# Universal Large Scale Sensor Network

Bachelorarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Jakob Schaerer und Severin Zumbrunn
FS 2016

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

# Universal Large Scale Sensor Network

Jakob Schaerer, Severin Zumbrunn and Torsten Braun
Communication and Distributed Systems, Institute of Computer Science
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland

jakob.schaerer@students.unibe.ch, severin.zumbrunn@students.unibe.ch, braun@inf.unibe.ch

***Abstract*— We have created a large scale sensor Network with a node distance of up to 1km, low-power consumption and low cost hardware. The design of the network nodes is modular and application specific sensors can be attached. This makes the network flexible to a wide range of applications. A low power optimized MAC and routing layer allows the energy efficient transmission of information through the network. The network can be accessed by connecting a PC via serial interface to any node. From the selected node the network configuration and the sensor data readout of any other sensor is possible. The sensor network is designed for stand-alone applications in the context of long-term monitoring of environmental conditions in remote areas. We show what characteristics are important to establish and maintain networks with low data rates and long ranges.**

## I. INTRODUCTION

During the past few years, wireless sensor networks (WSNs) with different measurement goals emerged. Most of these WSNs use 2.4GHz radio controllers which limits the distance in between two nodes. Wireless sensor networks that cover large areas [3] are mostly for predefined applications. They often have a fixed set of built in sensors, or consume so much power that it can not be covered by solar cells. To fill the gap between those existing solutions, we created the Universal Large Scale Sensor Network (ULSSN), which is able to transmit information over a large distance, is energy efficient and, has a modular design and allows the attachment of different sensors. With this flexibility the network can be used for a variety of different applications.

We used an integrative approach, which combines hardware and software development to have a perfectly adapted system. Therefore, we designed and developed a printed circuit board (PCB) that features a TICC1110 (low-power, 868MHz, 10dBm, system-on-chip) from Texas Instruments (TI), a debug connector, a connector for various types of sensors, which are connected via Universal Asynchronous Receive Transmit (UART), a 64MB Flash memory, two indicator LEDs and a dc-dc step-up-down converter circuit (Figure 1). The hardware was optimized for low power consumption and is mostly based on the design guidelines from TI [1].

In this work the focus is set on the protocol stack in particular, on the MAC and the routing layer. In section II we discuss the relevance of several existing approaches for the layers of our protocol stack. In the following sections III. and IV. we describe our implementations of the selected protocols and discuss the additionally included features. Finally, section V presents some measurements of our implementation.

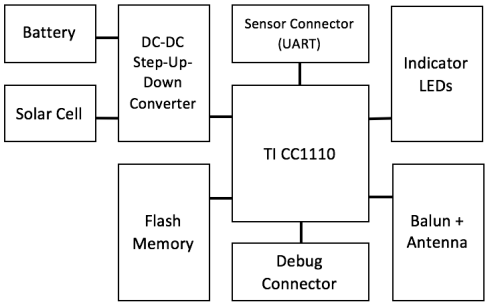


Fig. 1. Hardware overview

## II. RELATED WORK

In general, battery-driven WSNs face the issue of having limited energy capacity. Therefore, many different measures in applications and hardware have been taken to minimize energy consumption for communication. This can be achieved by reducing overhead, adopting the output power to the required transmission range or lowering node-to-node ranges. As there is the need to support longer ranges, we developed a protocol stack, that is built of a PHY, MAC, routing and application layer. This work focuses on the MAC and routing layer.

### A. MAC Layer

In a sensor network multiple network nodes access the same physical medium to exchange information. In order to receive and decode messages from other nodes, every message needs to be sent in a standardized format. The medium access control (MAC) protocol defines how the network nodes access the physical medium. It is the purpose of the MAC protocol to create a reliable connection between nodes and prevent data loss. Additionally, the method with which the medium is accessed can influence the signal range, low power dissipation or data rate. In WSNs the MAC layer needs to focus on low power consumption and collision avoidance.

