors, a certain degree of flexibility can be achieved. Each neighbor is a possible parent, i.e. it can be the next station on the path towards the sink. Hence, for each neighbor, the node has to store the hop distance and the potential it would have if using the link to this particular parent. The parent which gives the highest potential is marked as the favorite parent; all other parents are alternative parents. In normal operation, all traffic is routed towards the favorite parent. Should that favorite parent be unreachable at some point of time, the node can quickly react and activate one (the one with the best potential) of the alternative parents, All further traffic is then sent through this alternative route.

Of course, nodes should only add another node to their list of possible parents, if they are sure that this node is not farther away from the root of the tree. It is the task of a routing algorithm to decide which nodes can be added to that table, and which nodes can't (or even have to be removed).

3.2 Adaptation to the WFDN

3.2.1 Requirements

A sensor network with a single sink leads to a tree structure at the routing level as nodes generally don't have to communicate with each other. A tree structure means the following: There is a root node (the sink) at the bottom of the tree and the sensor nodes distributed over the branches of the tree. Every node (except for the sink) has exactly one parent node, which is the next step on the way to the root. At the same time each node can have zero or more children.

A routing algorithm for this WFDN has to respect several requisites. The most important aspects that should be considered are:

Energy consumption

- The routing algorithm should be able to build up a routing tree by sending a minimum of routing packets.

- Also because of limited energy resources, the transmission range of the nodes is limited, so not every node is able to communicate directly with the sink.

Stability/Reliablility

- The algorithm has to assure that there are no loops in the routing tree.
- It should consider conditions of the MAC- and Application-Protocol (e.g. not too many children for the same parent, to reduce probability of collisions; limited maximum hop-count to ensure that delays are within specifications).
- Unreliable links or links with weak signal strengths should be avoided.

Flexibility

- The algorithm has to be able to quickly adapt to changing environment (e.g. broken links)

3.2.2 Flood Propagation

To construct a routing tree, nodes first have to collect information about their neighborhood. An easy way to do so is a flooding, where all nodes just forward their information to all other nodes in their neighborhood. The designated root of the tree, the sink, can initiate a flood wave by sending out its potential and hop count. Because the sink must have the highest potential in the network, it sends a potential of ∞ and a hop count of 0. All nodes that receive that information can now add the sink to their parent table and calculate their own potential. Now it's their turn to send out their information, and so the flood wave can propagate through the network. It is important that the nodes don't forward a flood wave every time they receive one of these flood messages, otherwise an endless and huge amount of traffic would be generated. One possibility to limit the extent of a flood wave is to allow the nodes to forward a flood wave only once, which still would give each node an opportunity to broadcast its information. The here described routing algorithms work a bit different. They let the nodes forward the flood wave as long as they change their potential. The broadcasting stops as soon as every node has found it optimal parent.

To avoid collisions, a node should wait for some random time before it forwards a flood message, otherwise all nodes that received that message would try to forward it at the same moment. In addition to this, it proved profitably to let nodes collect information from others for a certain time period before they can forward their own information. This slows down the propagation of the flood wave, but doing so, the nodes have to send less messages and the network is less congested. This effect can be explained with a simple example: Imagine the situation in figure 3.1: Two nodes A and B receive the flood wave from the sink at the same moment. They both forward it so that node N receives it. If for example link distance (in meters) is a metric in potential function, node N has a better path (via A) and a worse one. Which of the two paths is first known to N depends on which node succeeds first in reserving the medium. If now there were no broadcast delay, node N may receive the path over B first and forward that instantly. Right after that it would find an even better path over A and forward it again. But if N waits for a certain time and forwards only the best path it had received during that time, it could save a message (and so would all further nodes that come after N in the tree). In fact, simulations have shown that even a large waiting time of 1 second at each node could speed up the overall duration of the routing phase remarkably compared to non-delayed flood propagation. More details can be found in chapter 4.4.1.2

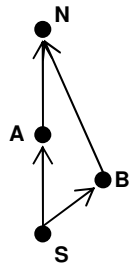

Figure 3.1: Flood Propagation

3.2.3 Simple Hop-Count Based Algorithm

The first version of the routing algorithm developed in this work uses the number of hops between a node and the sink as the only metric to calculate the node's potential. The potential function is defined as follows

$$\varphi(h) = 1 - \frac{h}{M+1} \qquad (1)$$

where h is the number of hops from node n to the sink and M is the maximum number of hops allowed in the network. In our WFDN application, M is given by the maximum delay for status messages and the size of the collection window Δ_m (see section 2.4). It is obvious that this function is monotonically increasing towards the root of the tree so this is a valid potential function.

As described above, flood propagation is used to let each node find out its hop count and potential. The nodes store all neighbors with a potential value at least equal to their own in a table with possible parents. This excludes loops, as two nodes with equal potential cannot be child and parent of each other.

PKT TYPE	SEQ #	BCAST ID	SRC ADDR	LAST ADDR	SINK ADDR	HOP CNT	METRICS

TYPE=ROUTING

Figure 3.2: Routing Broadcast Frame

Figure 3.2 shows the structure of the routing information packet that is broadcasted by the nodes.

- SRC ADDR defines the node that initiated the routing (normally the sink, except for a local rerouting after a link break)
- LAST ADDR is the node that broadcasts that packet
- SINK ADDR is the address of the sink
- HOP CNT is the number of hops from LAST ADDR to SINK ADDR
- METRICS may contain alternative metrics that are needed to calculate the potential (e.g min. RSSI along path or remaining energy). This field is not used in this version of the algorithm.

With this information, all receiving nodes can compute the potential they would have if using this node as a parent.

As soon as all nodes have found their optimal parent, the routing phase is finished and there is no further routing related traffic as long as all links are stable or until the next routing phase is initiated by the sink. Of course, some link may be unreliable and a node may fail to transmit a message to its favorite parent at some time. In this case, the node has to switch to the next best parent in its list. If the hop count or potential of the node changes after switching parents, the node has to inform all of its neighbors about its new potential and hop count (for the WFDN application, it is a precondition that every node knows its exact hop distance to the sink). The neighbors then have to update their tables with the new information or remove the node from their tables if the new potential is worse than their actual potential. A (slightly simplified) schematic of the actions a node has to perform on arrival of a routing information packet is presented in figure 3.3:

