CHALMERS

RFID in Wireless Sensor Network

Christer Englund
Henrik Wallin

Communication Systems Group
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, April 2004

EX034/2004

Abstract

This report begins with a brief introduction to sensor networks and RFID. The task given is to integrate RFID into a sensor network in order to, from a user point of view, extend the range of the RFID system. There are high requirements regarding the power consumption because the nodes in the network are supposed to be battery driven.

Most of the work is concentrated on the design and implementation of a sensor network protocol. The suggested protocol is a MAC protocol that is very energy efficient and broadcast oriented. It is based on a cluster structure and the nodes in the network haven't got specific addresses. The authors think this will make the network more scalable than it would have been with most existing protocols.

The designed protocol was tested both in real world with designed hardware and in a network simulator on a computer. Real world tests were only performed in small scale as the number of nodes were limited. The scalability with a larger number of nodes was tested in the computer simulator. Both the real world tests and the simulations showed that the protocol worked in the way it was intended.

Keywords: Sensor network, RFID, Radio Frequency Identification, TinyOS

Sammanfattning

Den här rapporten inleds med en kortfattad introduktion till sensornätverk och RFID. Den givna uppgiften är att integrera RFID i ett sensornätverk för att ur en användarsynvinkel utöka räckvidden hos RFID systemet. Hårda krav ställs på effektförbrukningen eftersom noderna i nätverket kommer att vara batteridrivna.

Huvuddelen av arbetet är koncentrerat kring design och implementering av ett sensornätverksprotokoll. Det föreslagna protokollet är ett MAC-protokoll som är väldigt energisnålt och broadcast-orienterat. Protokollet är klusterbaserat och noderna i nätverket har inga bestämda addresser. Författarna tror att detta kommer att göra nätverket mer skalbart än vad som hade varit möjligt med existerande protokoll.

Det designade protkollet testades både i verkligheten med tillverkad hårdvara samt i en nätverkssimulator på en dator. Verklighetstesterna genomfördes bara i mindre skala eftersom antalet noder var begränsat. Skalbarheten med ett större antal noder testades i simulatorn. Både verklighetstesten och simuleringarna visade på att protokollet fungerade på det sätt som var avsett.

Nyckelord: Sensornätverk, RFID, Radio Frequency Identification, TinyOS

Acknowledgments

We appreciate the opportunity we were given to conduct the work at Flextronics Design Gothenburg. A big thank you to our supervisor at Flextronics Design, Richard Menzinsky for his guidance and encouragement throughout the work. We also would like to thank our examiner and supervisor at Chalmers, Prof. Arne Svensson for answering the questions we have had and for giving feedback on the work.

Gothenburg, April 26, 2004

Christer Englund Henrik Wallin

CONTENTS

List of Figures

List of Tables

List of Algorithms

1 INTRODUCTION

In this report the reader is first given a short general introduction to *sensor networks* and *Radio Frequency Identification (RFID)*. Then these two techniques are combined to create a wireless self-organizing sensor network in which each node uses RFID technology.

1.1 Introduction to Sensor networks

Sensor networks were in February 2003 predicted by MIT's *Technology Review* to be one of the most important technologies in the near future [17]. Depending on what type of application the sensor network is designed for it has different features. Sensor networks are often self-organizing, has no central administration and responds dynamically to failures of important nodes in the network. Further they often manage movement of nodes in the network and have some sort of multi hop routing capability between the nodes. The actual routing algorithms are often simplified by the fact that the routing scheme is often predefined. This is because the objective with a sensor network is often to collect data from the nodes, and therefore each node just need to know how to forward data towards the data collecting node.

The goal is often to get the sensor network as autonomous as possible, which is important in application where for example automatic surveillance is preferred or where the environment is hostile to humans. Todays technique make it possible to build extremely small units that can be deployed as nodes in a sensor network and perform different tasks. Areas like surveillance and data acquisition are likely to be revolutionized when the cost of these networks decreases.

1.2 Introduction to RFID

RFID is another promising technique, and has been developed since the 1960s. In recent years the cost has dropped to a level allowing it not just to be used in new kind of applications, but also outperforming different existing systems.

As the name indicates RFID is used to identify objects using radio communication. The most common usage is to identify objects in for example a supermarket. Still, bar code systems outperforms RFID in this area because the cost of a bar code system is lower than that of an RFID system. The usage of RFID today is for example in road tolls for vehicle access, personnel access control and other security systems. These systems uses more of the potential that RFID offers in form of no line-of-sight required and operating distances of up to tens of meters depending on system used. This is pretty much where the market is today but RFID is continuously breaking

into new markets as the systems get cheaper and cheaper when volumes rise. [11]

Different applications have different demands on range, power consumption and so on. Therefore different types of RFID systems exist, where the frequency used, modulation scheme and other parameters differ between them. The main concept used in RFID is however similar for all systems. Each system consists of a reader, an antenna and a population of one or more *tags*. A tag is like a small memory module which can communicate with the reader and can be attached to objects one wants to track. The reader can search for a tag that is within range by sending out a predefined signal using RF. A tag that receives this signal responds back with its unique ID that has been preprogrammed in its memory. All RFID systems has this basic functionality, but most of the systems today also support multiple tag detection and the ability to write optional information to the tag memory.

The most commonly used carrier frequencies are 125 kHz, 13.56 MHz, 868 MHz and 2.4 GHz. Depending on which carrier frequency that is used, different methods for communicating between reader and tag exist. 125 kHz and 13.56 MHz uses inductive coupling to communicate whereas 868 MHz and 2.4 GHz uses electromagnetic coupling. The electromagnetic coupling is more efficient than inductive coupling implying that systems using electromagnetic coupling often have better reading range. The ability to propagate through solid material is depending on the carrier frequency, lower frequencies give better propagation than higher frequencies.

The tags can be either passive or active, meaning that active tags are powered with a power source of their own, while passive tags are powered by the energy radiated by the reader. An active tag has the advantage of longer reading distance at the cost of that battery replacement is needed after some time of operation. Passive tags is the opposite, they don't need replacement of batteries but instead have shorter reading range. The advantage of not having to replace batteries is preferable in many applications. The source of energy to power the passive tags is the magnetic or electromagnetic field that the reader emits. In the case of inductive coupling the reader uses an antenna to radiate an alternating magnetic field at a constant frequency. A coil in the tag induces voltage, which is used to power the electronics in the tag. By loading its coil the tag it is able to modulate the power drawn from the magnetic field. The reader can sense this and thereby demodulate the information the tag "sent". By using a predefined protocol the reader and tag can communicate by using the technique described. More information about how different RFID systems work can be found in [3].

Because there are so many different types of systems it is impossible to describe them all in this report. The system that is used in this project follows a standard called Philips I•CODE. This standard uses the 13.56 MHz carrier with inductive coupling, has ability to read from and write to the tag and can read multiple tags simultaneously. More information about Philips I•CODE can be found in Appendix A.

1.3 Task Description

The normal reading range for the RFID system used in this project is only about 5-10 cm[1], which means that reader and tag cannot be physically separated more than this distance in order to function.

The purpose of this project is to from a *user point of view* be able to read RFID tags from distances that is well beyond that of the range of ordinary RFID readers. The basic idea is to connect the RFID reader to an RF transceiver which can forward information to and from the reader over distances approximately $100 - 200$ m depending on which RF transceiver that is used. By making the RF transceiver transparent to the user he would have the feel of being able to read tags that are $100 - 200$ m away from him. The design of a convenient protocol would improve the system by letting the user interact with multiple RFID readers simultaneously and thereby covering a larger area where tags are deployed. To be able to cover an even larger area and reach further, the RF transceivers must also be able to route information. The total system can then be seen as a network of nodes, which consists of an RFID reader and an RF transceiver. Each node shall be able to function as a router and pass on messages to the right destinations. In this network it is likely to have a central node from which the user is performing his actions and where to the information should be forwarded. An overview of the network can be seen in Figure 1.1. The routing nodes shall not need any special configuration compared to the ordinary nodes. Ordinary nodes shall take the routing responsibility and become routing nodes when they detect that outer nodes need routing assistance.

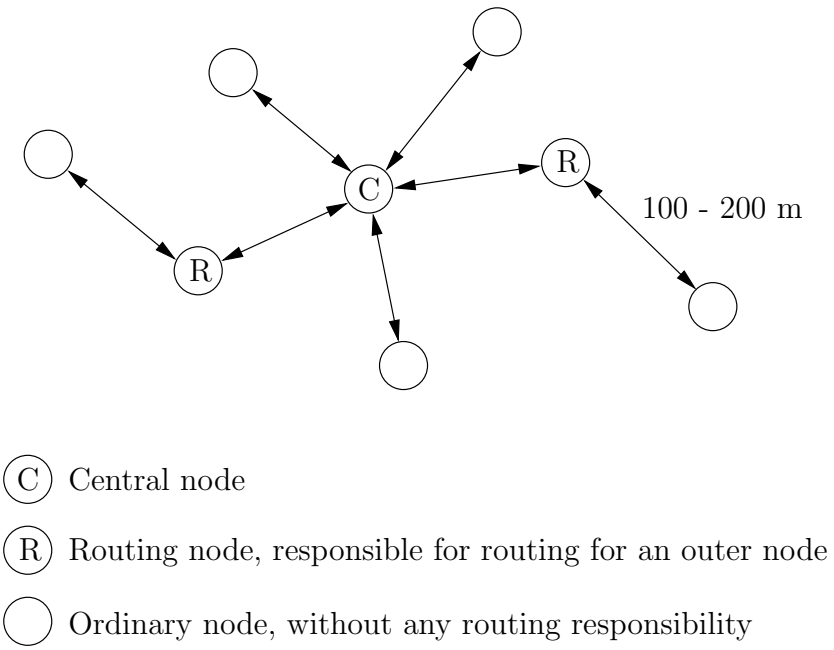

Figure 1.1. Overview of the network

[1]The reading range varies with the RFID antenna and RFID tag used and can be up to 1 meter for the I•CODE standard

Each node in the network except the central node is supposed to be battery powered and must be functional for years without replacement of the battery. This means that the nodes must be powered down when they aren't used and woken up again when they are to be used. Some sort of battery indicator shall signal when the battery level is low and the battery has to be replaced. The central node doesn't have this demand and it can be assumed that it will be powered by an external power source.

The nodes shall also be able to start up and run without any configuration. The number of nodes must not to be limited. That is, the complete system must be scalable.

The RFID system that is to be used in this system has to follow some well spread accepted standard. Philips I•CODE is the standard that is chosen as mentioned in Section 1.2. The reason why a well spread standard should be used is that other standard RFID readers that aren't part of this system should be able to read the tags in the system.

The RF device must operate in one of the license exempt frequency bands that are available (i.e 433 or 868 MHz). The regulations set in order to be allowed to operate in these bands of course have to be followed. Each link must reach at least 100 m, preferably 200 m, when line of sight exists. In the design process of the system, evaluation boards are to be used if possible, and no manufacture of PCB shall be needed.

1.4 Proposal

The main task in this project is to design and implement a protocol that nodes in the network can follow in order to fulfill the demands stated in Section 1.3. What is needed is a self configuring low power ad hoc network. The protocol design of these network are very application dependent and we need a protocol that

- Executes distributively.
- Provides loop-free routes, because looping means unnecessary communication.
- Minimizes routing communication.
- Minimizes energy consumption.
- Makes the network self-configuring.
- Minimizes response delay. This is necessary to preserve battery life in the units, when a unit needs to route information.

A MAC-layer protocol is proposed which is energy efficient and broadcast oriented, since all radio communication by nature is broadcast oriented. The protocol uses a

cluster structure with one master in each cluster that controls communication within its cluster. All nodes will use periodic sleep to preserve power, and the master is responsible for synchronizing sleep periods for slave nodes within a cluster. The routing layer will use the cluster structure to find a loop free route to the data collecting node.

The design of the protocol is very application dependent and the ideas behind it have been collected from different existing protocol designs and from own thoughts. References to work of others will be given throughout the text where it is appropriate.

1.5 Outline of report

In Chapter 2 some main assumptions, limitations and clarifications about the project are stated. The hardware used in the network is introduced and it is motivated why it was chosen. Chapter 3 explains the thoughts behind the proposed protocol and how it was designed. Chapter 4 describes the software implementation of the protocol and the other parts of the system. In Chapter 5 it is verified how the protocol works. Tests are made both in reality and in a simulator. Some information about the regulations for radio communication in the license exempt frequency bands is given, and it is also verified that the regulations are followed by this system. The theoretical power consumption is calculated in order to give a prediction of the battery lifetime of the system. The report ends with Chapter 6 where some thoughts and conclusions about the project are given and how the results agree with the task description given in Chapter 1. Some suggestions of future work are given in the end.

2 SYSTEM DESCRIPTION

From the task description given in Section 1.3 one can get a pretty good picture of how the final system will look like. The system proposed in this project will integrate RFID technology into an ad hoc network in order to easily collect information from multiple RFID tags spread over a large area. Each node in the network will have the ability to read RFID tags of a certain type and pass that information on to a central node in the network, from here on called *sink*. The sink will be connected to a PC from which the user shall be able to maneuver the data collection and analyze the collected data. This means that the user from a single point can gather data from RFID tags spread over a very large area. Further, the nodes will have the capability of route information to the right destination, which will make it possible to reach nodes even further away from the sink. The system will automatically integrate new nodes which are placed in the range of the current network. If crucial nodes are removed the network infrastructure will automatically be reconstructed, which means that the network will be completely self configuring.