*1) Low Power Listening:* To optimize the power consumption of network nodes the devices need to keep their transceivers turned off as long as possible. For the transmitting device the time to enable the radio is given by the time the packet is sent. For the receiving node a mechanism is needed to turn on the radio in order to receive the packet. Obviously, the receiving node needs to listen to the channel at the time the packet is sent. To recognize a packet transmission the receiver can probe the channel periodically with an interval $t_i$ (Figure 2). During a probe the receiver turns on it's radio device and checks the channel for activities. When no activity was detected the radio can either be turned off or it can receive the packet otherwise. By periodically probing the channel the problem emerges, that the probe most probably does not start to listen at the beginning of the packet and only a part of the packet is received. To avoid this problem the packet needs to be announced. When the announcement of the packet lasts for $t_i$, it is guaranteed that the listening node is always probing during the announcement time of the sender. This strategy of, probing the channel is called Radio Duty Cycling (RDC).

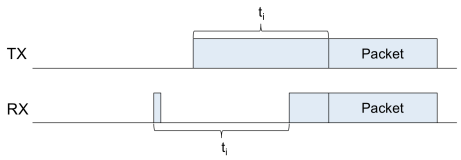


Fig. 2. Radio Duty Cycling

There are several protocols that use radio duty cycling to reduce on time of the radio transceivers. Those protocols can be classified into two groups by the way the transmitting device is announcing the packet. Protocols like WiseMAC [10] and B-MAC [11] use a long preamble to announce the packet and protocols like ContikiMAC [7] and X-MAC [13] send packets as announcements. In section III-B we show the advantages of the preamble over the packet approach.

*2) CSMA/CA:* In a sensor network multiple nodes are sharing one medium, but only one device can send on the medium. When multiple devices are transmitting data concurrently, the signals are mixed and the sent packets cannot be reconstructed at the receiver. To prevent the nodes from sending concurrently, a way to access the medium has to be defined. Most of the wireless devices do not support full duplex mode, thus, they are not able to listen to the channel while they are transmitting and hence cannot detect possible collisions. Without the ability to detect a collision, they need to be prevented by Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA). Collision avoidance is done by probing the channel before a transmission. If the probe detects a free channel, the packet will be initiated, otherwise the node waits for some back-off-time and repeats this step (Figure 4).

### *B. Routing Layer*

A routing layer is responsible for establishing routes inside a network topology and to setup a path over a finite set of nodes. Routing protocols typically have to address changing topologies caused by node failures, reachability changes or interferences but also due to network load and energy constraints of battery-powered nodes.

Since the goal is to gather sensor data from every node by connecting to a single node in the network, there are basically two approaches available for such WSNs. The first is called data dissemination, where data is not actively pulled from the network, but every node advertises it's new information by some ADVERTISEMENT packet containing meta-data that specifies the actual sensor information that has become available. Any other node receiving this packet can check whether it already has the advertised data and if not it may request the complete data packet from the advertiser by initiating a REQUEST packet. Using this approach, data is transmitted thorough the network continuously. This approach is called three-way-handshake and is described in the SPIN protocol [4]. A benefit from this approach is, that no further routing effort is needed as data disseminates through the network automatically. This means that nodes do not have to store additional network information like routing tables and this protocol is obviously not prone to node failures. However, a crucial disadvantage is the necessity for every node to store the complete distributed knowledge of the node set, which leads to excessive memory usage and is therefore not scalable for a larger number of nodes due to the limited memory capacity.

The second approach focuses on the creation and maintenance of routing paths. It is the more common way as it is broadly used in Internet protocols. Usually, in these routing protocols, nodes can be either routers or endpoints. An endpoint can only send data by transmitting it to the next router in the network. A router knows other routers or endpoints to which it can forward the data to or has at least the default gateway in order to send packets to unknown destinations. Since in our application, every node is a potential base station, this means that every node is a router and endpoint at the same time. Usually, every router stores a routing table, which holds cost information and network addresses from all surrounding routers. This information has to be maintained to keep routes valid and detect errors in routing, e.g. loops. Especially in battery-driven WSNs, it is often the case that packets are lost due to interference or link loss. Therefore, WSNs are mostly considered as lossy networks, without the need for every packet to be received.

Another aspect to be considered is, that all routing messages that have to be sent for route maintenance are basically overhead and hence increase energy consumption. To have limited overhead most WSN protocols are simply creating a tree graph, where every node knows it's parent node and passes all sensor data to it. Here the only necessary routing message is the rank-in-graph message, whose rank is increased and resent by every receiver. By this way it is

possible to send unicast messages from every node to the root. But the opposite is not possible due to the missing routing table. One protocol that is based on building such tree graphs is called "Routing Protocol for Low-Power and Lossy Networks" (RPL) [5].