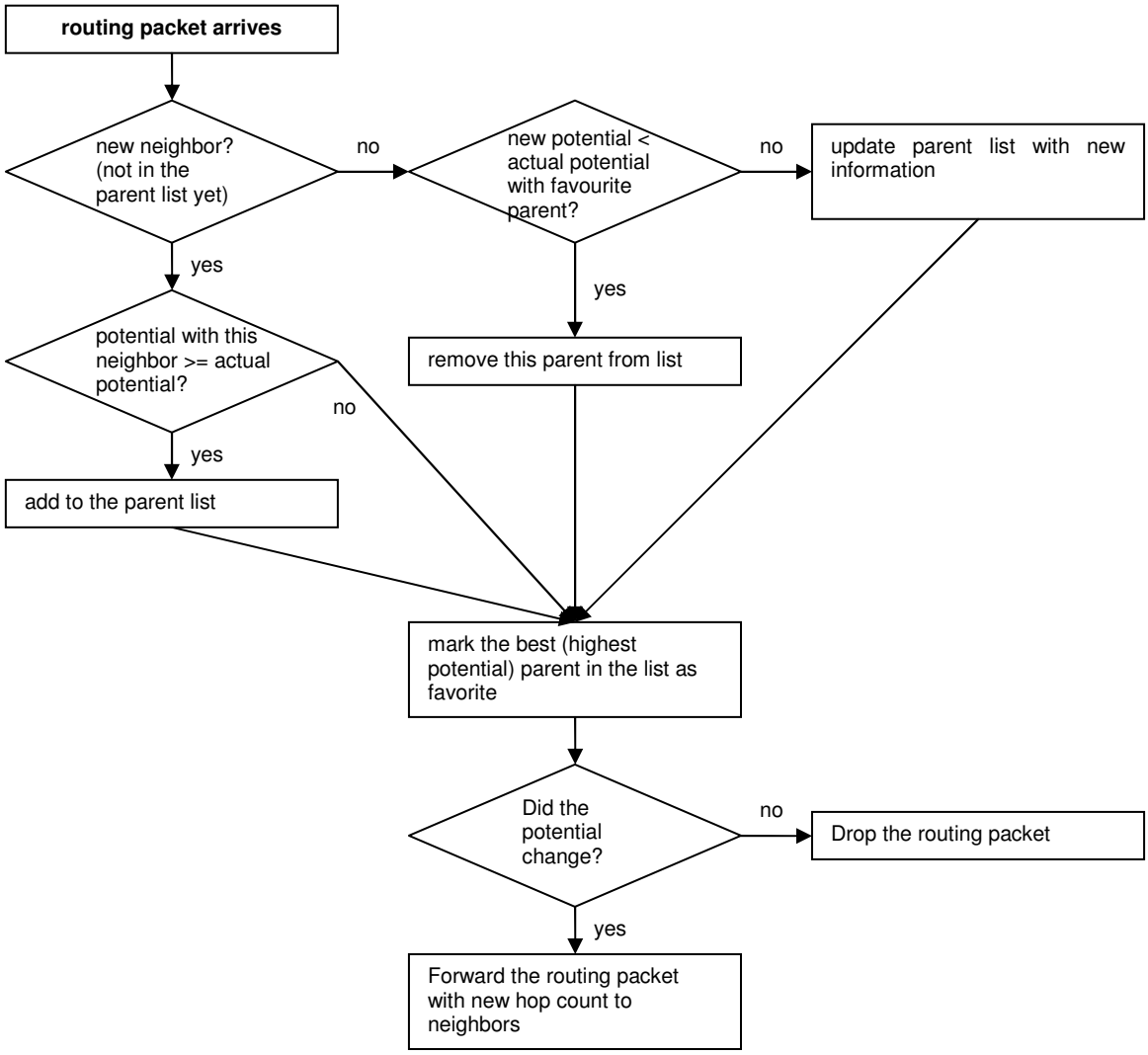

Figure 3.3: Process flow for incoming routing information

As mentioned, a complete routing can only be initialized by the sink. If the sink starts a new routing, it has to send out a routing information packet with a new sequence number. If the nodes see a new sequence number, they remove all entries for their parent tables. This helps avoiding outdated entries staying in the tables too long.

3.2.4 RSSI-Based Routing Algorithm

The above mentioned hop-count based algorithm is very simple and doesn't need much time to construct a complete routing tree. If there are no collisions, each node has to send only one routing packet. However, this algorithm has one big disadvantage: A node may be located just at the edge of the communication range of the sink. Of course, it chooses the sink as the favorite parent (as the hop count is only 1), but as soon as there is a slight change in the environment between the two nodes (e.g. a door closes), the link between the sink and this node is broken. The node recognizes this in the moment it has to forward some status messages from the sensor application. Now it has to switch to another parent and inform all of its neighbors about that action (start a local rerouting), which causes traffic (and hence energy consumption) as well as delay to the forwarding of the status messages.

Because reliability is very important for fire detection sensors, the second version of the routing algorithm version evaluated in this thesis tries to avoid such weak links already in the initial phase of routing. This is done using RSSI (Receive Signal Strength Indication). The receiver tells the routing protocol about the signal strength of an incoming message, and the routing protocol uses this information to create the routing tree. The main idea in this RSSI-based algorithm is to increase reliability by maximizing the signal quality of the worst link along the path to the sink. The algorithm is based on the principles of the hop-count based version, the main difference is the function to calculate the potential. This function is now defined as

$$\varphi(S_{min},h) = f_q(S_{min}) - \frac{h}{M+1} \quad (2)$$

where S_{min} is the minimal signal strength of all links on the path between node n and the sink and $f_q(S_{min})$ is a quantization function that converts S_{min} to a limited number of integer values (more about this in section 3.2.4.1). For example, $f_q(S_{min})$ can take the values 1, 2, 3, or 4 (4 is the best). The second term is equal as in the previous version and takes on values smaller than one. It is used here to guarantee that the function is strictly increasing, which makes the concept of field-based routing applicable.

The routing algorithm basically works still after the concept described in figure 3.3, but with a few alterations. The routing information that is sent by a node N contains now its hop count and its value of S_{min}. After checking if the link to node N is worse than S_{min}, a receiving node can compute its own potential with N as a parent.

A node inserts all neighbors to its parent list that have either

- a better or equal potential value OR
- smaller or equal hop count.

This guarantees that a node does not insert its own children to the parent list. Accordingly, a node has to remove a parent from its list if that one sends an update with none of the above conditions fulfilled, i.e.

- it has now a worse potential AND
- a bigger hop count.

The necessity to remove such parents can under specific conditions (e.g. link breaks, delayed routing packets due to collisions, see example in fig 3.6) cause nodes ending up with an empty parent list, even if there were available parents. To avoid such scenarios, nodes have to reply and broadcast their routing information as soon as they receive a routing packet from another node indicating that this node has no parent (hop count ∞).