2.1 Assumptions and limitations

It's obvious that the protocol that is to be designed must be rather complex in order to fulfill all the demands given. The protocol also has to be designed with the objective to be very power efficient in order to cope with the battery lifetime stated in the task description. A perfect protocol design would let the network have short latency, high data rate, good self configuring capabilities and very low power consumption. It is difficult to satisfy all these demands at the same time because they sometimes conflict with each other. This means that some of these demands have to be restrained in order to fulfill other demands. Both good self configuring and low power consumption are needed in order to be able to follow the task description. The data rate is not important in the sense that we can get shorter transmission time because the amount of data to be transmitted is relatively low anyway. But it is favorable to have a high data rate because a shorter transmission time will result in preserved energy and better power efficiency. The latency in the network is very much protocol dependent, and by allowing a little bit of latency in the network one can use different methods and techniques to make the system very energy efficient.

The system has a central node from which the user sends commands and receives information about the available tags. This node shall be connected to a PC to which it shall send the data acquired from the network. This means that every node must not be able to forward messages to a *specific* node in the network, but rather in the direction towards to or away from the central node. This assumption will make the routing algorithms much easier and saves a lot of overhead information that otherwise would have been needed.

2.2 Component choices

To be able to build an energy efficient system it is important to assemble hardware components that are power efficient themselves, otherwise it would be pointless in designing a protocol that is power efficient. The components used in this system have been chosen with the objective to be power efficient, but it should be pointed out that if this system would be manufactured in bigger scale, ASIC, FPGA or similar techniques would be used to integrate everything on a single chip. This would lower the energy consumption even more.

2.3 Tag+ unit

The nodes in the network include an RFID reader and an RF transceiver as has been mentioned in Section 1.3. However, some more components are needed in order to make the nodes functional. First, some sort of micro controller must be used in order to coordinate the different components in the node. The micro controller must be programmed with the designed protocol stack and handle the transmission and reception of data from the RF transceiver. The micro controller is also used to control the RFID reader, and signal to other devices to go into sleep mode when they are not needed.

Each node in the network consists of the following components:

- Micro controller
- RF Transceiver
- RF Antenna
- RFID Reader
- RFID Antenna
- Battery
- Components to connect components together

Together they form what we from now on will refer to as a *Tag+* unit, which can be seen in Figure 2.1 and Figure 2.2. The Tag+ is battery driven, which put requirements on low power consumption of all components included. The size of the Tag+ with the components we have chosen is approximately 55x40x20 mm, battery excluded. The schematics of how the Tag+ is connected can be found in Appendix B.

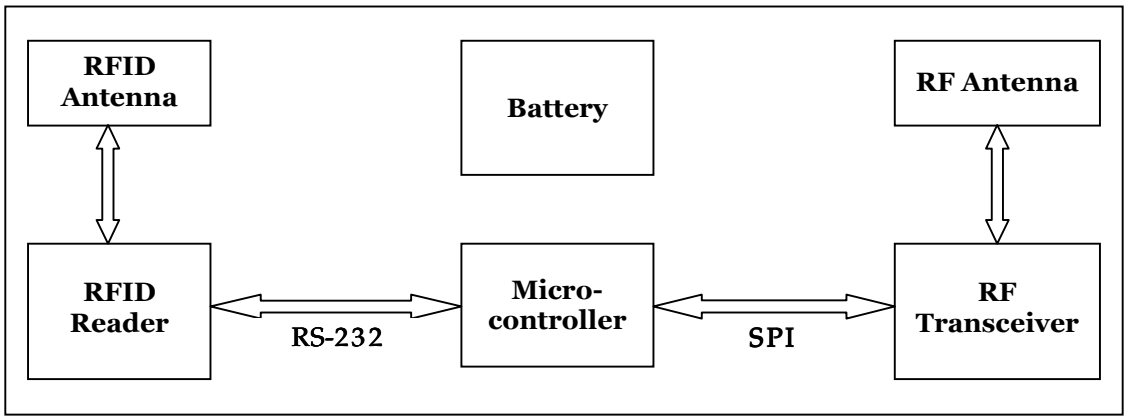

Figure 2.1. Schematic picture of Tag+ unit

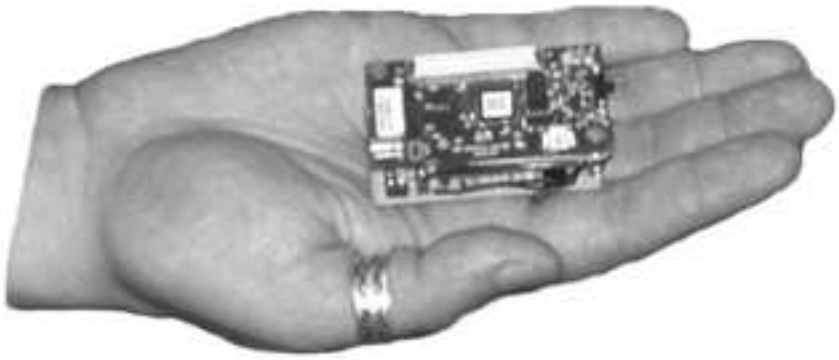

Figure 2.2. Tag+ unit

The micro controller is used to coordinate the Tag+ units. Clock frequency and memory size of the micro controller determine the amount of instructions that can be performed and thereby the complexity of the system. The Tag+ should have at least 32 kB of memory in order to fit the protocol, and the clock frequency should be high enough to be able to coordinate the Tag+ effectively. Though the clock frequency should be kept as low as possible because higher clock frequency corresponds to higher power consumption for the micro controller.

The RF transceiver must operate in one of the license exempt frequency bands. It should also have very low power consumption in order to save energy. The RF

antenna must be small but still efficient enough not to waste energy. The distance one can reach with an RF link is of course depending on the transmitted power. Outdoors where it is line of sight between transmitter and receiver it is often not a problem to reach $100 - 200$ m and still be able to cope with the maximum allowed transmitted power that is set by regulations in most countries. Indoors and in other environment with a lot of obstacles between transmitter and receiver it's not likely to be able to attain this range.

The RFID reader must follow the I•CODE standard. It must also be very small and power efficient. The RFID antenna to be chosen must be as small as possible but still have a range of about $5 - 10$ cm.

2.4 MICA2 Platform

MICA2 is a platform that includes a micro controller, an RF transceiver and an RF antenna, developed by Berkeley University, and manufactured and distributed by Crossbow Technologies. The MICA2 platform is designed for wireless sensor networks, which have many similarities to this project with respect to low power consumption and small physical size.

In this project we will use the MICA2 platform, which has an Atmega128L micro controller (from Atmels 8-bit AVR®-serie). This processor has 128 kB of In-System Programmable Flash, 4 kB EEPROM, 4 kB SRAM, programmable UART, SPI interface and is powered by a 3 V battery. It also supports power down modes, which can be used to lower the power consumption. The clock frequency used on the MICA2 platform is 7.37 MHz. The characteristics of this micro controller will easily fulfill the demands we have on the micro controller in the Tag+ regarding memory size and processing capability.

The SPI interface is used to communicate with the RF transceiver located on the MICA2 board. The RF transceiver is the CC1000 chip from Chipcon. It is configured to operate in the 433 MHz frequency band. It has very low power consumption and manages FSK data rates up to 76.8 kbit/s. It also supports automatic hardware Manchester coding and will then have a maximum data rate of 38.4 kbit/s. The transceiver has high sensitivity and RSSI (Received Signal Strength Indication) output. The data rate this chip manages should be high enough for the Tag+. The CC1000 chip seems to be one of the most power efficient RF transceivers on the market for the given data rate and will be convenient for the Tag+. A quarter wave monopole RF antenna without any ground plane comes with the MICA2 platform. This antenna makes a good balance between size and performance.

The MICA2 platform is also a good choice in the aspect that all the work wiring the RF transceiver to the micro controller is already done on the PCB, which makes the platform very small. It is also very easy to program the micro controller with the special programming board that comes with the evaluation kit.

2.5 Medio S002

To be able to read the I•CODE tags a reader that supports this type of tags is needed. Tagsys Medio S002 have support for a number of different types of tags, which can be found in [24]. It is a small size reader that is used in hand held RFID reader systems. The small size makes this reader a good choice for the Tag+. It supports TTL, RS232, RS485 and S422 as communication interface with the micro controller, and to preserve energy it can be put into stand by mode where the power consumption is low. The reader is powered by a voltage of 4 to 6 V and can be interfaced with at a maximum data rate of 38.4 kbit/s using RS232.

The I•CODE tags used belongs to a range of tags that are often referred to as *smart labels*. This is a concept used to describe the usage of the tags. Smart labels are small and cheap to produce which make it possible to use them in large scale labeling of objects. The physical shaping of the tags are done to make it easy to attach them to objects. The tags used in this project are almost paper thin and has a size of about 20x20 mm. Still, they are very durable to be able to manage tough environments.

A 50 ohm RFID antenna is connected to the reader. Due to the small size of the antenna (30x30 mm) the reading distance of the RFID reader is approximately only $5 - 10$ cm.

2.6 Other components

The RS232 protocol is used to handle the communication between the RFID reader and the micro controller. The micro controller cannot handle RS232 signals so a MAX3223 is used to convert between TTL and RS232 signals.

The batteries used should be as small and light as possible, but still have enough energy to be able to power the Tag+ for a long time. The batteries must be able to supply 3 V to the MICA2 platform and 4 to 6 V to the RFID reader. There is a problem with the RFID reader, and it is that the reader loads the batteries heavily during the time it's reading tags. The RFID reader requires batteries that can maintain their output voltage even if the load is high, and this often implies that the batteries must be quite big and unproportional to the rest of the design. Four alkaline AA batteries were chosen and the fact that these are rather big sized is ignored because the Tag+ is only supposed to be a prototype.

3 PROTOCOL DESIGN

As already mentioned in Section 1.3, the nodes should have a battery life of several years. Because of this, the main focus when designing the protocol was to make it very power efficient. The protocol should also be robust to massive deployments of nodes in limited space, so it needs to be very scalable.

3.1 MAC protocols - Related work

When designing a protocol for wireless applications, a problem that's not present with traditional wired communication arises. It's called the hidden terminal problem [9] and an explanation can be seen in Figure 3.1. Since node C cannot hear node A, then simply listening to the medium to try to detect a transmission doesn't work, since node A can be sending to node B. If node C then decides to transmit, it will interfere with the ongoing transmission. With the MACA [9] protocol, the authors suggest using an RTS-CTS[1] communication to reserve the medium. Thus, when A wants to transmit a message to B, it first tries to reserve the medium with an RTS call. Then all terminals which hear A will refrain to transmit during the time specified for A:s transmission. Assuming the medium is free, terminal B will answer with a CTS. That way, terminal B effectively reserves the medium for the remainder of the transmission. In MACAW [1] the basic CSMA-scheme is extended with a

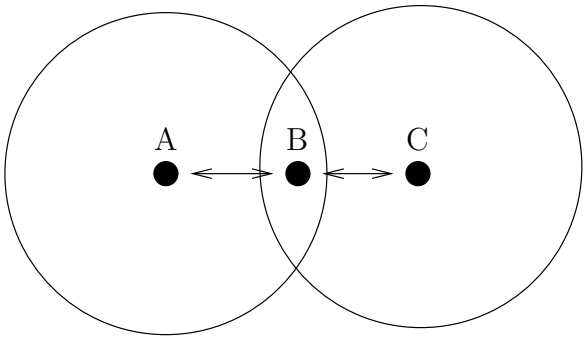

Figure 3.1. The hidden terminal-problem illustrated. Terminal A can hear terminal B but not terminal C. B can hear both A and C, and C can only hear B.

MAC-layer[2] ACK after the RTS-CTS-DATA transmission. By adding the ACK, link level message recovery becomes possible, which is much faster than application layer recover because "the timeout periods can be tailored to fit the short time scales of the media" [1].

[1]RTS: Request To Send, CTS: Clear To Sent
[2]Media Access Control Layer

A lot of research have gone into the subject of sensor networks in recent years. At Berkeley, the units are called smart dust because the size of the individual units are predicted to approach that of a grain of sand.

One common problem for sensor networks is their finite energy source. This makes it important to have a power efficient design all the way from the lowest levels of hardware to software and protocol design. One of the first pioneers in the area of power efficient MAC protocols is PAMAS [21], which tries to reduce the overhearing between nodes by using a separate signaling channel. This requires an extra radio, which increases the cost and complexity of the nodes.

Another protocol which uses two radios is STEM [19]. It uses a periodic sleep scheme, and aims to reduce the power consumption for the nodes in "monitoring state". To wake up the nodes, a wake up signal is sent on the signaling channel, and the length is set to be longer than the sleep+listen time. If the traffic on the network is very sparse, this protocol would be more efficient than a protocol that strives to synchronize the sleep periods.

Some work has also gone into using CDMA in sensor networks [31]. This also uses two radios, and CDMA is probably a bit to computationally intense to use in low cost micro controller units.

When it comes to TDMA, ECTS [30] which uses the GRAND-algorithm[3] is a very effective protocol. The authors identified that the time spent switching the radio between modes is a source of unnecessary energy consumption, and strives to minimize the mode switches. However, since it uses GRAND, the maximum number of nodes has to be known in advance.