*1) RPL:* RPL is a routing protocol that is designed for low-power and Lossy Networks (LLNs) and operates with constraints on energy and memory. It features multipoint-to-point, point-to-multipoint and even point-to-point mechanisms. RPL has become one of the most important protocols as it is implemented in common operating systems, like Contiki [12] or TinyOS [12] for low-power devices such as WSNs. In RPL, every node may work as a router and endpoint. The RFC6550 - RPL standard [5] was created to have sensor networks that are compliant to IPv6 in order to be interoperable and can be directly connected to the Internet. In our application we have an isolated network connected to a single Personal Computer instead of the Internet.

In RPL there are four modes of operation (MOP). MOP 0 does not maintain downward routes, hence it does not need to store routing tables or additional information other than the single parent it has selected as gateway. MOP 1 is also called non-storing mode and features downward routes in the manner that every node sends it's children addresses to the parent. This way the root node will have the complete knowledge on the topology of the network. But point-to-point (p2p) connections are still not available. MOP 2 and 3 feature such p2p connections.

MOP 0 uses two types of packets for route creation and maintenance: Data Solicitation Information (DIS) and Data Information Object (DIO). The DIO packet is sent by every node in some interval $t$ and holds data about the node position in the tree graph, which is also called Destination Oriented Directed Acyclic Graph (DODAG). This positioning information can be any kind from geo-location to rank metric. An Objective Function (OF) then determines, which node should be selected as a parent based on the lowest transmission costs. While the DIO message is used for tree creation, the DIS message can be used to ask any other node for it's DIO message. Doing so, an old and potentially invalid route can be checked before sending an actual data packet to the root. In order to save energy while keeping the routing information up to date an efficient algorithm for RPL was provided:

*2) Trickle Algorithm:* A Trickle Algorithm [8] is used whenever an application requires to send as few information as possible. This is done by minimizing redundancy, using exponentially increased intervals and fast reactions to invalid information. It is one of the core parts of RPL and is an RFC standard as well (RFC 6206 [8]). It fits perfectly for the DIO transmission in RPL, because it allows rapid adaptation of routing paths and saves a lot of energy by minimizing radio transmissions.

## III. MAC Layer

As mentioned in section II our protocol stack has several different objectives. We set the focus of our Universal Large Scale Sensor Network (ULSSN) to signal range and low-power dissipation.

### A. Datarate-Range-Power Model

The range of the signal is determined by the output power of the device, the device attenuation, the antenna peak gain, the free space path loss, and the receiver sensitivity. The receiver sensitivity is higher at low data rates [1] [2]. The receiver can demodulate a signal with a lower power level on low data rates and more power can be dissipated over the medium. Therefore, a higher distance can be reached with a lower data rate.

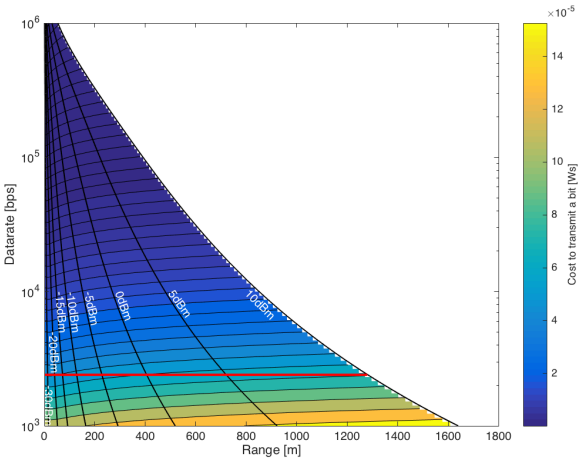


Fig. 3. Datarate-Range-Power model

We have developed a model (Figure 3) to determine the reachable range with a certain data rate and a specific output power level. In this model we assume free space propagation. Hence, the attenuation chain consists of twice the device attenuation $V_{DEV}$ $[dB]$ (measured), twice the antenna gain $V_{ANT}$ $[dB]$ (specification) and the free space path loss (equation 1). $r$ $[m]$ is the distance, $f$ $[Hz]$ is the frequency and $c$ $[\frac{m}{s}]$ is the speed of light. $P_T$ $[dBm]$ is the output power set on the transmitting device. The receiver sensitivity $P_{Rmin}$ $[dBm]$ is given by the radio module (see section V-F).