Another special case comes up with the ability of the nodes to add other nodes to their parent list with a hop count equal to their own. This means that they can add nodes from the same sub-tree, i.e. nodes that are children of the same parent as they are themselves. Imagine the following situation (figure 3.4): Two nodes N have the same parent P with a potential of 3.9, and both of them have stored each other as an alternative parent with just one additional hop, potential 3.8. If now the parent updates its potential to 1.9, they both would remove P from their list and choose each other as new parent. Then they would forward the information that they now have one hop more. They would both hear that, and add another hop, This would go on until they reach the maximum hop count and their potential drops to zero. Only then they will switch back to their original parent P with a potential of 1.9. To avoid this count-to-infinity problem that causes a lot of unnecessary traffic, a special procedure is executed whenever a node receives routing information from its actual favorite parent with lower potential. Instead of removing this parent P, the node N removes all other nodes from its parent-table that may be in the same sub-tree with root P as itself. This way, the new potential

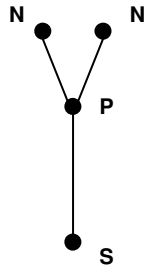

Figure 3.4: Sub-tree with root P

information of the root just propagates through that sub-tree, and each node can update its potential. During this process, nodes that were initially removed from the tables will automatically reappear with correct potential values.

The following illustrations (fig 3.5 and 3.6) give some simple examples on the behavior of the RSSI-based algorithm.

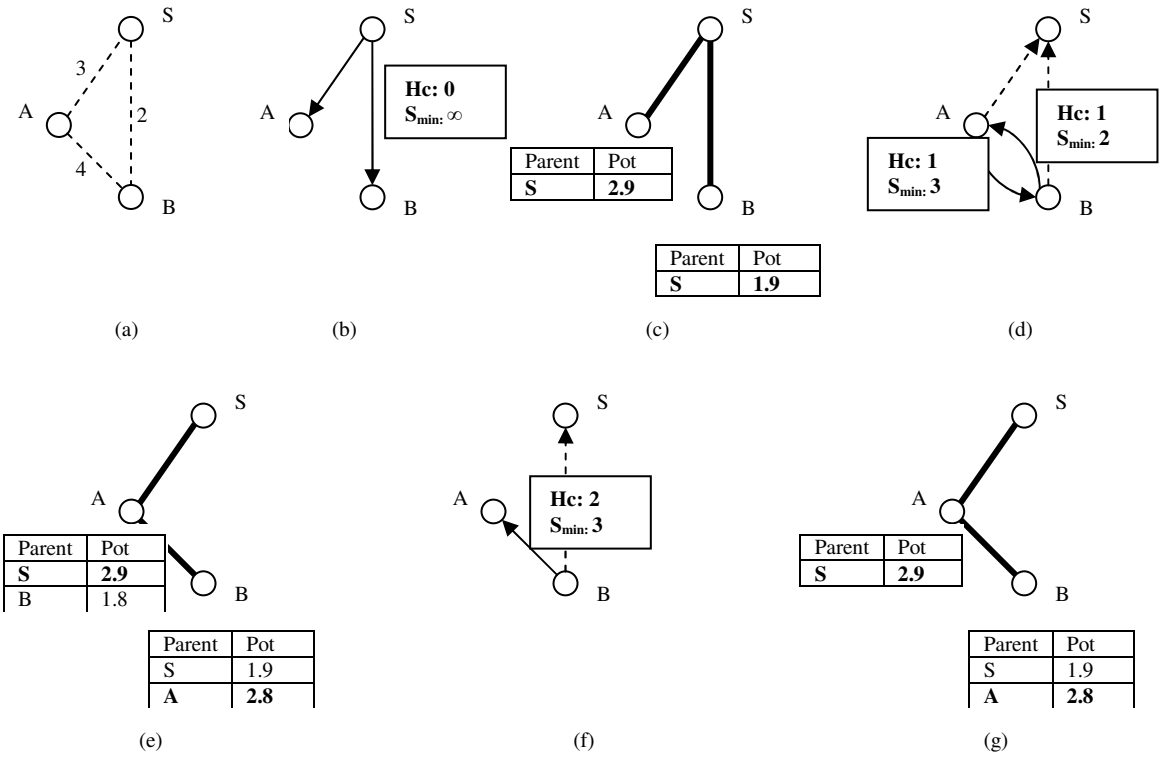

Figure 3.5: Routing tree construction (example)

Figure 3.5 demonstrates the construction of the routing tree in an arrangement of three nodes. The link quality between the nodes is indicated in fig. (a) and is already converted to discrete levels. The routing is initiated by the sink in fig. (b). The sink sends its hop count (0) and S_{min} together with a new sequence number, which causes the receiving nodes to flush their parent lists. The two sensor nodes both enter the sink into their parent list and mark it as their favorite parent (c). Now both nodes have to forward the packet with their hop count and S_{min} values because they have changed their potential (d). In (e) they have both entered their neighbor into their parent list (they can enter them because the hop count is equal but not larger than their own). The hop count based algorithm would stop at this place as both nodes would stick to the sink as favorite parent. Here, node B recognizes that it can reach a higher potential using A as parent and thus marks A as its favorite. Again, B has changed its potential (from 1.9 to 2.8) and forwards this information (f). Now node A recognizes that B has a lower potential AND a higher hop count than itself, which means that it has to delete B from its list to be sure that no children are contained in it (g).

For that kind of routing to work properly, it is a precondition that link qualities stay at a more or less constant level during the initial routing phase. Some unstable link could prevent the algorithm from stabilizing which would increase the duration of the routing phase and hence energy consumption. It is therefore probably the best to schedule routings at night, when human induced disturbance is lower.