S-MAC is a contention based MAC protocol not unlike IEEE 802.11 [10]. The protocol uses a number of mechanisms to reduce energy consumption and still maintain a tolerable latency and throughput. All nodes in the network synchronize their respective sleep/wake schedules, and all nodes that maintain the same schedule belong to the same "virtual cluster". So S-MAC is not really a cluster based protocol, which probably introduces some scalability issues when the number of nodes in the network grow fairly large.[4]

The main difference between S-MAC and IEEE 802.11 in power save (PS) mode is that the latter is "designed for single-hop networks" [10], whereas S-MAC includes synchronization between the nodes in the network.

To reduce the overhead generated from control messages, S-MAC introduces *message passing* [29]. When transmitting a message, the risk that it will be destroyed is proportional to the number of bits in the message. Hence, long messages will have a high risk of being received incorrectly and will have to be retransmitted. In IEEE 802.11 [10] the long messages will be fragmented into smaller ones, and each

[3] The GRAND (Galois Radio Network Design) algorithm is a very efficient topology transparent transmission scheduling method.

[4] A more detailed discussion of this is provided in Section 3.2.2

fragment requires an RTS-CTS-DATA-ACK communication. This way only smaller messages get retransmitted, which improves the general efficiency of the network. In message passing, the RTS-CTS in between fragments is skipped, instead the first RTS-CTS reserves the medium for the remainder of the transmission. This reduces the number of control messages and end-to-end latency for longer messages at the cost of fairness. But as the authors point out, "for sensor networks, application-level fairness is the goal as opposed to per-node fairness". To reduce the hidden-terminal problem, the ACK contains a field that specifies the duration of the transmission. Hence, the ACK works like a CTS and reserves the medium for another fragment on the receiver side.

In order to further reduce the general listening time for idle nodes, S-MAC uses the concept of a NAV[5]. Every time a node hears an RTS or CTS message, it updates its NAV with the value from the duration field. As time goes by, the NAV is decreased, and when it reaches zero, the network is supposedly free. This is called "virtual carrier sense". The medium will only be considered free if both the virtual and the physical carrier sense don't detect an ongoing transmission. S-MAC adds power saving features to the virtual carrier sense. When the NAV is nonzero, and the ongoing transmission doesn't involve the current node, the radio is powered off until the NAV reaches zero.

PCM (Power Control MAC) [8] is an extension to the standard IEEE 802.11 MAC protocol, but it can be used with any protocol applying CSMA/CA. The main purpose of the Power Control MAC protocol (PCM) is to use less power (compared to 802.11), and still retain the same throughput. This is done by transmitting DATA and ACK at the minimum possible power, but transmitting RST and CTS at maximum power. To reduce collisions, the data transmission periodically switches to maximum power, so that neighbors can sense the medium as occupied, and refrain from sending RTS.

3.2 CSMAC

The MAC-layer is responsible for allocating the medium so that multiple units doesn't interfere with each others transmission. Since this layer is responsible for passing most messages, it's also the appropriate place to make some power saving optimizations. The MAC-layer protocol will hereon be called CSMAC, which stands for Clustered Sensor Media Access Control protocol. The name is derived from the protocol S-MAC mentioned in Section 3.1, from which it inherits many power saving features.

[5]Network Allocation Vector [10]

3.2.1 Reducing collisions

There are four basic channel access methods by which MAC-layer protocols can be categorized:

- **FDMA** means Frequency Division Multiple Access. Sending and receiving units receive a unique frequency to transmit data. Only one transmitter/receiver pair may use a frequency channel at the same time. For our application, a lot of the communication will be in the form of many-to-one which means the benefits of FDMA disappears.
- **CDMA** or Code Division Multiple Access. There are actually a number of different subtypes to CDMA depending on how the spreading is done. But the common factor is that CDMA uses spread spectrum modulation techniques based on pseudo random codes, to spread the data over the entire spectrum [22]. While CDMA would be ideal in many ways for this protocol, it adds quite a lot of complexity and would be too computationally intense for the micro processor.
- **TDMA** stands for Time Division Multiple Access. Collisions are avoided by dividing the time into slots, and during each slot, only one unique node is allowed to send. Because of this, TDMA protocols suffer from scalability issues.
- **CSMA** or Carrier Sense Multiple Access is a scheme that tries to detect when the medium is busy. There are two basic subtypes, CA which means Collision Avoidance and CD which means Collision Detection. CSMA has proven to be very scalable by it's use in IEEE 802.11 [10].

For this protocol, a mix of TDMA and CSMA/CA was chosen. By combining the two, collisions on critical messages can be reduced by assigning these to specified time slots and scalability can be retained due to the fact that the rest of the communication uses CSMA/CA.

When a data message is to be sent, the medium is first controlled. If it appears to be free (at the sender), a first RTS message is sent. If the intended recipient receives the RTS, and the medium is free at the receiving end, a CTS message is transmitted. The medium will now be reserved for the duration of the transmission.

The RTS and CTS messages belong to a special group of messages, called control messages. All control messages are initiated by the MAC-layer, and they only carry information that's irrelevant to application layers.

Due to the stochastic properties of wireless communications, reserving the medium is not a guarantee for a proper reception of a transmitted message. To increase reliability, a `DATA-ACK` system is used to increase the reliability. In Algorithms 1 and 2, the data message communications are outlined in more detail.

Algorithm 1 Send msg

```
Fragment DATA
Send RTS.
if CTS then
  while Fragments to send do
    Tries ← 0
    repeat
      Send fragment
      Tries++
    until ACK or Tries=MAX_TRIES
  end while
  if Tries=MAX_TRIES then
    Signal error to upper layer.
  end if
else
  Medium is busy at receiver, sleep until next period.
end if
```

Algorithm 2 Receive msg

```
if Medium free then
  Send CTS
  while Data to receive do
    Wait for DATA for specific amount of time
    if DATA is correct then
      Send ACK
    end if
  end while
  Defragment data
  if Data received then
    Pass data to upper layer
  end if
end if
```

Another important aspect of the MAC-layer is choosing an appropriate back off if the channel is busy. This is handled by the physical layer in our implementation (the MICA2 CC1000 radio stack).

3.2.2 Power efficiency

There are four ways to reduce power consumption in a wireless network:

1. Reduce sending time
2. Reduce power while sending
3. Reduce listening time
4. Reduce switching time [30]

Let's examine each of those points a bit closer. Sending is usually the most power hungry action, since the used power can be in the order of $10 - 100$mW. So a reduction of the actual time a node is sending data will clearly reduce overall energy consumption. Some factors, such as maximum bit rate, are limited by the physical hardware. Others, such as channel coding and error correction belong in lower levels and is not in the scope of this work.

Reducing sending time

To reduce the time spent on sending data, the ratio of information bits per bit sent must be maximized. Of course, collisions come in here since if two messages collide, the actual information bits transmitted are nonexistent. By using the RTS-CTS system devised in Section 3.2.1, this problem is minimized.

In all packet based protocols, a control packet overhead comes in [29]. This includes pure control packets like RTS, CTS and ACK. By using message passing in CSMAC, the number of control messages (and hence the overhead generated by those) will be reduced. This is illustrated in Algorithm 2, where an `ACK` is sent after each received data fragment, and in Algorithm 1 where only one RTS is used to reserve the medium, regardless of the number of fragments that will be sent.

In CSMAC, a lot of work has also gone into reducing the length of the headers of messages. This came to the drastic conclusion of not using addresses for the units at all. Since the network should scale to an arbitrarily large size, the address space needed for fixed addresses needs to be very large. Also, because the network should be self configurable (see Section 1.3), the user shouldn't have to specify address space or maximum number of nodes prior to network startup.

Another reason to get rid of traditional addresses is that in sensor networks the nodes themselves aren't general purpose. There's usually no need to ask a specific node about some specific information, but instead most queries are for the information itself! So the idea here is to instead let the application layer define some kind of address based on what data the node contains!

A problem that application layer addressing creates is that these addresses aren't visible to the underlying layers. One problem where this is apparent is during an RTS-CTS handshake. If two nodes send out an RTS-message destined for the same host, and the host replies with a CTS, both hosts will start sending messages. A workaround for this is to use sequence numbers as message identifiers. In the implementation, an 8-bit sequence number is used. This in itself will not eliminate the possibility that two nodes will share the same sequence number. If we assume that the sequence numbers are generated at random from a uniform distribution of integers from 0 to 255, then the possibility of at least one collision is

$$P_1 = 1 - \frac{n!}{(n-m)!n^m}, \qquad \text{for} \quad m \leq n \tag{3.1}$$

where n is the number of nodes and m is the sequence number space. P_1 will quickly approach 1 when $m \to n$. But, since there's also a back off involved when a message is sent, the probability of multiple nodes sharing the same back off and the same sequence number is greatly reduced. Also, if two nodes where to transmit an RTS at the same time, with the same sequence number, their messages would probably interfere with each other so that nothing would be received.

Reduce sending effect

The approach used by PCM (described in Section 3.1) should integrate nicely with CSMAC. But because the PC simulator simulates the older MICA-platform, it didn't support changing the output power and readings of signal strength. So this feature couldn't be put to large scale testing and hence was not implemented.

Reduce listening time

Since the radio consumes approximately equal amount of energy while transmitting and receiving[6],we should strive to minimize the time the radio is listening at all. Ideally, the radio should sleep all the time it's not transmitting or receiving anything. But to be able to receive anything, the radio needs to be on when another station is transmitting. CSMAC tries to solve this problem by letting the nodes periodically sleep and wake up. A complete cycle of listen and sleep is called a *frame* [29]. Within a frame of time T_f, there is a listen time T_l and a sleep time T_s, and $T_f = T_l + T_s$.

During the listen time, the nodes listen for communication. This implies that all nodes that want to communicate with each other use the same sleep schedule, which is accomplished by a `SYNC` packet. Nodes that have the same schedule all belong to the same cluster.

[6]This implies that we use the methods described in [16]. The current is approximately 5mA for both modes.

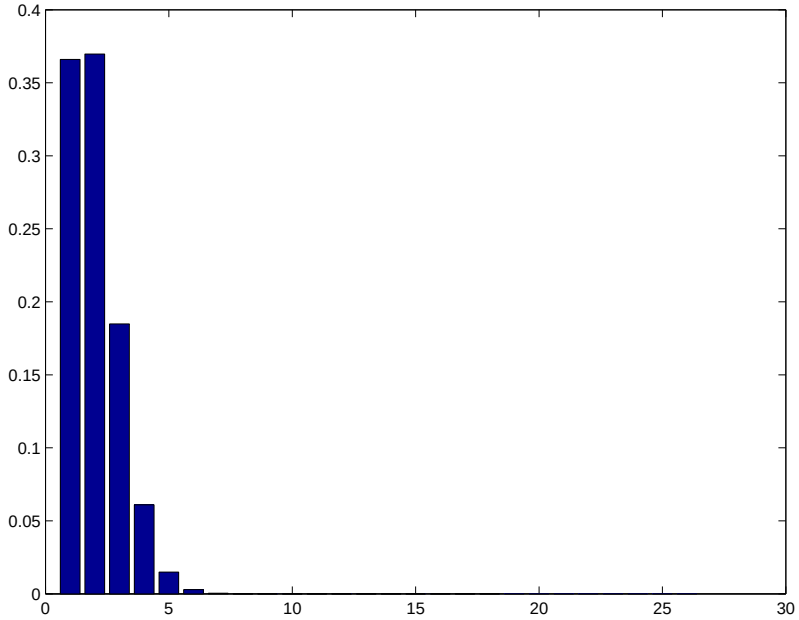

Figure 3.2. Here we see the probability distribution for $n = 100$, $p = 1/100$ and $x \in \mathbb{Z} \cap \{0, 25\}$. The probability of x nodes sending simultaneously decrease rapidly, and the probability of "wasted energy" by collisions is merely about 26.4%.

CSMAC changes the way S-MAC deals with clusters in the way that only the cluster masters are responsible for keeping the synchronization in the cluster, and are thus the only nodes that broadcast `SYNC`-packets. This has some benefits. First of all, the period in which nodes should listen to a `SYNC` can theoretically become smaller, because there's no contention about the medium for the `SYNC`[7]. This also automatically increases scalability, because there is no need to tailor the contention period based on the number of nodes that are in the vicinity of each other.

To illustrate this, suppose that the contention period wouldn't have to be decreased, instead the probability that one node among n nodes should send the sync should ideally be $1/n$. The expected value of the number of nodes to send `SYNC` will then be equal to one. The number of nodes x that would have to contend for the medium if we assume n nodes is expressed by the binomial distribution

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{3.2}$$

which has an expected value of np. So far, everything seems good. But, we cannot assume that the distribution of nodes is known in advance. Even the number of nodes to be deployed is unknown! So a reasonable assumption is that the number of nodes in the vicinity of one node can probably be in the range of $1 \leq n \leq 1000$. If the probability of a node sending a sync is chosen as $p = 1/500$, which is in the middle of the interval, then when it's two nodes no sync will get sent about 99.6%

[7] This is where the TDMA part comes in...

of the time. With the same p for 1000, contention would be needed 73% of the time. But since CSMAC uses real clusters, it won't have this scalability issue.