$$FSPL = 10\log_{10}\left(\frac{4\pi r f}{c}\right)^2 \quad [dB] \tag{1}$$

In order for the receiver to be able to demodulate the signal, inequality 2 must hold.

$$P_T - 2\cdot V_{DEV} + 2\cdot V_{ANT} - P_{Rmin} > FSLP \tag{2}$$

Now we can determine the maximum possible distance for a given output power and data rate.

$$\log_{10}(r) < \frac{1}{20}(P_T - 2\cdot V_{DEV} + 2\cdot V_{ANT} - P_{Rmin})\frac{c}{4\pi f} \tag{3}$$

Figure 3 also shows the energy cost in Joule to transmit a single bit. The costs depend on the data rate, which determines the duration to send a bit and the current consumption (see section V-E) to transmit at a certain power level.

Based on this model (Figure 3) we decided to use a data rate of 2.4kbps (red line). With this data rate we predict that the range between two nodes can reach up to 1.28km in free space. However, depending on the environment this will be limited by trees, houses and other obstacles that hamper the signal propagation.

### B. Low Power Listening

On low data rates, preamble oriented protocols have the advantage that they can send a flexible amount of bits and, therefore, have a flexible interval $t_i$ (Figure 2). The packet oriented announcement strategies lack this flexibility as they need to send full packets, which disables short periods. Thus, we decided to implement radio duty cycling with a long preamble to announce the packets. The preamble is 24 bytes long and lasts for 80ms at a data rate of 2.4kbps. According to the datasheet [1] the receiver has to probe the channel for at least $757\mu s$ resulting in a radio duty cycling of $1\%$.

*Packet detection and Fast sleep:* In low power listening it is important to listen to the channel as short as possible. This is mainly done by probing and checking if there is activity on the channel. When the channel is noisy it can happen that the detected noise is falsely seen as an activity. In this case the power consumption can be reduced by stopping the listening either immediately or when a packet is not addressed to the current node.

In our implementation the device is probing the channel when it wakes up from sleep mode. This probing is done by listening to the channel and comparing the first valid Received Signal Strength Indication (RSSI) with a certain threshold. When the RSSI is below this threshold the channel is assumed to be inactive and the radio is turned off. When the channel power level is above the threshold the device continues to listen. Because the listening probe will always start to listen in the preamble, we can use this preamble to qualify if the activity on the channel belongs to our protocol. This is done by the Preamble Quality Identifier (PQI). The PQI is a counter, which is incremented when two consecutive bits were not equal and it is decremented by 8 otherwise. If there is $PQI \leq 0$ after eight bits, the node assumes that it is listening to noise and switches to idle. When the radio stays active the frame of the packet is received. Here we can save energy, if we only listen to packets that are addressed to the current node. This is done by comparing the first address byte of the packet with the first byte of the own address. If it is a match, the node keeps listening for the complete packet.

### C. CSMA/CA

In our ULSSN we use a very simple CSMA/CA algorithm with channel probing and backoff-time (Figure 4). To probe the channel the radio switches to receive (RCV) mode and waits until a valid RSSI sample is obtained. At the data rate of 2.4kbps, this probe needs $757\mu s$ until a valid RSSI level is established. If the RSSI is above a certain threshold it is assumed that the channel is occupied and the device waits for 80ms (preamble time) backoff-time and repeats the process until the RSSI is below the threshold and it can transmit the packet. With an additional random delay before the first probing, we prevent collisions after a broadcast request where all nodes probe the channel at the same time, assume it as free and then transmit concurrently. This random delay lasts up to 5% of the preamble time and extends the transmission time of an average packet by up to $1\%$.

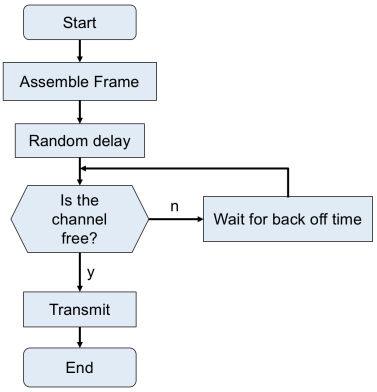


Fig. 4. CSMA/CA

### D. Power Adaptation

The output power of the transmitter defines how far the signal will reach. In a sensor network the distance between two nodes is not always the same and reflections may hamper signal propagation. To prevent oversaturation of the receiving node and to save energy, the output power should be set to a level low enough that the signal can still be demodulated on the receiving node. So, the optimal output power for every link is different.