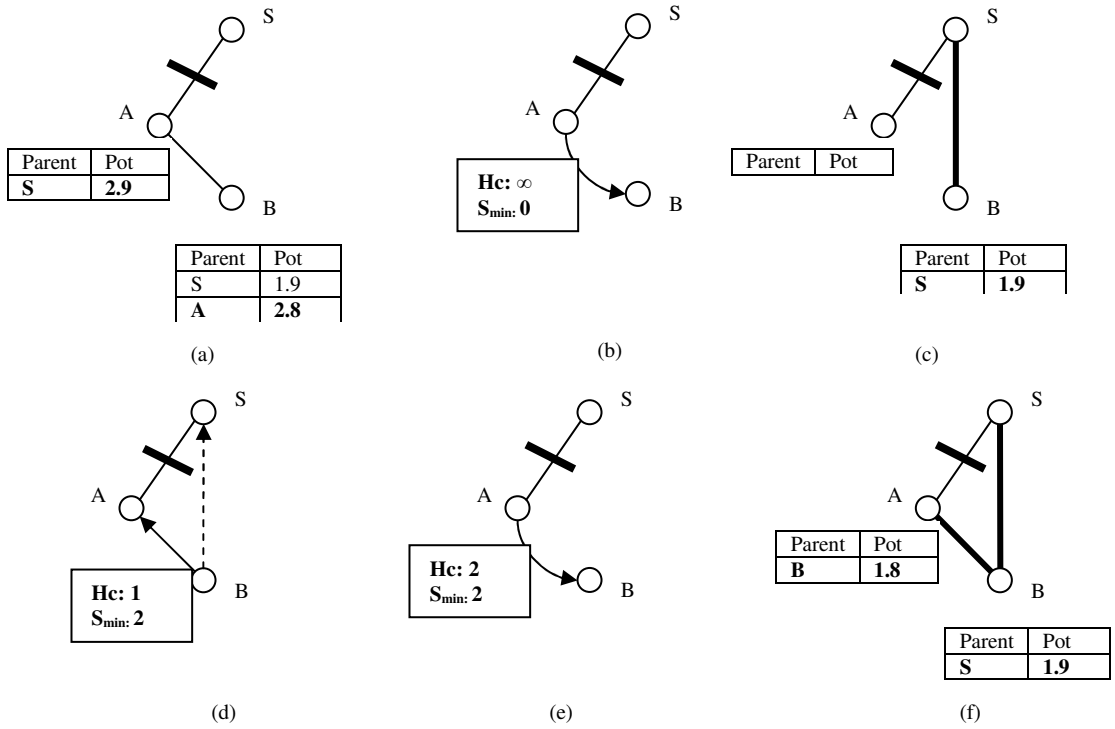

Figure 3.6: Rerouting after link break (example)

The sequence in figure 3.6 describes the procedure after a link break. The initial point is the system of figure 3.5, but the link between node A and the sink is now disturbed by some obstacle (a). Node A gets aware of that situation as soon as it has to transmit any packet to the sink. Ideally, node A would be able to switch to an alternative parent, but at this moment, there is no alternative. Node A has to inform its neighbors about this situation and sends a routing information with hop count ∞ (b). Node B receives this info, removes node A from its list and switches to the sink as parent (c). Nodes receiving a routing information with hop count ∞ have to reply, so node B broadcasts its hop count and S_{min} (in this case, B has changed its potential, so it would have sent the information anyway) (d). Node A can now add B to its list and forward its updated hop count/potential (e). This has no further effect as B just drops this packet because of the worse hop count and potential. The procedure ends with both nodes having a stable route to the sink again (f).

3.2.4.1 Quantization of the RSSI Values

As mentioned above, the RSSI values from the transceiver have to be converted to a limited number of levels to be used in the routing algorithm. It is for example possible to just distinguish between a good and a bad link. In this case, it needs only one threshold for the signal strength that separates the good links from the bad ones. A good link would be level 2, and a bad link level 1 (if the signal strength is too weak to have a transmission, there is no link at all). Of course, it may make sense to distinguish between a bigger number of levels. Simulations have shown that the number of levels is a very sensitive parameter of this routing algorithm. The more levels are used to categorize the link quality, the better (in terms of link quality) is the resulting routing tree. The prize to pay for this is a bigger number of messages (=energy consumption) that have to be sent until the routing tree stabilizes.

The case with only one level is a special one. One RSSI Level means that all links are treated the same in terms of signal strength. The only factor influencing a node's potential is hence its hop count, which reduces that single-level case to the simple hop-count-based algorithm.

Details about simulation results regarding this issue can be found in chapter 4.4.1.1.

4 Evaluation

4.1 Simulation Platform: GloMoSim

Simulations of the routing algorithm have been performed using GloMoSim (Global Mobile System Simulator) [4], a network simulation library offering a strictly layered communication protocol stack. For the propagation model as well as for radio and MAC layers, the already existing WiseNet implementation in GloMoSim was used. On application level, a simple version of the WFDN application layer has been created.

Layer	Protocol
Application	WFDN
Transport	UDP
Network / Routing	IP / RSSI-Based Routing Protocol
MAC	WiseMac
Radio	WiseNet Transceiver

Table 4.1: Protocols used in simulation

As GloMoSim is a strictly layered network simulator, network and transport layers have to be used in simulations. In these simulations, the influence of UDP and IP protocols is limited to added network and transport headers to application packets. It is clear that in a hardware implementation of the sensor nodes, network and transport layers could be left away and the WFDN application would communicate directly to the routing protocol.

The following table gives a short overview on the most important parameters that were used on MAC and Radio layers:

RADIO-FREQUENCY	866 MHz
RADIO-TX-POWER	3.5 dBm
RADIO-ANTENNA-GAIN	-2.0 dB
RADIO-RX-SENSITIVITY	-101.0 dBm
RADIO-RX-THRESHOLD	-85.0 dBm
RADIO-BANDWIDTH	25 kbps
RADIO - Setup Time	800 us
RADIO - TurnAround Time	400 us
WISEMAC-WAKEUPPERIOD	200 ms
WISEMAC-BACKOFFWINDOW	50
WISEMAC-MEDIUMRES WINDOW	10

Table 4.2: Simulation Parameters

4.2 Topology

In all simulations, the simulation area was divided into equal sized cells each containing exactly one sensor node. The position of the sensor node within its cell was random. The propagation model used was free-space-like up to a distance of 1 meter, and came with a larger decay index of 3.5 for larger distances to get a bit closer to indoor conditions. This resulted in a radius of the transmission range of roughly 34 meters.

Unless otherwise noted, the data sink was always positioned in cell 0 (top left corner).

4.3 Average Potential as Indicator of the Routing Quality

To compare different routing trees, some quality indicator is needed. As one of the most important requirements for a WFDN is reliability, the signal strength of the links, especially of the weakest links, is a good measure. The potential function that is used in the RSSI based routing algorithm calculates exactly that signal strength of the weakest link. In the simulation model that was used here, all nodes store the signal strength of their worst link on the path towards the sink in the parent table. At the end of the simulation, a potential value was calculated for each node. To compare different versions of the algorithm (e.g. hop count based version/1 RSSI level vs. multiple RSSI levels), the potential function that was used at the end of the simulation was always based on 10 RSSI levels, regardless of the number of levels that had been used to construct the tree. Hence, an average potential of all nodes indicates the quality of the links that are in use.

The quantization function used in this model uses the same decay index (3.5) as the function to calculate the path loss. This means that a link that is at the edge of the transmission range is converted to a 1, a link over a distance of about half of the transmission radius is a 5, and a very short distance link is a 10.

Besides the fact that a weak link is more likely to break completely, the error probability increases with weaker links. The BER (bit error rate) for FSK is $BER = Q(\sqrt{SNR})$. The Q-function approximately decreases exponentially, and as the signal strength (and hence the SNR) also decreases with the power of 3.5 with bigger distance, a node with a potential of 5 has a much lower BER than a node with potential 2, even if it needs more hops to reach the sink.

These are the reasons why the average potential of the nodes was calculated after each simulation run and used as an indicator for the quality of the routing tree.

4.4 Simulation Results

Unless otherwise noted, all results are based on data from 200 simulation runs. For all results, the mean and the 95% confidence interval have been calculated. For measurements where the confidence interval was not negligible, it is indicated in the charts. For measurements of hop count and average potential, the confidence interval was normally around 1% of the mean and is therefore not displayed in the charts.