CSMAC also borrows the NAV-concept to let nodes sleep when the medium is allocated by other nodes. Because CSMAC uses message passing, RTS and CTS messages contain information about the whole length of the current transmission. This information is used to update the NAV, and when the NAV is nonzero, the radio is put in power save mode. [8]

Reducing switching time

Because CSMAC uses CSMA, this is one area where it's power saving features are sub-optimal (as compared to ECTS). The only way to reduce the number of switches in CSMA-based protocols is to try to piggy-back transmissions. In CSMAC, this would mean to piggyback broadcast messages to `SYNC`-packets, but this would only marginally impact the power savings. CSMA based protocols will inherit this disadvantage in comparison with TDMA based protocols.

3.2.3 Functional description

To start up a network, a special node, the sink is needed. That node is the information gatherer in the network. Since the nodes themselves are more or less dumb terminals with a single purpose, there's no intuitive way of extracting any information from them. Their purpose is just to send the information to the sink and let the controlling application deal with the information.

When a general node is powered on, it's unaware of it's surroundings. What it needs to find out is if there are other nodes and clusters nearby, and whether to join one of those clusters. What it does is that it listens passively for 3 complete frames (as defined in Section 3.2.2). If it hears one `SYNC` within that time, it will know that it is in range of one master, and will then become a member of that cluster and start synchronizing with that master (see Algorithm 3).

Cluster addresses - Potential

The case will be a bit more complex if a node hears several masters during the startup listen time. Which cluster will it join? One argument could be made that because of the power saving feature of PCM (mentioned in Section 3.1), a node should join the master that has the strongest received `SYNC`. But the network stability will increase if we join the master that is closest in number of hops to the sink.

[8] Because of timing issues in the PC-simulator, this couldn't be tested, so it's not in the current implementation. However, with accurate timing data, the implementation is trivial.

Algorithm 3 Startup

1: Wait $3T_f$ for `SYNC`
2: **if** received `SYNC` **then**
3: **call** startup::ReceivedSYNC
4: **else**
5: Pick random back off and broadcast `SYNC`-packet.
6: Potential ← `MAX_POTENTIAL`
7: Send SYNC
8: **end if**

By assigning cluster addresses based on the number of hops from the sink, this is rather easily accomplished. Because of the tendency to increase further away, and approach zero the closer we are to the sink, these addresses are called the clusters *potential*.

Because of this, when a node has several neighboring masters, it will know based on their potential which one it should join. If several of the masters have the same potential, it is of no importance to the network structure which one the node would choose to join, but there might be some benefits of choosing the one whose signal strength is greatest. The procedure is illustrated in Algorithm 4.

Algorithm 4 Startup::ReceivedSYNC

Add potential of this master to Master-List
Remember when this host is to SYNC next
while Still haven't received 2nd sync from all SYNC-hosts AND time passed from last SYNC is less than T_f **do**
  Listen for other hosts.
  if Received SYNC from another host **then**
    Add this potential to Master-List
    Set waiting time to T_f from now
  else if Received 2nd sync from known host **then**
    Adjust T_f for this cluster
  end if
end while
if more than one schedule **then**
  if more than one schedule with with same (lowest) potential **then**
    Adopt the schedule of the master where the receiving power was greatest.
  else
    Adopt the schedule of the host with lowest potential as primary
  end if
end if

In Figure 3.3 two problem scenarios are illustrated that has to be dealt with. Both unconnected nodes are outside the transmission range of their nearest master, and thus will not receive any `SYNC`. After their initial wait-period, they will become their own masters, with a predefined maximum potential. This means they are "floating

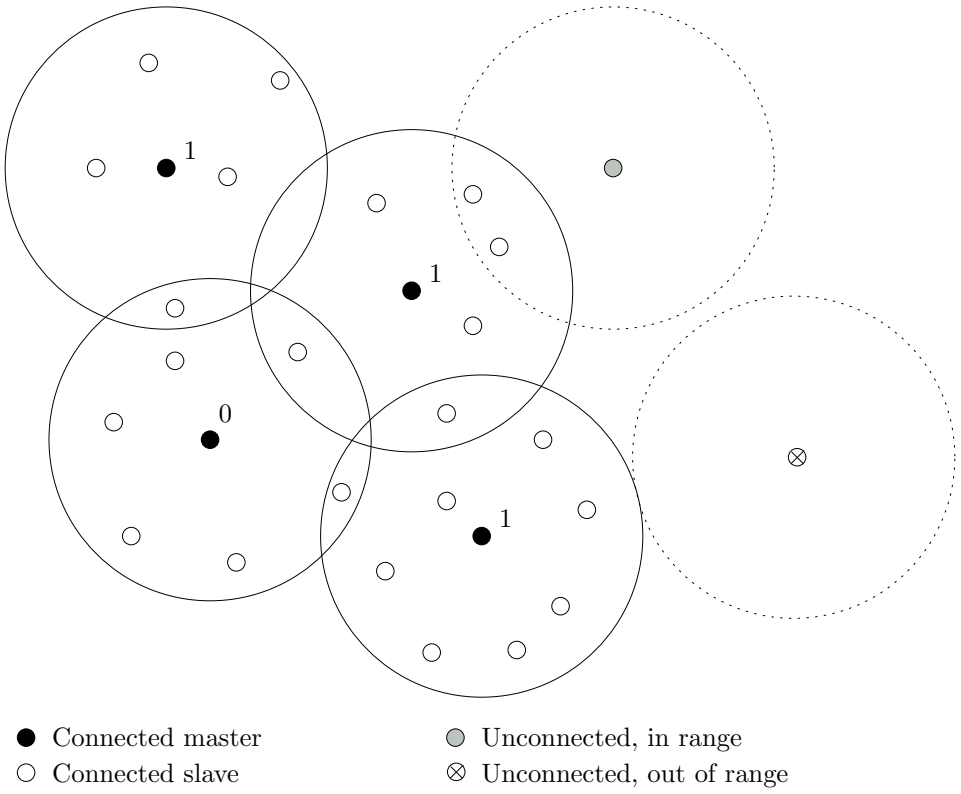

Figure 3.3. The large circles represent the approximate sending range of the masters. The numbers represent the clusters potential.

masters". A floating master never has any members in it's clusters, and several floating masters can be in range of each other. If a floating master were to hear a `SYNC` from a connected master, it will immediately join that cluster. By letting them periodically wake up for a whole period to listen for a `SYNC`, they won't be permanently unconnected provided a master will be in transmission range. And by assuming their own `SYNC` period, they will still operate energy efficiently even when unconnected.

The problem that remains are the nodes that are in range of the slave nodes, but out of range of the nearest master. By letting the slaves also periodically wake up outside of their "standard" `SYNC`-time, the slaves in range of the floating master will sooner or later hear the `SYNC` with max-potential. It will then proceed to send a control message with the purpose of changing the potential of the floating master. So after a while, the unconnected but in range master in figure 3.3 will become a master for a cluster with potential 2. This procedure is described in detail in Algorithm 5.

Algorithm 5 Periodic Neighbor Discovery

This "function" should be called from a "sleep function" if it's time for periodic neighbor discovery. Basically it just listens for SYNC-messages instead of sleeping.

```
while Time left do
  if SYNC then
    Update neighbor list
    Add this cluster
    Adjust SYNC-period for this cluster
    if SYNC-Potential < Potential then
      Switch primary cluster
      if SYNC-potential > Potential +1 then
        Tell SYNC-node to change potential to current potential +1.
      end if
    end if
  end if
end while
```

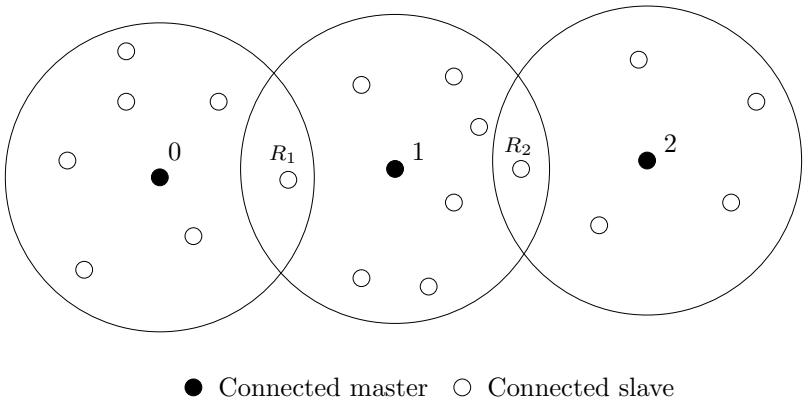

Figure 3.4. A simple network layout consisting of three clusters with potential 0, 1 and 2. The nodes called R_1 and R_2 are routing nodes connecting the clusters.

Error handling

When a cluster master goes down, that cluster will not be operational until a new master is assigned. But just assigning a new master might complicate things in some cluster layouts, because the new master might not have contact with all the nodes that the previous master could reach. Also some care needs to be taken so that nodes will not be assimilated into outer clusters, because those outer clusters might have gotten their potential from this cluster when it was operational. To illustrate with a simple example, look at Figure 3.4. If the master in cluster 1 would suddenly disappear, the outer cluster with potential 2 is no longer connected to the network. Also, it is impossible to find a new master that is in range of both R_1 and R_2.

The solution CSMAC uses is to let R_2 kill the outside cluster. At the next possible `SYNC` the master in the outside cluster will send a control message that informs the

cluster it's going to die a predefined number of periods from now. During this time, the nodes that aren't routing nodes are trying to find other clusters of a potential lower than 2 that are still connected. If a node finds a cluster with potential 1, it will quickly inform the master that a new connection has been made and the master will revert to normal operation. When the slave node hears the `SYNC` they will also revert to normal operation.

The nodes in cluster 2 that are routing nodes will just like R_2 try to kill their respective outside clusters. In this way, the whole branch that depended on the now disconnected master will prepare for a disconnection. When contact has been established again with a master of the same or lower potential, the routing nodes will inform their outside clusters that they can revert to normal operation again.

If no contact is established before the clusters die out, the procedure for rebuilding the network is exactly the same as when a node first wakes up.

3.3 Network layer

The network layer is responsible for routing data to the sink and queries out to the "branches" in the network. The need for energy efficient design is still present, and a major design goal was to reduce the overhead associated with the routing protocol. The following properties were identified as important both for functionality and energy efficiency:

- **Executes distributively**
- **Provides loop-free routes:** If there were loops in the routes, a lot of energy would be wasted because multiple copies of the same message would traverse the same path until it's discovered that this is indeed a loop.
- **Minimizes communication overhead:** This is done in two ways, first by localizing algorithmic reaction to topological changes when possible, to conserve available bandwidth and increase scalability. But also to minimize general communication overhead to find routes.
- **Establishes routes quickly:** Since the network is supposed to be semi static, the topology may change without notice. By establishing routes as quickly as possible, the probability that there will exist a route in the vicinity of a topological change increases.

3.3.1 Potentials and routing

As introduced in Section 3.2.3, the clusters themselves are already organized with cluster addresses that increase with hop distance from the sink. Using the potential

to find routes quickly turns out to be really good if measured by the properties above. If we always travel either up or down in potential, there will never be any looping routes. The routes themselves can be established as fast as the nodes get connected to the network. Communication overhead can also be reduced if the communication used to connect a new cluster (as the unconnected, in range node in Figure 3.3) is integrated with the communication to establish a route to this cluster.

The drawback in this scheme is that all communication is designed to have the sink as either it's main source or main destination. There is really no way to easily determine a route to a specific node in the network. But, since the nodes main purpose is to monitor some kind of possibly volatile data, the need for communication with a specific node is decreased. Instead, we believe that communication or data gathering will be based on what information the nodes have at the moment, or possibly in response to an event that happened somewhere in the network.

If addressing a specific node is important to the application, the proposed solution is to use application level channeling. This would work like a TCP-connection, except that the actual connection is initiated in the application layer. By letting the nodes that participate in the channel stay awake, messages can be sent at high speeds while the channel is open.

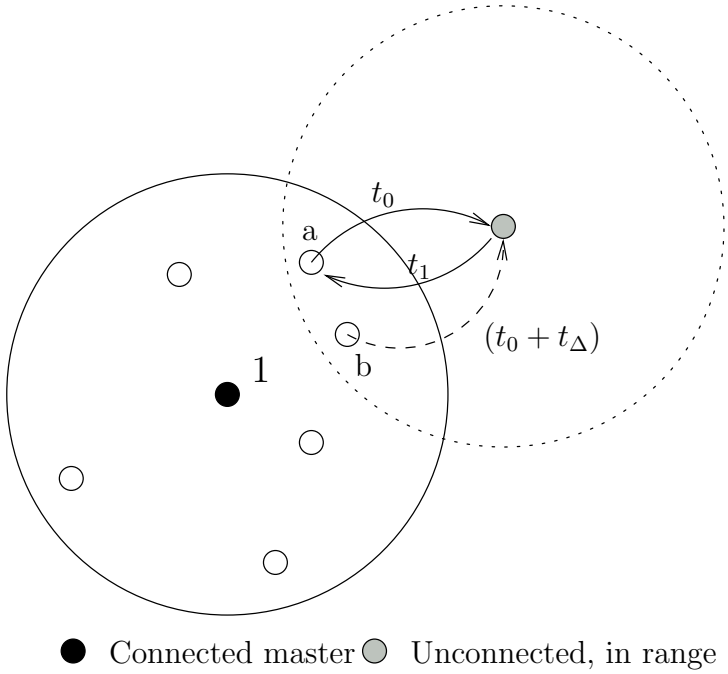

Figure 3.5. At time t_0, node a sends a message to the unconnected node to change potential (as detailed in Algorithm 5), and then receives an `ACK`. Node b also tries to change potential of the outer node, but doesn't get an `ACK`.