The implementation of the power adaptation affects all non-broadcast and non-acknowledgement packets. The band between -90dBm and -60dBm defines the desired power level on the receiving node. When a received packet is not within this band a request to adapt the power is sent to the source node of the packet. Initially, all packets are sent using the defined standard power level. This level can be set to a value that fits to the particular application and is lower for small scale than for large scale networks. Each node maintains a list of its neighbor nodes and the according power levels. If a node receives an adaptation request, it adapts the power level of the requesting node in its own neighbor list.

## IV. Routing Layer

In our protocol stack we have identified RPL as the foundation for our routing protocol as it features lightweight methods. However, as it was stated by Clausen et al., common implementations of RPL use more than 50kB of memory [6]. Thus, we had to focus on program size since our microcontroller features only 32kB of Flash memory. It is best practice to reduce memory usage by removing

unnecessary functions and information. Therefore, we did an analysis on the functionality range and discovered that for our application we require only a limited set of functionality to get a working system. This not only reduced the program memory size itself but also the overhead of our packets. With this measure we were able to decrease the DIO and DIS packet size by 84%, which results in total energy savings of 1.8mJ per packet. This can be seen in Figure 5, which depicts the modified RPL packet format as needed for our application-tailored RPL version.

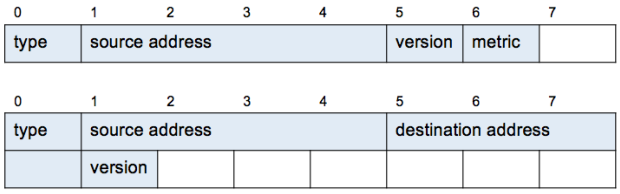


Fig. 5. Modified DIO and DIS packet definitions

We also restricted ourselves to the minimum functionality of RPL by choosing RPL Mode 0, since we do not require peer-to-peer (p2p) connections. Though, we want to be able to send downward configuration messages from the base station. This is done by an additional functionality described later in section IV-B. In our application, the parent selection using an Objective Function (OF) is done by comparing the rank in the tree of different nodes and does not include any other metric. It would be conceivable to define the OF so that battery level, link quality indicator and other metrics are considered. But since we only have one type of sensor node in our application, they share similarity in battery and memory capacity, calculation speed etc. Therefore, it is a viable approach to keep the OF as simple as possible.
Additionally, we limit the number of different DODAGs to 1 and thus, can completely remove the functionality for distinguishing DODAGs. This has a direct influence on the parent selection method, because a node that currently does not know about the DODAG will simply send a DIO message of maximum (infinite) rank. If a node receives a packet containing invalid DODAG information, e.g. infinite rank, it may respond with a DIO message that holds it's current finite rank in the DODAG. This way, the consistency of a DODAG is always guaranteed and disconnected sensor nodes can join the DODAG automatically.

Another common way of saving energy is extending the transmission interval for routing messages. However, when lowering the timeliness of routes, erroneous transmissions can increase due to the possible obsolete routes nodes hold. Therefore, only extending intervals is not the appropriate solution but together with a mechanism that also addresses the issue of obsolete information it is possible to save a lot of energy. This is done with the so called Trickle Algorithm as it is described by Levis et al.[8].

### A. Trickle Algorithm

The Trickle Algorithm is a very simple process described in a few steps:

1) A value $I$ is randomly selected out of an Interval $[I_{min}, I_{max}]$, where $I_{min} \in \mathbb{N}$ and $I_{max} \in \{I_{min} * 2^n, n \in \mathbb{N}^+\}$. Two variables, the consistency counter $c$ and the time counter variable $a$ are set to zero and a redundancy constant $k$ is defined. The value $k$ describes the amount of redundant routing packets we tolerate in our network.
2) When an interval begins, $c$ and $a$ are set to zero, a transmission variable $t$ is chosen randomly from our interval $[I/2, I)$ and $a$ then starts counting up.
3) For every redundant packet (a packet that does not change our own configuration, e.g. that comes from a parent node and holds the same cost as we already know), we increase variable $c$ by one.
4) At time $a = t$, if $c < k$ we initiate a DIO transmission. Otherwise, the transmission is suppressed due to too much redundancy in our network.
5) When $t > I$, we double the interval $[0, I]$ to $[0, 2 * I]$ until $I > I_{max}$ then, we simply set the interval to $[0, I_{max}]$.
6) At any time, if we receive an inconsistent packet, we reset the Trickle Timer by setting the interval size back to $[0, I_{min}]$ and continue with step 2.