4.4.1 Variation of Routing Algorithm Parameters

There are basically three important parameters that can be chosen for the RSSI-based routing algorithm. The first one is how often the sink initiates a new routing phase. It is obvious that this parameter has a direct impact on the energy consumption of the nodes as routing causes additional traffic. But besides that, it doesn't have any interesting effects on the routing itself.

4.4.1.1 Number of RSSI Levels

The second parameter is the number of RSSI levels that are used to calculate a node's potential. This one has a much bigger influence on the behavior of the routing algorithm. The chart in figure 4.3 shows the average potential the nodes have as well as the sum of all messages that have been sent by the nodes during the routing phase. It can clearly be seen that the quality of the routing tree (indicated by the average potential) improves remarkably, but only up to about 4-6 RSSI levels. Here, the algorithm needs about 3-4 times more messages to create the routing tree than the hop count based version (1 level). But there are still only about 120 messages, which makes about 4 per node. Beyond 6 RSSI levels, there are only small improvements whereas the energy consumption (indicated by the number of messages) still increases.

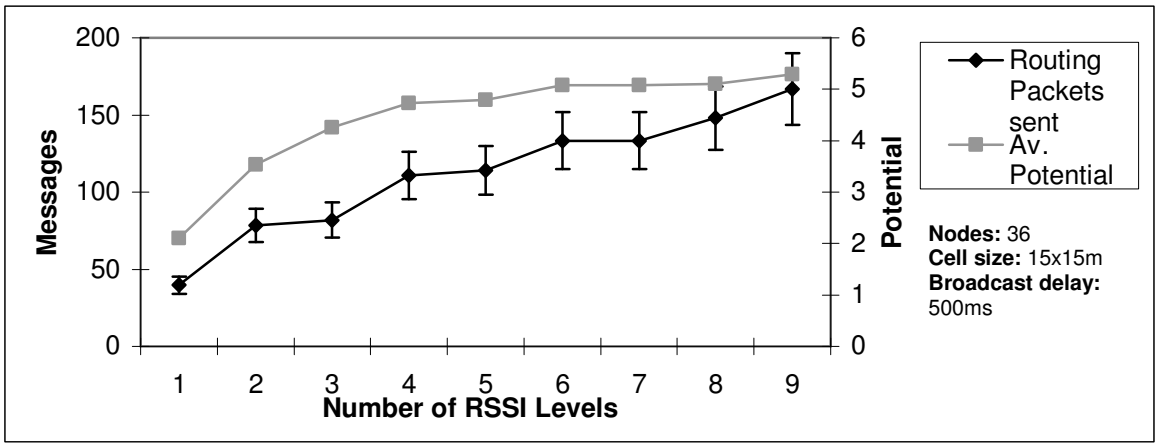

Figure 4.3: RSSI levels

Another point that could be observed during this simulation was that the average hop count of the nodes increases from about 2.5 with one RSSI level (hop count based algorithm) to 5.3 with nine levels. Unfortunately, the initially proposed parameters for the application layer (collection window 3s, maximum delay 30s) limit the maximum hop count to 10. This leads to the conclusion that the maximum number of nodes is not much higher than the here used 36, otherwise that hop count limit will be reached. The duration of the routing phase increased similarly to the number of messages sent. In this simulation it went from 5.5 seconds with 1 level up to 23 seconds with 9 levels.

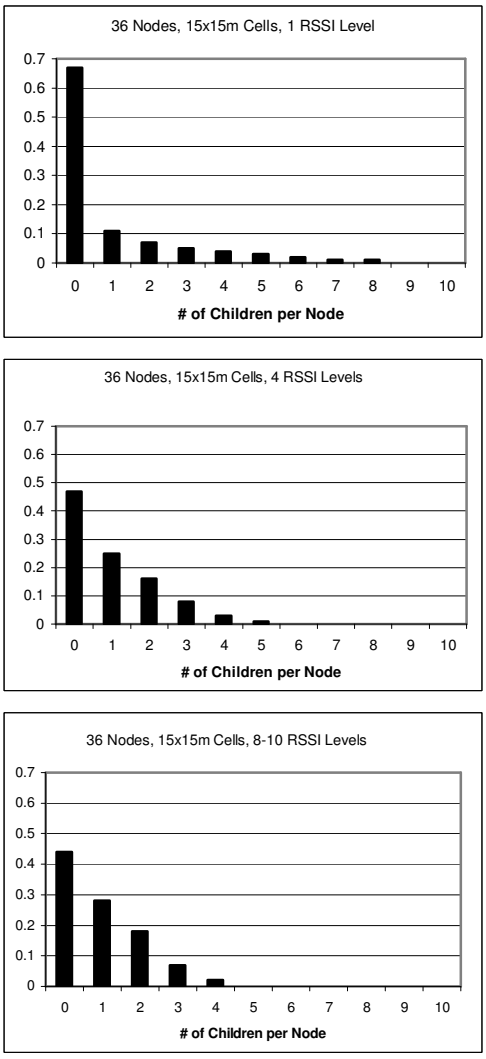

Figure 4.4: Distribution of the Children

The RSSI quantization has also an influence on the number of children the nodes have. This effect is visualized in figure 4.4, which indicates the distribution of the number of children per node after 200 simulation runs. The charts show that with the hop count based algorithm, two thirds of all nodes are leafs, i.e. they have no children, while over 10 percent of the nodes have more than 3 children, which produces an increased risk of collisions. With more RSSI levels, these parents with many children disappear. The reason for this is the following: If there are many nodes distributed around a parent node, they are very likely to choose a multi hop connection to that parent instead of a direct connection, assumed they are not all at the same distance to this parent. This results in fewer nodes with no children and fewer nodes with many children.

The illustration in figure 4.5 shows the differences between the RSSI based and hop count based routing on the basis of one particular arrangement of 16 nodes.