The process of router selection is outlined in Figure 3.5. To guarantee that only one node will be router between the two masters, the new outer master will work as a negotiator. The first node to receive an `ACK` to their change potential, will respond with a control message that confirms for the outer master that it has established

a route. In the figure, the node b tries to send a control message of type change potential a time t_Δ after the first transmission is initiated from node a. If t_Δ is less than the transmission time, the message will be scrambled and no control message will reach the outer node. If b missed the conversation between a and the unconnected node, and still sends a change potential message, then the outer master will not reply with an `ACK`. It will then give up further attempts to connect this node to the network and assume it has already been connected. However, if it where to hear a `SYNC` during the next discover period, it will once again try to connect that node (which may be a different unconnected master). See Algorithm 6 for more details.

Algorithm 6 Router Selection

```
if Received control message change potential then
  Lock potential change to current request
  if Potential > New Potential then
    Send ACK
    if Received routing message within time limit then
      Listen one T_f
      if No SYNC heard then
        Adopt new potential
        Become master and send SYNC
      end if
    end if
  end if
  Unlock potential change
end if
```

3.4 Data aggregation

The main purpose of sensor networks is usually to gather data in the nodes, that are to be processed in the sink. The idea, inspired by AIDA [7], is to let the leaf nodes gather data before sending it towards the sink. That way, fewer messages can be sent and the message passing functionality (Section 3.2.2) can be used at its full extent. A difference compared to AIDA is that we moved the data aggregation to the application layer, so that all data messages are passed to the application layer. This is done because application level compression can potentially be much more powerful than a general compression algorithm. Suppose for example that we are gathering data on the number of occurrences of some specific events. The application layer can be aware of this and thus knows that the order of the different events is unnecessary information and only the number is relevant. It can thus easily just count the events and assemble a new message with it's own event data added in a very space efficient manner.

The idea of the data aggregation and fragmentation is to use the clustered network structure to naturally aggregate data. The cluster masters are natural "local sinks"

to temporarily store, compress and fragment the data of the entire cluster[9] before forwarding it to the appropriate routing node.

The data aggregation should work as follows:

1. Base node sends a prepare data broadcast. All routing nodes pick it up and will prepare to forward it.
2. When the neighboring cluster master wakes up, the routing node forward the request.
3. The neighboring cluster masters sends broadcast and it's routing nodes forward as in item 2.
4. After neighbor clusters sync and broadcast the prepare data broadcast, data gathering can start.

[9]Depending on the size of the aggregated data, the master might have to send out smaller portions if the amount of data is larger than the actual free memory

4 IMPLEMENTATION

The nodes in a sensor network often have very high resource constrains. Still they have to be reactive and be able to participate in complex distributed algorithms. These facts makes it inappropriate to use ordinary operating systems. These operating systems includes functionality which is not needed in the nodes of a sensor network. Another characteristic for a operating system running on a node is that it should be adaptable to changes in hardware. This is because the hardware of the nodes evolve rapidly over time and it should be easy to port the application to new platforms. This also makes it easier to run the same application on different platforms.

The applications are often closely tied to the hardware and each node is likely to run only one application at a time. By making this assumption it would be possible to statically allocate all resources when compiling the application. Further it would be practical if one could use fixed software components which could be assembled to form the final application. This is because all applications have different needs and it would be more efficient to assemble only those components that are needed in the specific application. Below is a list in which a number of characteristics for sensor networks is pointed out [6]:

- **Driven by interaction with environment.** Nodes in sensor networks react to the environment or external signals rather than that they are used for computation like ordinary computers. This makes the nodes more event-driven, which puts demand on the operating system. Because data processing and events are concurrent there is a need for a concurrency model in order to manage possible race conditions.

- **Limited hardware resources.** The demands on low cost, small size and low power consumption imply that the hardware resources available are very limited. Improvement in new technology is more likely to result in smaller size of the nodes than better hardware resources. This means that applications that run on the nodes must be very efficient and small.

- **Reliability.** Sensor networks are used in environments where it is supposed to be operational for a very long time. This means that we do not want nodes to fail because of software related bugs. Bugs are often hard to predict, but there are methods in order to minimize the likeliness of them.

- **Real-time requirements.** The real-time requirements in the nodes have been proven not to be critical. The timing constraints are easily met by having total control over the application and limit the utilization of the resources. Radio communication could gain from better real-time capability but the nature of wireless communication of being unreliable spoils the gain.

4.1 nesC

nesC [5] is an open source programming language that is specialized for networked embedded systems like sensor networks. The implementation of nesC addresses the important characteristics mentioned in the list above. nesC is an extension of C, which is a language that is supported by many micro controllers and includes the necessary features to interface with hardware. The extension is needed because C has limited support for writing structured application and it is difficult to write safe source code. nesC defines a component based model in order to make it possible to split applications into separate parts which communicates with each other using bidirectional interfaces.

nesC does not permit separate compilation as C does. This is because nesC uses whole program analysis to improve the performance and make the source code more safe. Because the size of the application often is relatively small the need for separate compilation is not very critical. nesC is a *static language* meaning that the memory allocation for the application is fixed after the compilation. This has the disadvantage that it's not possible to use dynamic memory allocation and function pointers. The advantages are that it is possible to further improve the source code safety at compile time to detect possible data races and to make it easier to optimize the source code for better performance. nesC also has a simple concurrency model and with the compile time analysis most data races resulting from concurrency can be detected.

4.2 TinyOS

TinyOS is an event driven operating system designed for sensor networks, where demands on concurrency and low power consumption are high but the hardware resources are limited. TinyOS is written in nesC and much of the design of nesC was actually done in a way to increase the performance and utilization of TinyOS. TinyOS provides a number of system components that can be reused in many applications. The components are wired together to the final application by using implementation independent wiring specification. The event-based concurrency model TinyOS uses has a close relation to the concurrency model that nesC uses. TinyOS uses two types of concurrency, *tasks* and *events*. Tasks are run to completion and cannot preempt each other. They are to be used for computation processes where timing requirements are not strict. The tasks can be posted by the components and are run when the scheduler says. Events also run to completion but can preempt other events and tasks. They can be used to handle time critical operations and hardware interrupts. The simple concurrency model that TinyOS uses offers relative high concurrency but with low overhead in contrast to threaded concurrency which requires a lot of overhead. The data races that can occur when using concurrency are detected by the compile time analysis that nesC compiler offers.

The core of TinyOS is only about 400 bytes in total which makes it possible to use it on almost any modern micro controller today. TinyOS is used on the MICA2 platform and many of the software components needed to use the platform are already written and are available in open source code [27]. This means that the actual interfacing with the hardware is very easy.

4.2.1 TOSSIM

To verify that the implemented applications work in the way they should, it is convenient to be able to simulate the applications on a computer. A good simulator can be even better than doing tests in reality with ordinary hardware nodes, because it is easier to overlook and analyze the network in a simulator. It would also be almost impossible to simulate hundreds of nodes in reality because of the space needed and also because of the high cost. A simulator offers a reproducible environment and the benefits of being able to simulate the system are many. For example different system designs can be evaluated and possible problems that are not obvious during the design can be detected.

TOSSIM is a simulator for TinyOS wireless sensor networks and has been designed to capture the following properties [12]:

- **Completeness.** The simulation covers as many system behaviors as possible.
- **Fidelity.** The simulator is able to capture the behavior of the nodes *in detail*.
- **Scalability.** It has the capability of simulate a large number of nodes simultaneously, else it would be impossible to simulate an entire network.
- **Bridging.** Errors often depend on an incorrect implementation of a proper algorithm. The simulator uses the same code that is used to program the hardware, which means that the errors in the implementation will be detected.

TOSSIM manages to simulate thousands of nodes from the same code that is used to program the hardware. Only a few low-level hardware components are replaced which guarantees the completeness, fidelity and bridging. The low-level components are replaced by simulated events sent by TOSSIM. The network are simulated at bit-level which gives a great granularity of the simulator.

TOSSIM uses a simple but effective way to model the network. It uses a directed graph where the vertexes corresponds to the nodes in the network, and the edges between two nodes hold the bit error probabilities for the communication between them. The model supports asymmetric links which is common in wireless communication. Two vertexes a, b in a graph can have two different edges between them (a, b) and (b, a) which means that different bit error probabilities can be given each direction. Every node has a list which holds the nodes it is supposed to hear. With

this simple model it is possible to simulate many of the situations that can come up in a real network. Different radio models can be assigned by using different bit probabilities on the edges. The bit error probabilities can be chosen by the user and can also be changed during run-time which make it possible get a realistic simulation with nodes joining and leaving the network.

However, TOSSIM has its limitations. It neither simulate power consumption or radio propagation delays, and interrupts are simulated as non-preemptive which is not the case in TinyOS. TOSSIM does not fully supports the MICA2 platform because the low-level simulation of the radio stack uses a model of an old RF transceiver used in a former model of the MICA2 platform.

TinyViz

TinyViz is a visualization tool for TOSSIM which allows better control and analyze capabilities for the simulation of the network. The information generated by TOSSIM is used as input for TinyViz which then visualize the information on the screen. TinyViz can also send information to TOSSIM which make it possible for the user to interact with the simulation. For example it is possible to switch off a node in TinyViz, which then tells TOSSIM to switch it off. By loading different plugins TinyViz can be configured to different needs. A radio model plugin allows the bit error probabilities to be dependent of the distances between the nodes in the graphical user interface and by moving the nodes around one can test different network configurations.

4.3 Protocol implementation

A subsection of the protocol as defined in Chapter 3 was implemented on the MICA2. Using the existing physical layer as of TinyOS version 1.0, some adaptation had to be made to use the CSMAC packet structure. Since the actual implementation is more of a prototype, a decision was made to encapsulate the CSMAC packet into the existing AM-packet used by TinyOS. It is still possible to verify the functionality of almost every aspect of the protocol, with the exception of actual power usage and transmission times which would be skewed upwards because of the extra and non-optimized headers.

4.3.1 MAC component

The MAC component is responsible for passing messages to the radio and decide what layer (if any) that should receive messages that are received from the radio. But it's also where most of the power management features that are described in Chapter 3.2 are implemented, some of them by using help components such as

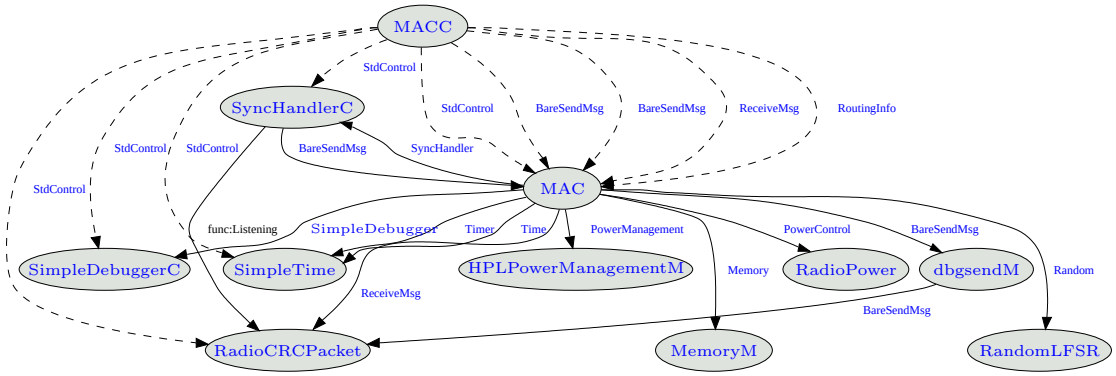

Figure 4.1. The component graph for the MAC-component. The dashed lines describe the "wirings" (as explained in Section 4.1). The solid lines are the interfaces that connect the components.

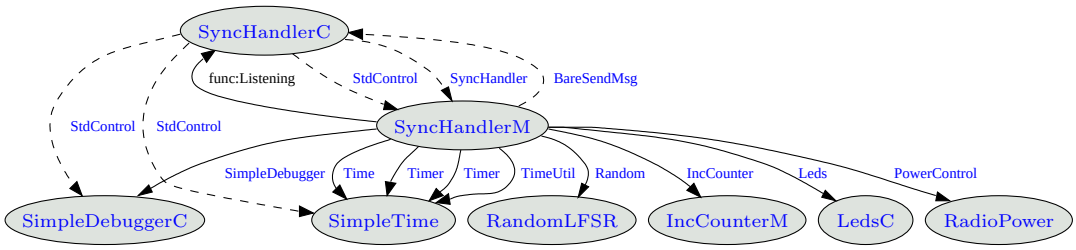

Figure 4.2. The component graph for the SyncHandlerC-component.

SyncHandlerC. In Figure 4.1 we see the component graph, or the call graph for the MAC-component. At the bottom, the radio is interfaced by RadioCRCPacket, which in turn is a configuration component that consists of wiring to the actual radio implementation (see Appendix D).