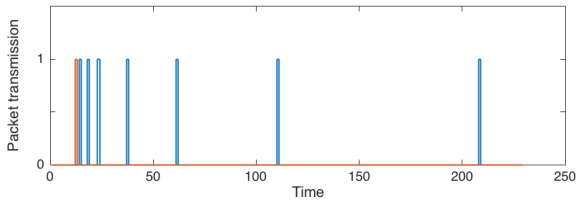


Fig. 6. Exponentially incremented interval of a Trickle Timer

With these six steps we are able to send only as few information as needed. Let us assume that we have a network of four nodes that are all interconnected and one of them is our base station, which in our terms means that it has rank zero and all others have rank 1. Let $k = 1$. Now when the base station initiates a DIO message, all other nodes will receive it and increment their value $c$ by 1. This way no other node will transmit it's information since there was already enough redundancy sent. Next, every node will double it's interval and only one node (the one with the lowest value of $t$), will initiate another DIO message and so on. Therefore, in a perfect environment without any disturbances, we end up in sending only one routing message per hour, which results in having a low-power routing protocol.

### B. Downward routes

To have the ability to send configuration messages from the root node into the network, we were constrained to use additional mechanisms as RPL Mode 0 does not maintain downward routes. Since every configuration packet is sent

as a broadcast message, it will be re-sent multiple times by every node. This causes a lot of overhead and will finally result in deadlock as the transmission buffers of the routing protocol overflow with copies of the same packet. In order to distinguish between upward and downward traffic we have added an ID to every configuration packet to identify the source where the packet came from. With that measure we are reducing the transmission of redundant information because everytime a packet is received from a lower rank the packet should be forwarded downwards as it originated from root. By this simple algorithm, nodes that are on the same level will not respond to packets from each other as those are lossy broadcast packets. This mechanism not only provides a simple way for sending messages downwards the tree, it also saves a lot of energy due to it's low overhead and fast algorithm for checking the timeliness of a packet.

## V. Measurements

### A. Low Power Listening

To measure the current drawn by our device we used a micro benchmark as proposed by A. Dunkels [7]. We measured the voltage over a 10 Ohm shunt resistor with a Teledyne LeCroy HDO6104 oscilloscope to determine the current consumption of the node. Figure 7 shows the current consumption of a listening probe. In this measurement three parts can be identified. First, the controller starts up from sleep mode. In this part timers are updated and direct memory access channels are initialized. This wake-up of the operating system needs $\sim 590\mu s$. After wake-up, the radio device is calibrated. This is done at every 4th start of the radio and needs $\sim 754\mu s$. In the last step the radio is listening to the channel for $\sim 930\mu s$.

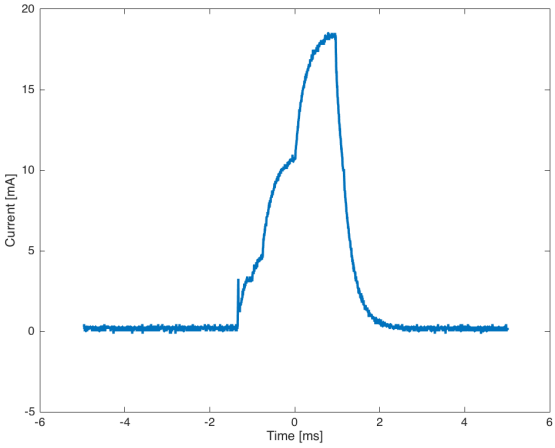


Fig. 7. Listening Probe

Using this measurement we could determine that waking up from sleep and probing the channel needs $10.5mA$ on average and lasts for $1.7ms$. The current consumption in sleep mode is $198\mu A$. With a listening interval $t_i = 80ms$ the resulting average current consumption is $420\mu A$ as shown in equation 4.