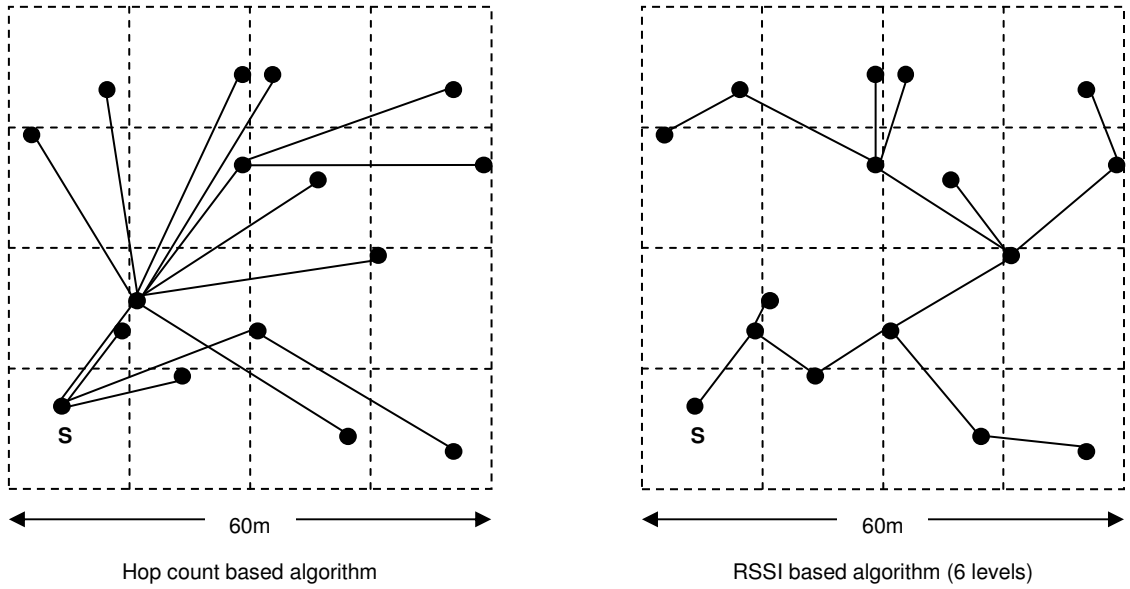

Figure 4.5: Hop count based vs. RSSI based routing tree

In the hop count based version, all nodes just choose the shortest path (in terms of hop count) to the sink, which results in rather long link distances and, more important, in a single node having 8 children. As all of them would want to transmit their status messages at the same time, the probability of collisions gets very high. This makes it likely that not all 8 children will be able to transmit their messages within the 3 seconds of the collection window.

The RSSI based algorithm on the other hand produces a much better structured tree with remarkably shorter link distances. Also, the maximum number of children now is only 3, which keeps the probability of collisions reasonably low. The drawbacks of this version are the additional effort during the routing phase (47 packets in 10 seconds vs. 17 packets in 3.7 seconds) and the larger hop counts causing larger delays for the status messages.

4.4.1.2 Broadcast Delays

The third parameter of the routing algorithm that was investigated was the delay that is introduced in the flooding of the routing information. One would expect that a larger broadcast delay would result in a longer duration of the routing phase. Figure 4.6 shows that in fact, the contrary is true.

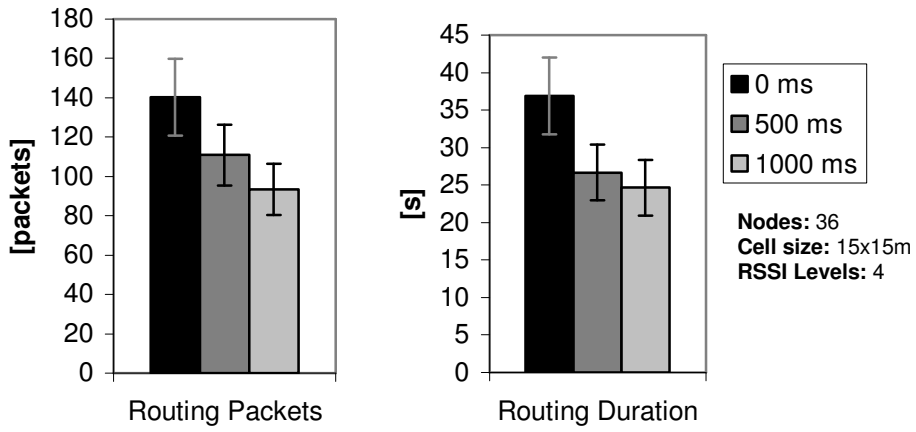

Figure 4.6: Broadcast delays

With a delay of 1 second, the routing needs over 30 percent less time and routing messages than without delay. The result of the routing (average potentials and hop counts) was the same in all three cases. Note that in all cases (even in the undelayed version), a random broadcast jitter between 0 and 100 ms is added to the delay. This is needed to mitigate the problem of collisions.

It is clear that this effect appears only with a certain number and density of nodes. If the nodes had only a very low number of neighbors, the additional delay could not lower the number of messages and would therefore rather increase the routing duration.

4.4.2 Variation of the Topology

For reasons of simplicity, the topology for these simulations was limited to the above described arrangement of cells containing the sensor nodes. Under these preconditions, there are three parameters that were investigated: The position of the sink within the simulation area, the absolute number of sensor nodes and the size of the cells in which the nodes are placed. The latter is equal to the density of the sensor nodes.

4.4.2.1 Node Density

It has to be mentioned that the radius of the transmission range of the nodes in these simulations was about 34 meters, which may not exactly correspond to the actual transmission range under indoor conditions. The node densities that are indicated here have therefore rather to be seen in relation to the transmission range (about 3630 m^2). Anyway, the crucial point for the behavior of the routing algorithm is not the absolute value of transmission range or cell size, but the number of neighbor nodes that are present within another node's transmission range. Of course, in a fire detection network, the distance of the nodes is basically determined by the topology of the building and the sensing range of the fire detector device, so it is the transmission power that has to be adapted to get the transmission range into a reasonable relationship with the node distances. Varying the cell size or the node density, respectively, has the same effect as varying the transmission power.

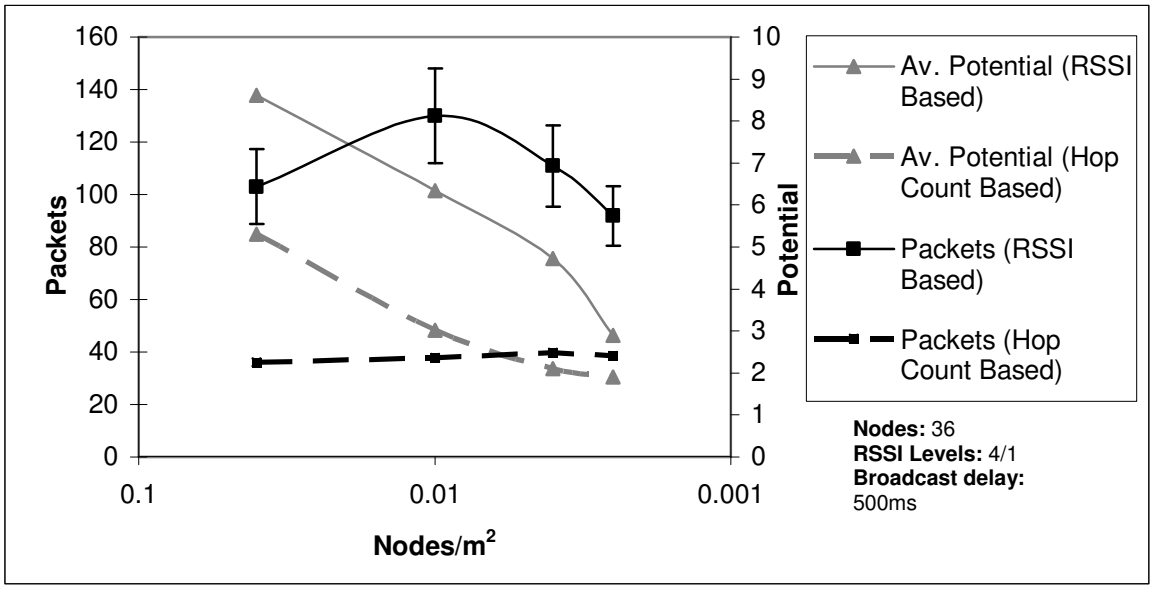

Figure 4.7: Node density

Figure 4.7 shows the routing effort for a 36-node network depending on the node density or cell size. At first glance, it may seem surprising that the number of routing packets decreases for high node densities as well as for low densities. The explanation for this effect is the following: If the nodes are close together, the RSSI values of the links get more and more similar, so the routing algorithm classifies many of the links equally. Also, the nodes can