SYNC component

The SYNC component (see Figure 4.2) is in a way the core of the power management features in CSMAC, the periodical and synchronized sleeping of the radio. It is actually just a helper module for the MAC component, and interfaces with that component by the defined interfaces BareSendMsg and SyncHandler as can be seen in Figure 4.1. BareSendMsg handles a higher prioritized sending of SYNC-messages, which needs to be sent on a more precise time. The SyncHandler interface provides some commands for the MAC component to control it's operation. All intelligence is hence committed in the MAC module, and information such as new masters or potential changes are communicated by commands to the SYNC component. That component in turn signals back relevant information to the SYNC layer, such as if a SYNC has or has not been received by the appropriate master, and when potential changes in SYNCS actually happen. But it also interfaces with the RadioPower component and will thus turn off the radio when a sleep period is entered.

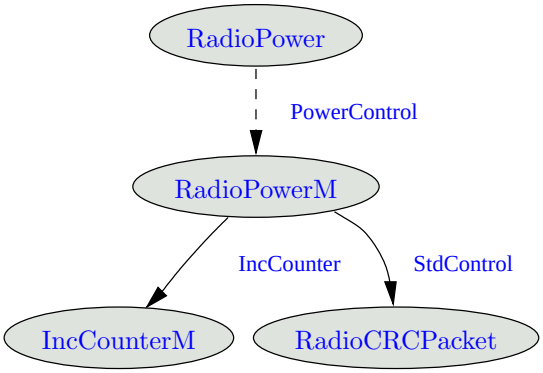

Figure 4.3. The call graph for the RadioPower-component.

Memory module

TinyOS doesn't have a built in `malloc`-function[1]. Instead, it works by reserving buffers in each module and passing pointers between layers. So when for example a routing message should be forwarded, its content should be copied to a routing buffer in the NetworkM-module. That pointer is then copied down to the MAC-layer and downwards to the physical layer. The inherent limitation to this is that it's impossible to queue messages unless it's known in advance how many messages should be generated, and from what layer. Because of this, the memory component was implemented. It's a simple memory allocation component that is designed to store messages. Therefore each allocated segment is the same size as the internal message type.

RadioPower component

To modularize the interface for power management (see Figure 4.3), a components was introduced which purpose is to centralize the actual power off for the radio. This makes it possible to use several different schemes for energy efficiency in all layers, without them communicating their status to each other. A help module, called IncCounter stores a counter value for the radio power component. Every component that wants to turn the radio on signals RadioPower, which increases the counter. When the counter is non-zero, RadioPowerM makes sure that the radio is on via the RadioCRCPacket component which is the highest level interface to the physical hardware. Likewise, when the counter reaches zero, the power is turned off.

[1] `malloc` is the C command for allocating a buffer of memory.

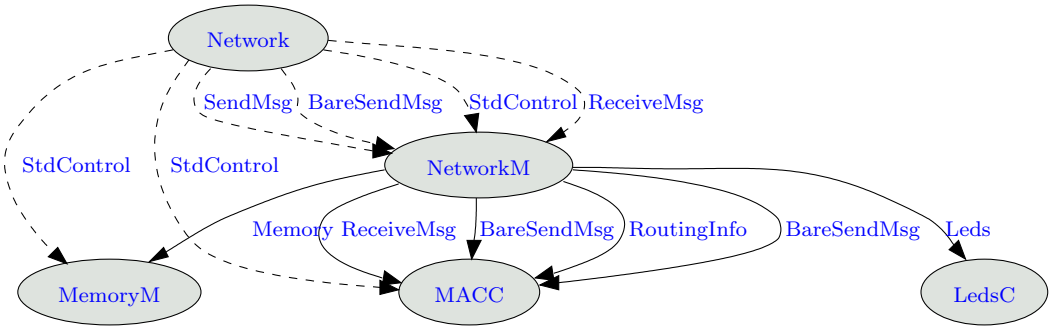

Figure 4.4. Network (routing) component

TinyOS built-in components

As can be seen in Appendix D, the complete application consists of a lot of components, of which some are built in to TinyOS. Some that should be mentioned and exist in Figures 4.1, 4.2, 4.3 and 4.4 are:

- **SimpleTime** This is a very important component that provides timer interfaces and handles general timing events. The provided timer functions can be set to execute periodically or only once. For example, the SyncHandlerC sends SYNC by setting a periodic timer to execute with period T_f. It also provides an absolute time interface that can be set to execute x ms from now.

- **RandomLFSR** This is the pseudo random number generator that's needed for among other things selecting random back offs and sequence numbers.

- **LedsC** The component that handles the three LEDs on the MICA2 board.

- **PowerManagement** This component can set the radio in idle mode, instead of turning it off completely.

4.3.2 Network component

The component call graph can be seen in Figure 4.4. This component handles the network layer, and receives it's messages from the MACC-layer. Messages that need to be routed get encapsulated within a routing packet, and received routing messages get either forwarded, discarded or "unpacked" and passed up to the application layer. The component also interfaces to the same MemoryM as the MAC-layer, and thus messages with for example routing information can be generated within the component. Also, message types that should be forwarded and received simply get copied instead of having additional latencies associated with waiting for pointers from upper layers before forwarding the message.

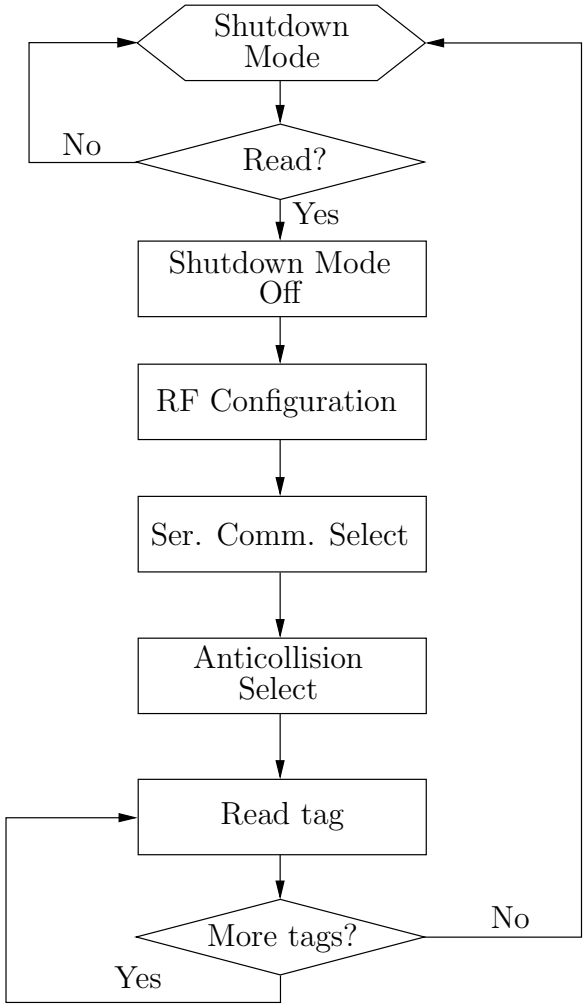

Figure 4.5. RFID readers state diagram

4.4 RFID reader

The RFID reader also has to be integrated in the system and shall also be controlled by the micro controller. The communication is done using RS232 serial communication and a predefined protocol specified in [23]. TinyOS has very good support for communication with such devices because everything below byte-level is already taken care of. The reader is put in shutdown mode between each read to preserve energy, and therefore it has to be reconfigured each time it is started before it can read tags. Every command sent to the RFID reader is followed by an acknowledgment or the return of data from the reader. The RFID reader is programmed to follow the state diagram in Figure 4.5.

As long as the reader is not triggered to read it stays in shutdown mode. When it is instructed to read tags the reader leaves the shutdown mode and is reseted. Then a number of instructions are sent to the reader. First the RF is configured to be

activated by turning off the reader supply mode, which reduces the consumption of the reader at a cost of a higher latency. Then the serial communication type is set to RS232 and the reader is instructed to leave standalone mode, which means that the reader will wait for an instruction to read tags instead of reading tags continuously.

The anti collision select instruction select all chips in range and assigns each found tag a time-slot in which it is possible to communicate with that specific tag. All tags are read, family and application numbers are neglected. Maximum number of time-slots are set to 4 in this system, which means that the maximum number of tags that can be read simultaneously is 4. Finally the reader is instructed to communicate with the tags that are within range and asks them for their tag IDs which is stored in the first two blocks of the tags memory. Only one time-slot can be read at a time and this is repeated until every time-slot have been read. Finally the reader is instructed to put itself in shutdown mode again and wait for a new read instruction.

4.5 Graphical User interface

The central node is connected to a PC as described in Chapter 2. A simple GUI (Graphical User Interface) has been created in order to show how the information from the tags can be collected and presented for the user. The user can also trigger the collection of data (tag IDs) by sending a command to the central node which then forwards the request through the network in the way the protocol describes. All nodes responds to the request by first read the tags in range of the reader and then forwards the IDs of these towards the central node, which forwards it to the PC.

The GUI is a Java application and a screen-shot can be seen in Figure 4.6. The screen is divided into three different parts. In the upper-left corner is the *Tag Input*, where information about incoming data is reported. In the lower-left corner the *Product Info* shows information about the selected product. Each product is assumed to has an RFID tag attached to it so when the computer receives a tag ID it can look up which product it corresponds to. The available information about the product is product name, product ID (the unique tag ID associated with the product), time when the product information was last updated, time when the product information was first updated and the number of times the information about the product has been reported (i.e how many times the RFID tag has been read).

For each packet received it is possible to get its corresponding RSSI value. This is as the name indicates a measurement of the strength of the received signal. By knowing the transmission strength it is possible to calculate how much the signal has been attenuated on its way to the receiver. The attenuation is depending on the distance between transmitter and receiver which means that it is possible to calculate the distance between the two if one knows how the dependency look like. A very simplified dependency is to think of the the signal propagating in free space

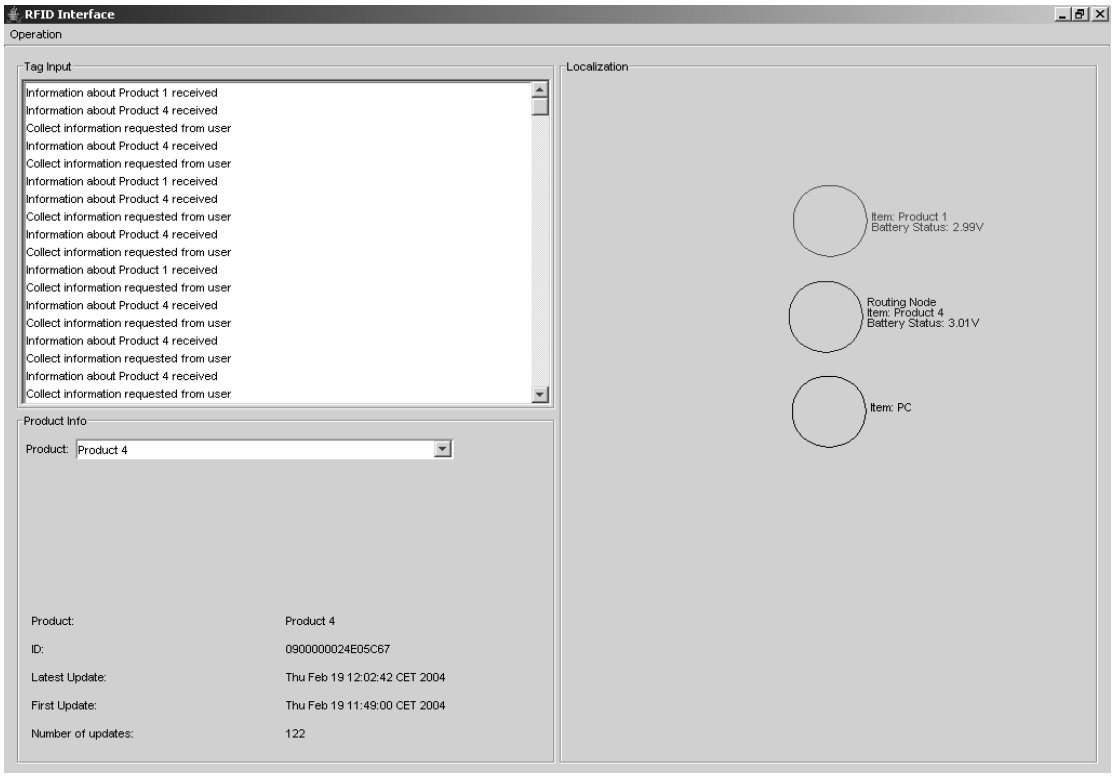

Figure 4.6. Graphical User Interface

where the attenuation in proportional to $1/r^2$, where r is the distance between the nodes. In reality this dependency is far more complex and is very much affected by the surrounding environment.

To the right in the GUI there is a very simple form of localization display of the nodes, *Localization*. By using the RSSI values on the packets received[2] the distances between nodes can be calculated with the simplified dependency assumed. Without further improving the method it is easy to understand that it is not possible to determine the direction of the nodes, but just the distance to it. Each circle corresponds to a node in the network, and the node in the middle is that connected to the PC. Whenever an information packet is received at the PC the source node from where the packet was originally sent is drawn on the screen. As one can see if a node act as a router it is illustrated in the user interface by the text "Routing Node" next to the node. Also the battery status of each node is displayed next to each node together with the product name if a tag was found and read by that reader.

[2]If packets are routed the RSSI value that the routing node got when it received the packet, must also be forwarded to the central node as side information to the forwarded data packet.

5 VERIFICATION AND EVALUATION

5.1 Radio communication regulations in Sweden

The radio communication in Sweden must follow fixed regulations. Depending on which carrier frequency that is used, different regulations apply. In our case, where we have a carrier frequency of $f_c = 434.026$ MHz, the communication must follow the regulations that apply to the frequency band $433.050 - 434.790$ MHz. This is a license exempt band where, if you fulfill all the conditions, do not need a license to use the system. A detailed list of conditions that have to be fulfilled are given in [15]. The condition to be met for our system in the frequency band $433.050 - 434.790$ MHz is to keep the transmitted power below 25 mW (13.98 dBm) EIRP (Effective Isotropic Radiated Power). Another constraint is often that the system has to use a duty cycle that is lower than some fixed value, but this is not present in the swedish regulations for the given frequency. Another important aspect is that of testing that the system really meets the conditions. In order to let a product out on the market there are regulation about product information, verification and marking of the product. This system that is not to be released on the open market does not have to fulfill these conditions and still is allowed to be demonstrated on for example trade fairs exhibitions. More detailed information about this can be found in [14].

The EIRP is calculated as

$$P_{EIRP} = P_{out} + G \tag{5.1}$$

This means that in order to calculate the EIRP value one needs to know the antenna gain, G. The CC1000 chip has a maximum transmission power, P_{out}, of 10 mW (10 dBm). 25 mW equals 13.98 dBm, so if the antenna gain is below 3.98 dBi it would be allowed to transmit with maximum power with the CC1000. The loop wire antenna that is used with the motes has not been calibrated to provide an exact gain value. However since it is a quarter wave, monopole without any ground plane, the gain will not be higher than 1 (0 dBi). As long as a monopole or a dipole antenna is used, one should be able to cope with the maximum regulated power.

5.2 System performance

5.2.1 Power consumption

The expected battery life time of the Tag+ shall be a year or more, which puts hard constrains on the power consumption of the Tag+ units. To be able to meet this the Tag+ has to go into sleep mode when it does not transmit, receive or communicates with the RFID reader. The ratio between the amount of time it is in sleep mode

and the amount of time it is in operational mode is given by the duty-cycle. A lower duty-cycle results in shorter time in operational mode which decreases the energy consumption, but it also increases the latency in the network. A theoretical calculation how long the expected battery life time is made below. It should be pointed out that these calculation are only in theory and the power management was not implemented on the hardware. However the protocol is designed in order to be able to use it.

First we make some assumptions regarding the power consumptions of the different devices. The energy used by the MAX3223 circuit (see Section 2.6) is neglected because in a real product it is not likely to use a converter at all. The RFID reader has a current consumption of 500 μA when it is put in shutdown mode. The AA batteries that is used in this project have a capacity of about 2000 mAh. With the consumption of the RFID reader in shutdown mode the batteries will last shorter than half a year. However it would be possible to cut off the power source to the RFID reader completely and thereby eliminate the power consumption when the RFID reader is not used. From now on we assume that the RFID reader does not have any power consumption when it is not used.

To be able to estimate the energy consumption of the RFID reader when it is operational we must make some more assumptions. First we assume that when the reader is instructed to perform a read there will be one tag within range. From this it is able to estimate that the amount of data that is sent between the reader and the micro controller is about 60 bytes per read instruction. This figure is important because during this transmission the RF field of the RFID reader will be on which means that the reader has a very high current consumption. The different current consumptions of the RFID reader in different operating modes can be found in [24].

The transmission rate is set to 38.4 kbit/s with each transmitted byte is followed by a stop bit which implies that each byte corresponds to 9 bits. The time it takes to send the information, T_s, is therefore given by

$$T_s = \frac{60(8+1) \text{ bits}}{38.4 \text{ kbit/s}} = 14.0625 \text{ ms} \tag{5.2}$$

Further the time it takes for the reader to respond to a command given is approximately 1 ms [23]. In total 5 commands are sent to the reader when one tag is within range resulting in the total respond time, T_r, equals 5 ms. Finally the time it takes to wake up the reader from shutdown mode, T_w, is 10 ms, and during this time the RF field is not on resulting in a lower current consumption. The current consumption of the reader while the RF field is on, $I_{RF_{\text{on}}}$, equals 115 mA and the consumption when the RF field is off, $I_{RF_{\text{off}}}$, is 18 mA. The energy required to go through the read sequence, E_{readtag}, is given by

$$E_{\text{readtag}} = UT_w I_{RF_{\text{off}}} + (T_s + T_r) I_{RF_{\text{on}}} = 7.1166 \text{ mJ} \tag{5.3}$$

where U is the supply voltage, 6 V.

The life time of the batteries is dependent on how often the RFID reader is trying to read a tag. If we assume that the reader is instructed to read a tag every minute it is possible to calculate the average current that the reader consume, I_{RFID}

$$I_{RFID} = \frac{E_{\text{readtag}}}{60U} = 0.03954 \text{ mA} \tag{5.4}$$

Current consumptions of the MICA2 platform in full operation and in sleep mode are given in [25] together with average current consumption with a duty-cycle of 1 percent. A 1 percent duty-cycle would be realistic to use with the protocol designed in this project, however the latency would be quite long. The average consumption of the micro controller, $I_{\mu C}$ for the given duty-cycle is 0.0879 mA and for the RF transceiver, I_{RF}=0.0920 mA.

The battery life time using the given duty-cycle and making the assumption of reading one tag per minute can be calculated. Of course the life time is depending on which types of batteries that is used. Different batteries are used for different applications and the capacity of them differs a lot. The battery used in this project, high quality AA batteries, have a capacity, Q_{batt}, of about 2000 mAh. The life time of the battery, T_b can be calculated as

$$T_b = \frac{Q_{\text{batt}}}{I_{\mu C} + I_{RF} + I_{RFID}} = 9114.26 \text{ hours} = 1.04 \text{ years} \tag{5.5}$$

5.2.2 Simulator

Several simulations where conducted using TinyViz (described in Section 4.2.1). There are some problems with the TOSSIM and TinyViz environments, as of TinyOS version 1.1. There are some bugs with the simulations of the timers as used in the ATMega micro controllers, which makes it virtually impossible to optimize timing constraints for the actual hardware. Also, as mentioned in Section 4.2.1 the actual hardware simulated is that of the MICA platform and not the MICA2 platform that the actual implementation was made of. Also, because of the new radio on MICA2, sending times and mode switching times are not correct neither.

What TinyViz was used for was instead to test the protocol on more complex network settings, many nodes and up to 10-hops in the longer paths. By TinyViz, it was also easy to simulate masters running out of power and testing the rebuild process of the network. The implemented features worked out well in this controlled environment, and simulations where run for long periods of times with maintained functionality in the network.

5.2.3 Real World Test

In the real world, another limitation presented itself in the form of a small limit of actual physical nodes. A network with only three nodes isn't really that complex, and can only form two kinds of networks. A one cluster network where both slave nodes belong to the master or a one hop network (with two clusters) where one node will be a routing node for the other "non-sink".

What was interesting with these real world simulations was to note how stable the actual RF links were in a more complex RF environment. Early simulations showed high information losses when the two-cluster model where used, the transmission power set to the lowest possible and the nodes where placed at almost maximum distance from each other. The strength of the received signal quickly diminish when the distance between nodes grow (the received signal is approximately proportional to $1/r^2$, where r is the distance between the nodes) and because of this, many critical messages (such as `SYNC`-messages), where lost due to the high packet losses. To reduce this problem, an RSSI threshold was imposed for a node to actually join a cluster. The cost of this is that more nodes will become masters, but if the duty cycle is long enough, the cost of missing a `SYNC` far out weights for a few more `SYNC`-messages.

6 CONCLUSIONS

The task of this project was to integrate an RFID system into a sensor network to be able gather data from RFID tags spread over a large area. Most of the work has been done in designing and implementing a protocol for this network. This paper has presented a complete protocol (with the exception of physical layer) for a sensor network with small, battery powered nodes. Everything in the protocol, from MAC-layer up to the data aggregating application layer has been designed with power efficiency in mind.

The CSMAC protocol inherits many features from the S-MAC protocol such as message passing, overhearing, periodic listen and sleep. It extends S-MAC with a cluster based design which should improve scalability, and also optimizations for reducing the sending effect without reducing throughput that's borrowed from PCM. Headers and control message sizes has been optimized, and in that process the addresses where removed to save additional space and energy. Since the use of addresses is limited in "anonymous" sensor networks, the usability cost is very small.

Without addresses, a different kind of routing protocol was needed because many of the traditional routing protocols assume that every node is uniquely identifiable. Also, because of the inherent characteristics of data aggregating networks, much of the communication will be to or from the sink. With these properties in mind, a simple routing protocol based on the cluster structure of CSMAC was presented. Because of it's simplicity, the overhead of routing packages is virtually nonexistent.

Finally, a simple data aggregation protocol that resides close to the application layer was proposed, that aggregates the data cluster by cluster and thus uses the localization features of the network to it's advantage.

The experience from using an RFID system is that it offers a very powerful tool for tracking items. Since RFID doesn't requires line-of-sight it has an important advantage compared to many existing tracking systems, like bar code and infra red readers. These systems require that the reader is directed towards the object in order to function. RFID is still very costly to implement in big scale and it will probably take a very long time before it is possible to outperform bar code systems in a economical point of view. Most RFID systems today can be found in applications where the tracked objects have high value which make the use of RFID systems feasible despite the high cost.

6.1 Future Work

In this section, some ideas about possible future work is given.

6.1.1 Localization

Nodes that possess some kind of information and act without human interaction introduces a new kind of problem, namely localization. In wired networks, a terminals address is associated with physical wires which makes the location known in advance. With sensor networks, this assumption no longer holds. Work has gone into developing methods for using localization with distributed wireless networks, see for example [2] and [18]. By using more nodes, more information sources are gathered. There are right now some ongoing work on implementing localization on TinyOS and the MICA2 platforms, such as described in [4] and Calamari [28]. Calamari uses, apart from radio signal strength, acoustic time of flight and ultrasound time of flight to make the positioning more exact.

6.1.2 Data safe aggregation

If the data contained in the network is very critical, a two level `ACK` for routing messages could be implemented to further increase the probability of correctly received messages. When a master is trying to send data to the sink, the packet gets encapsulated in a routing packet destined for the cluster with lower potential than the current cluster. In the current design, when a master has sent it's message and it's received by the routing node, it gets the `ACK` and thus drops the message. Instead, it could wait for a second `ACK` to arrive, which originated from the next-hop node (the master in the cluster with lower potential). The routing node would thus forward the `ACK` it gets from it's master. This would make routing messages a bit more stable to topology changes, for example if the routing node would cease to exist when it was about to forward a message.

6.1.3 Controlling outer clusters `SYNC`-timing

A network that uses CSMAC will work well for two types of data aggregation (or a combination thereof):

1. **User requested data aggregation.** All data aggregation requests originate from the sink, which might be a PC. The request is initiated by the sink, and all nodes that has some relevant information respond to the request.

2. **Event based data forwarding.** In contrast to item 1, let the leaves themselves instantiate the communication as a response to some sensor event, and

forward the data to the sink.

If the system is not a combination of the two, but solely uses one of the systems above, the overall latency can be greatly reduced by letting the inner clusters control when the outer clusters should synchronize. For a user requested data aggregation system, the outer clusters should synchronize a short time T_Δ after the inner cluster, and thus the aggregation request can be forwarded without a long delay to the outer clusters ($T_f/2 + nT_\Delta$ if n is the cluster number for the outermost cluster). If the system is based on events, the outer systems should instead synchronize a short time *before* the inner clusters, to be able to forward data after a shorter delay.

6.1.4 Updating data

Because of the localized structure of the data aggregation, the same nodes will usually inform the same masters of their data. If the data itself is non-volatile, or at least semi-static, much of the data that's sent will simply be a copy of the last transmission, or maybe just with slight modifications. By using a remote update algorithm such as rsync [26], the actual data transfered could be greatly reduced.

6.1.5 Further energy saving optimizations

Instead of transmitting at the maximum possible effect during data transmissions, the accumulated output energy can be reduced by using the scheme proposed in PCM. Every `ACK`, RTS and CTS uses the maximum available power, but the data messages themselves are transmitted with a lower effect. The actual effect used is calculated by the received signal strength, which is communicated back to the sender in the CTS message. In this way, nodes that are in the vicinity of each other use a lot less energy to transmit the same amount of data. To reduce the amount of collisions, the transmitter periodically switches to maximum effect so that nodes that could have missed the RTS-CTS communication still gets a chance to sense the ongoing transmission.

Bibliography

[1] Vaduvur Bharghavan, Alan J. Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless LAN's. In *SIGCOMM*, pages 212–225, 1994.

[2] N. Bulusu, J. Heidemann, and D. Estrin. GPSless low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, Oct 2000.

[3] Klaus Finkenzeller. *RFID Handbook*. Wiley, 2nd edition, Apr 2003.

[4] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman, and Sundeep Pattern. Distributed online localization in sensor networks using a moving target. Technical report, Information Sciences Institute, Department of Electrical Engineering-Systems, University of Southern California, 2004.