choose more direct and shorter paths to the sink, the average hop count decreases drastically. That causes the routing algorithm to stabilize quicker. The overall costs and energy consumption to cover the same area as with a lower node density would of course be much higher because more nodes were needed. In the arrangement with larger cells, the algorithm has just less alternatives (the nodes have less neighbors) which is the reason for lower routing effort in this case. The comparison with the hop count based algorithm shows however that the gain in link quality disappears when the node density becomes too low.

The conclusion of these observations is that the RSSI-based routing algorithm works best with node distances that are about a third to half of the radius of the transmission range. This guarantees that all nodes have a reasonable number of neighbors. With lower densities, the additional routing effort of the RSSI based algorithm brings no big improvement in the average potential.

4.4.2.2 Number of Nodes

The WFDN application is currently designed to support a maximum of 127 nodes. Simulations have shown that the routing effort doesn't vary much with increasing networks. In figure 4.8, one can see that the number of routing messages increases quite linearly over the range of 100 nodes.

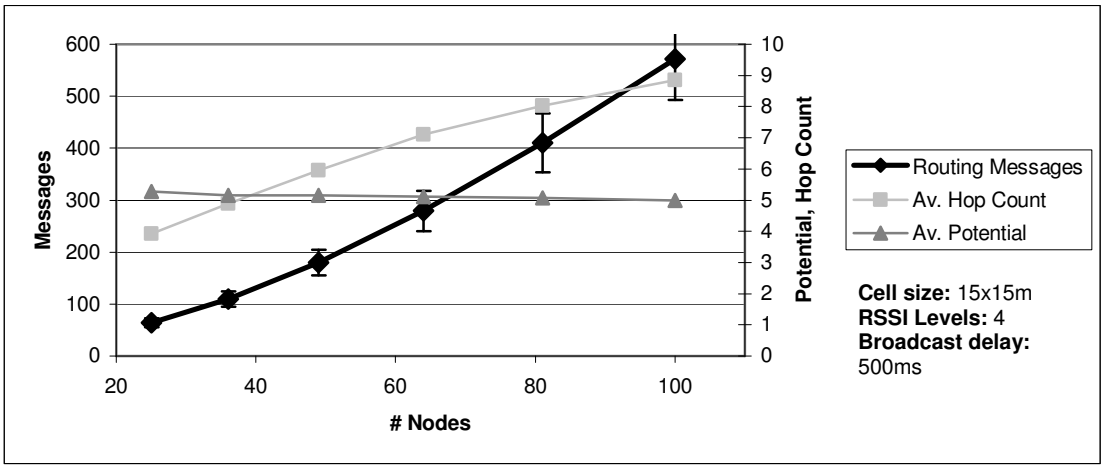

Figure 4.8: Size of the Network

Note that hop count of 10 has been disabled for these simulations. The maximum status message delay of 30 seconds would, in connection with the collection window size of 3 seconds, limit the hop count to 10 hops. Using 4 RSSI levels, these 10 hops were already reached with about 40 nodes. A possible solution to this problem might be to reduce the collection window size and randomize the starting point of the status message transmission within the collection window. If not all nodes would try to send their status message at the beginning of the collection window, the risk of collisions would be lowered considerably.

The fact that the average potential decreases only very slowly shows that all nodes still have good links, even if the network gets bigger and some nodes are farther away from the sink.

4.4.2.3 Position of the Sink

It is clear that a good position of the sink helps reducing the routing effort and the average hop count. Figure 4.9 compares a situation with the sink in the middle of the simulation area with the standard situation having the sink in a corner.

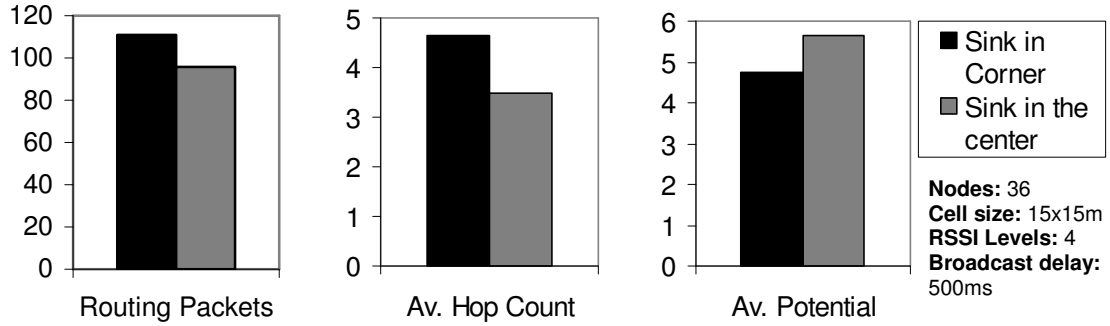

Figure 4.9: Position of the sink

Especially in larger networks with a bigger number of nodes, the hop count can be lowered with a good placement of the sink, whereas in smaller networks, this effect is clearly reduced.

4.4.3 Simulation of Link Breaks

To simulate broken links in this section, a simple method was used. If a link was declared broken, all packets for that particular link were just dropped at the radio layer of the sending node. As described above, nodes recognize broken links as soon as they fail to transmit a packet over that link. In such a case, they remove the corresponding parent from their list. Then there are three possibilities: Either they have an alternative parent that doesn't change their potential, a parent with lower potential, or their list is now empty. In the second and third case, a local rerouting is initiated. This rerouting affects only nodes that are in a sub-tree behind the broken link. Figure 4.10 shows an example of how the routing tree changes after a broken link.