[5] David Gay, Philip Levis, David Culler, and Eric Brewer. necC 1.1 language reference manual. http://nescc.sourceforge.net/papers/nesc-ref.pdf, May 2003.

[6] David Gay, Matt Welsh, Philip Levis, Eric Brewer, Robert von Behren, and David Culler. The nesC language: A holistic approach to networked embedded systems. *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, 2003.

[7] Tian He. Aida: Adaptive application independent data aggregation in wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 193–204, Los Angeles, California, USA, 2003. ACM Press.

[8] E. Jung and N. Vaidya. A power control mac protocol for ad-hoc networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking table of contents*, pages 36–47, Atlanta, Georgia, USA, 2002. ACM Press.

[9] P. Karn. Maca - a new channel access method for packet radio. In *Amateur Radio Ninth Computer Networking Conf.*, pages 134–140. ARRL/CRRL, 1990.

[10] LAN MAN Standards commmitee of the IEEE Computer Society, IEEE, New York, NY, USA. *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*, ieee std 802.11-1997 edition, 1997.

[11] Jerry Landt and Barbara Catlin. Shrouds of time - The history of RFID. http://www.aimglobal.org/technologies/rfid/resources/shrouds_of_time.pdf, Oct 2001.

[12] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. *Proceedings of the ACM Symposium on Networked Embedded Systems*, Nov 2003.

[13] Maxim. 1μa supply-current, true +3v to +5.5v rs-232 transceivers with autoshutdown. http://pdfserv.maxim-ic.com/en/ds/MAX3221-MAX3243.pdf, Oct 2002.

[14] PTS (Post och Telestyrelsen). *Lag (2000:121) om radio- och teleterminalutrustning*, 2000.

[15] PTS (Post och Telestyrelsen). *PTSFS 2002:3*, 2002.

[16] G. J. Pottie and W. J. Kaiser. Embedding the Internet: wireless integrated network sensors. *j-CACM*, 43(5):51–58, May 2000.

[17] Wade Roush, Alexandra M Goho, Eric Scigliano, David Talbot, M. Mitchell Waldrop, Gregory T. Huang, Peter Fairley, Erika Jonietz, and Herb Brody. 10 emerging technologies. *Technology Review*, 106:33–49, Feb 2003.

[18] C. Savarese, J. Rabaey, and J. Beutel. Locationing in distributed ad-hoc wireless sensor networks. In *ICASSP*. IEEE, May 2001.

[19] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1):70–70, jan-march 2002.

[20] Philips Semiconductors. *SL1 ICS30 01 I-CODE Label IC Chip Specification Revision 2.1*, May 2000.

[21] Suresh Singh and C. S. Raghavendra. PAMAS-power aware multi-access protocol with signalling for ad hoc networks. *SIGCOMM Comput. Commun. Rev.*, 28(3):5–26, 1998.

[22] D. Stiensra and A. K. Khandani. Iterative multi-user turbo-code receiver for DS-CDMA. In *Iterative Multi-User Turbo-Code Receiver for DS-CDMA*, volume 3, pages 842–846, 2001.

[23] Tagsys. *Medio S001/S002 Command Set Version 1.0*, Oct 2002.

[24] Tagsys. *Medio S001/S002 Product Guide Version 1.0*, Oct 2002.

[25] Crossbow Technology. *MPR - Mote Processor Radio Board, MIB - Mote Interface/Programming Board User's Manual, Rev A*, Aug 2003.

[26] Andrew Tridgell. *Efficient Algorithms for Sorting and Synchronization*. PhD thesis, The Australian National University, Feb 1999.

[27] Berkeley University. TinyOS. http://webs.cs.berkeley.edu/tos/, 2004.

[28] Kamin Whitehouse. *The Design of Calamari: an Ad-hoc Localization System for Sensor Networks*. PhD thesis, University of California at Berkeley, 2002.

[29] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. Technical Report ISI-TR-567, USC Information Sciences Institute, January 2003. Submitted for review to IEEE/ACM Transactions on Networking.

[30] Jong-Hoon Youn and Bella Bose. An energy conserving medium access control protocol for multihop packet radio networks. In *IEEE International Conference on Computer Communications and Networks*, pages 470–475. ICCCN, Oct 2001.

[31] L. Zhong, R. Shah, C. Guo, and J. Rabaey. An ultra-low power and distributed access protocol for broadband wireless sensor networks, May 2001. In IEEE Broadband Wireless Summit, Las Vegas, NV.

Appendix A

I•CODE

I•CODE Label IC is a chip that is used in passive RFID tags in the 13.56 MHz frequency band. It is designed to function as a smart label for logistics and asset tracking. The chip has advantages like anti collision which means that multiple tags can be read 'simultaneously'. The word simultaneously is not quite right because the tags are assigned a time slot within which it can communicate with the reader. So in the reader-tag point of view it is not simultaneously but to us human it is because the communication is so rapid. Depending on the antenna size, tag and the reader the reading range can be up to 1.5 m in no line-of-sight.

The power needed to run the tag is taken from the inductive coupling with the reader. The tag does not even need an own system clock because this is also obtained from the inductive coupling. The I•CODE IC has 512 bit of EEPROM which is divided in 16 blocks of 32 bits each. The first two of these blocks (the first 64 bits) is an ID which is unique for each tag manufactured. This makes it easy to uniquely identify an object because there is no risk two tags have the same ID. The next 32 bits in block three are Write Access Condition Data. This is used to write protect data in the memory, and once it has been write protected it cannot be unlocked anymore. The ID bits is write protected by default and can not be changed. By write protect block three on the chip, which contains the Write Access Conditions, the Write Access Conditions cannot be changed anymore. By doing so it is possible to lock the chip so it is not possible to change the data it contains anymore. Block four on the chip contains some configuration bits that is used to set up the tag and how it responds to the reader.

The I•CODE chip has the advantage that the reader can read multiple tags in the field (anti collision) and it is also possible to write information to the tags. 368 bits are available for the user to hold this information.

A more detailed explanation of how the I•CODE chip is built can be found in [20].

Appendix B

Schematics

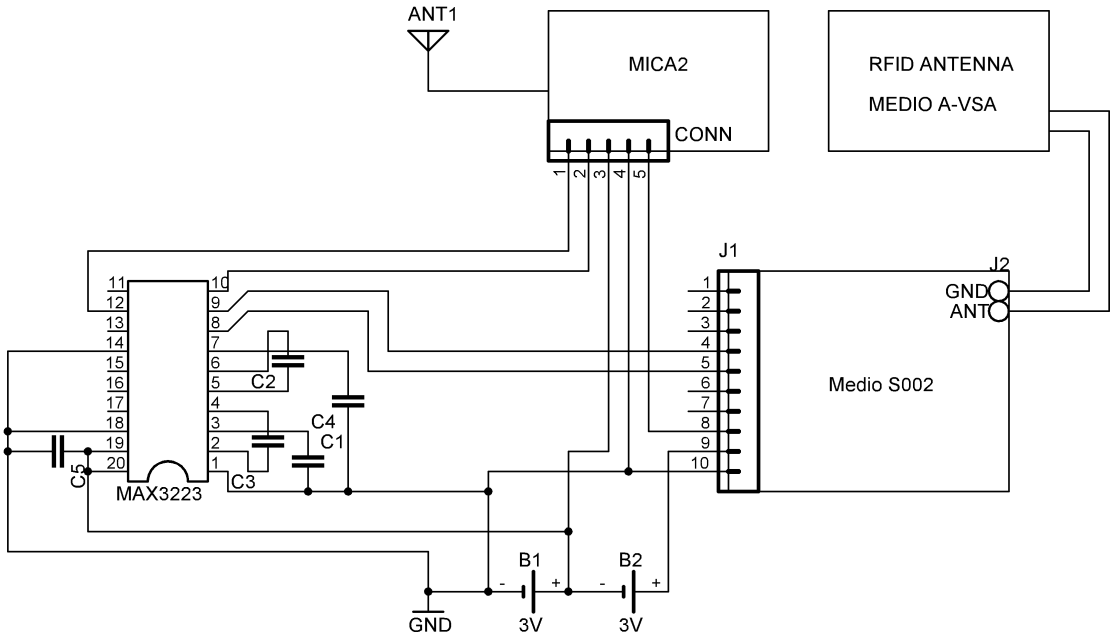

Figure B.1. Schematics

Component List	
Component	*Description*
$MICA2$	MICA2 Platform, See [25] for more information
$MAX3223$	Maxim MAX3223CPP, See [13] for more information
$B1$	Batteries, 2 alkaline AA batteries, 3V
$B2$	Batteries, 2 alkaline AA batteries, 3V
$MEDIO$	Medio S002 RFID reader See [24] for more information
$RFID\ ANTENNA$	Medio A-VSA
ANT	RF Antenna, (Quarter Wavelength Antenna)
$C1$	Capacitor 0.1μF
$C2$	Capacitor 0.1μF
$C3$	Capacitor 0.1μF
$C4$	Capacitor 0.1μF
$C5$	Capacitor 0.1μF

Table B.1. Component List

MAX3223		
Pin	*Name*	*Description*
1	$\overline{EN}$	Receiver Enable Control
2	$C1+$	See [13]
3	$V+$	See [13]
4	$C1-$	See [13]
5	$C2+$	See [13]
6	$C2-$	See [13]
7	$V-$	See [13]
8	T_{OUT}	RS-232 Transmitter Outputs
9	R_{IN}	RS-232 Receiver Inputs
10	R_{OUT}	TTL/CMOS Receiver Outputs
11	$\overline{INVALID}$	See [13]. Not connected
12	T_{IN}	TTL/CMOS Transmitter Input
13	T_{IN}	TTL/CMOS Transmitter Input. Not connected
14	$FORCEON$	See [13]
15	R_{OUT}	TTL/CMOS Receiver Output. Not connected
16	R_{IN}	RS-232 Receiver Inputs. Not connected
17	T_{OUT}	RS-232 Transmitter Output. Not connected
18	GND	Ground
19	V_{CC}	Supply Voltage, 3V
20	$\overline{FORCEOFF}$	See [13]

Table B.2. MAX3223 Pin Assignment

Connector J1		
Pin	*Name*	*Description*
1	*No Connect*	Not Connected
2	Y	See [24]. Not Connected
3	B	See [24]. Not Connected
4	TX_{RS232}	Transmit RS232
5	RX_{RS232}	Receive RS232
6	TX_{TTL}	Transmit TTL. Not Connected
7	RX_{TTL}	Receive TTL. Not Connected
8	$\overline{SHDW}$	Board Hardware Shutdown
9	VCC	Supply Voltage, 6V
10	GND	Ground

Table B.3. Connector J1 Pin Assignment

Connector J2		
Pin	*Name*	*Description*
1	ANT	Antenna Output Pin
2	GND	Reference Voltage for Antenna

Table B.4. Connector J2 Pin Assignment

Connector CONN		
Pin	*Name*	*Description*
1	$UART_{TXD0}$	UART0 Transmit, pin 28
2	$UART_{RXD0}$	UART0 Receive, pin 27
3	V_{CC}	Supply Voltage, 3V
4	GND	Ground
5	$\overline{MEDIO\ SHDW}$	Output Pin Regulating RFID Reader Shutdown, pin 44

Table B.5. Connector CONN Pin Assignment

Appendix C

Packet structures

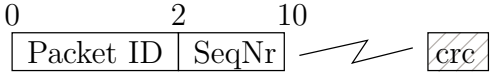

Figure C.1. This is the general packet header. These two fields are always the first two fields in all messages. The first three bits specify which of the following packet types that are being transmitted. The CRC-field is included in the message type, but is unused in the implementation because TinyOS implements it's own CRC-check in the data-link layer.

Figure C.2. This is the general `SYNC`-packet. Sync offset is used if a `SYNC` is delayed. Unused in the current implementation because of the low resolution in the timers (1ms). Payload can include a general byte, such as new potential in case of a change. The flags inform the cluster if there's a potential change coming, if the cluster is about to break or if there's data in the queue.

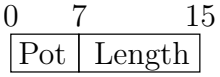

Figure C.3. The packet type used for RTS/CTS communications. Potential is the destination potential, and length specifies the length of the coming communication. The potential actually contains one flag-bit which specifies if the message is intended for the master. All communication is either originating in or targets the master.

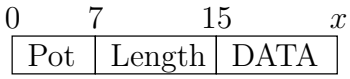

Figure C.4. The general DATA-packet. Length specifies the length of the data-field.

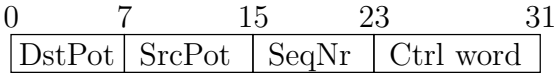

Figure C.5. Control packet type. This is used for changing potentials of other clusters, killing masters, establishing routes, `ACK` and `NACK`. Because these messages generally are triggered by a message sent by another node, the sequence number field is used to uniquely specify which message that this control message is responding to.

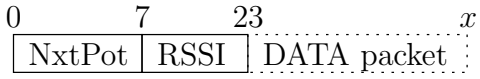

Figure C.6. The general routing packet type. The data message is encapsulated in the routing message. NxtPot signifies the next-hop node or next-hop cluster. The DATA packet is the same as in figure C.4. The RSSI field contains the signal strength from the last hop, which is used in the demo application.

Appendix D

Application call-graph