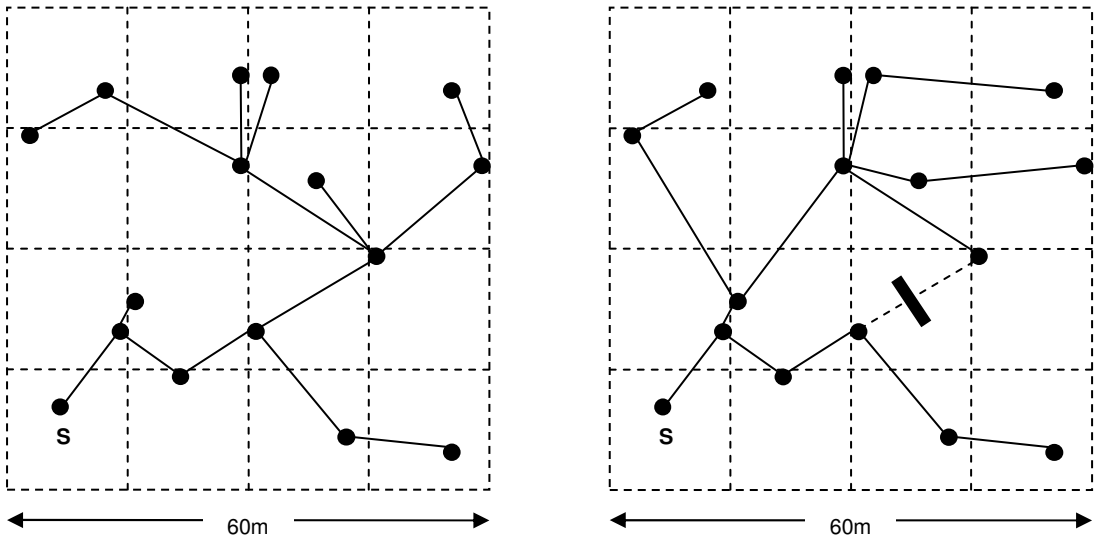

Figure 4.10: RSSI based routing tree after link break

The starting point is the situation from figure 4.5 but one link is made unusable. After the bad link is recognized, it takes the algorithm about 3.6 seconds or 13 routing packets to stabilize to a new routing tree.

Table 4.11 shows the results of 200 simulation runs on a 36-node network. In every run, one random link was removed and the time and number of messages the algorithm needed to stabilize were measured.

Number of Nodes: 36 **Cell Size:** 15x15m **RSSI Levels:** 4 **Broadcast Delay:** 500ms	Min	Max	**Mean** (95 %-Confidence Interval)
Number of Packets sent	1	38	**6.0** ± 1.25
Duration of Rerouting [s]	0.2	10.6	**1.8** ± 0.26

Table 4.11: Rerouting duration

It is clear that these simulations don't reflect real dynamic indoor environment. However, they show that the algorithm is able to react quickly and with a small effort whenever a link becomes unusable. The rerouting effort depends largely on the number of nodes that are in the sub-tree behind the broken link.

4.4.4 Energy Consumption

Energy consumption in this kind of network is directly related to the amount of time a sensor node spends in transmitting and receiving mode. The here used WiseNET transceiver consumes about 20 times more power in transmitting mode than in receiving mode, so for a rough estimate, it will do to concentrate on the transmitted packets. As simulation results showed, a node transmits an average of 3-6 routing packets during the routing phase if the RSSI based algorithm is used (only 1 packet with the hop count based version). Assuming a routing takes place once every day, there are 3-6 routing packets per day. In contrast to that are the status messages that are transmitted by the fire sensor application. Including the preamble, a status message and a routing packet are of comparable size. Each node sends one status message every 30 seconds, which makes 2880 packets in a day, plus, because the status messages are unicast packets, 2880 ACK packets. This indicates that the overhead traffic produced by the routing protocol has a reasonable low impact on the overall energy consumption of the nodes.

5 Conclusions

The topic of wireless sensor networks comes with a variety of challenges, one of the most important thereof being energy efficiency. Energy efficiency has to be considered at all levels of the communication protocol staple, including the routing layer. This means for example that the overhead traffic caused by a routing algorithm has to be very low compared to the traffic needed for the actual sensor application. In addition to this, a routing layer has to deal with trade-offs that have been made in other layers for the sake of energy efficiency. In the here described wireless fire detection network for example, the collection of status messages in the application layer reduces energy consumption but gives the routing layer a maximum hop count it has to deal with. Or the MAC layer synchronizes nodes and lets them communicate only at certain time slots which asks the routing layer to keep the number of children per node as low as possible. At the end, there are several constraints, some of them even conflicting, and all of them should somehow be considered in the routing algorithm.

The algorithms presented in this paper are based on field-based routing, which prevents loops in the network and allows using different potential functions, depending on the requirements a routing algorithm has to fulfill.

To minimize the routing effort, a purely hop count based potential function is the simplest choice. Regrettably, the resulting routing tree has unfavorable properties for the purpose of a fire detection network, such as unreliable or long-distance links.

To put the main focus on link qualities, a potential function that includes a rating of the links between nodes is used in the RSSI-based algorithm. This one needs clearly more routing traffic in the initial routing phase, but avoids reroutings due to broken links at later times.

Different aspects of these routing algorithms have been investigated in simulation. One point that might not be so important at first glance but showed remarkable influences on the behavior of the algorithm is the number of levels used to describe the signal strength in the RSSI-based algorithm. It came out that 4-6 distinct levels are enough to receive a good routing result. More than 6 levels only caused additional traffic but didn't bring any improvements. One problem that remains is the limitation of the number of hops. This restricts the network to rather small sizes if the RSSI based algorithm is used. A reduced collection window size, together with randomized transmission schedules could probably allow larger hop counts without increasing the probability of collisions.

Simulations have also shown that the RSSI based algorithm is only effective if the nodes have a certain number of neighbors they can choose from. If the node density gets too low, the algorithm doesn't have enough flexibility to tap its full potential.

Some simple simulations of link breaks indicated that the algorithm is able to react quick and with a small amount of overhead traffic in case of changing environment.

Overall, the RSSI based version (using a reasonable number of levels) generated about 3-5 times more traffic than the hop count based version. Nevertheless, the clear advantages in the resulting routing trees (like link qualities, distribution of the children) make the RSSI based version a more suitable choice for a WFDN routing protocol.

List of Tables and Figures

References

[1] A. El-Hoiydi, et al., “WiseMAC: An Ultra Low Power Mac Protocol for the WiseNet Wireless Sensor Network”, Proc. 1st ACM SenSys Conf., ACM Press, 2003, pp. 302-303.

[2] A. El-Hoiydi, et al., “Wireless Fire Detector Network (WFDN) Protocol Specification”, CSEM Technical Report, 12. 4. 2005

[3] Vincent Lenders, Martin May and Bernhard Plattner, “Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach”, In the Elsevier Journal on Pervasive and Mobile Computing, Volume 1, Issue 3, p. 343-370, September 2005.

[4] X. Zeng, R. Bagrodia and M. Gerla, “GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks”, In Proceedings of the 12th Workshop on Parallel and Distributed Simulations (PADS ’98), Banff, Alberta, Canada, May 1998