$$\frac{1.7ms \cdot 10.5mA + (80ms - 1.7ms) \cdot 198\mu A}{80ms} = 420\mu A \quad (4)$$

### B. Range Verification

To verify that we can reach a distance longer than 1km between two nodes we sent packets over this distance. Therefore, we have chosen an area with only a few obstacles and the possibility of a line of sight (LOS) transmission over 1km. With this setup we were able to establish a connection over a distance of 1.03km and transmit packets with a packet error rate of $\leq 4\%$.

### C. Power Adaptation

To verify the power adaptation algorithm we have created a simple test setup, consisting of three nodes. Two nodes A (red) and B (blue) are transmitting data and a third node C (white) is listening to the channel and logging the power level as it can be seen in Figure 8.

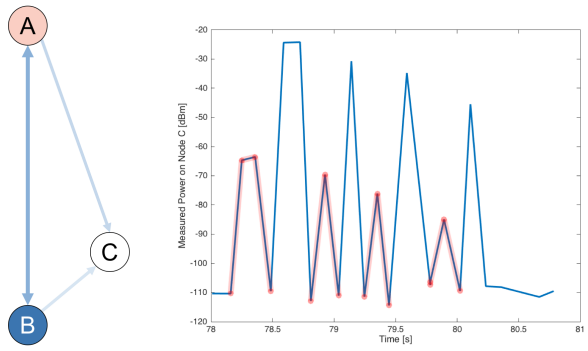


Fig. 8. Power Adaptation - Test setup and Measurement

To test the power adaptation, node A sends a packet to node B. Node B receives the packet and the power level is above the desired power band. Node B sends a request to A to reduce its output power. Node A adapts the output power for node B, but the request packet was above the desired power band too. So A requests B to lower its output power. This leads to a negotiation until both levels are within the power band and the output power is optimally adopted.

### D. Response Times

To measure the average response times, we implemented a PING application [9] on the application layer. PING is used for sending echo request messages to test if there are other nodes available.

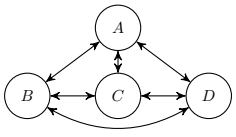


Fig. 9. Topology for PING measurement

In standard IP, nodes can be requested by initiating a PING message with either their direct, subnetwork or broadcast

address. Since we did not want to query single nodes, we only implemented broadcast support with a time-to-live (ttl) field. Following this strategy we are able to detect the nodes on different levels of the tree, e.g. with a ttl value of 2 we would only receive responses from nodes on either rank 1 or rank 2, but no messages from higher ranks.

When a PING request is initiated, a timer is started until a certain time has passed. For every packet, we log the time it took until the PING response was received. With this application it is not only possible to see if the routing and all corresponding layers work together, but also to have a tool available to measure the actual response times.

| | |
|---|---|
| 1:1 | 1065.9ms |
| 1:2 | 1701.1ms |
| 1:3 | 2470.2ms |
| 1:4 | 2752.5ms |

Fig. 10. Average response times for a fully meshed network with 2,3,4 and 5 nodes

Figure 9 shows the example topology $1:3$ used for measuring the response times. The notation $1:n$ is read as 1 base station is connected to $n$ nodes, where the base station is the node that initiates the PING. From the measured average response times (Figure 10) we can clearly see that the response time increases by around 583ms for each additional node. This is due to the fact that by adding a node $(n+1)$, the last packet is sent after a total time of $n * t_{window} + t_{window}$, where $n$ is the number of nodes and $t_{window}$ is the time slot for one PING packet, defined by parameters of the MAC layer.

### E. Transmission Costs

The costs in Joule to transmit a bit is given by the time to send the bit ($420\mu s$ at 2.4kbps), the voltage (3V) and the current consumption of the network node. We have used the same setup as shown in section V-A to measure the current at different output power levels. With these measurements we are able to approximate a transmission cost function.

Figure 11 shows the current measurements at different output power levels. We used the hypothesis function (5) to find an approximation to these points.

$$h_\theta = \theta_0 + \theta_1 x + \theta_2 x^2 \tag{5}$$

For the hypothesis $h_\theta$ function we found the parameters $\theta_0 = 2.45 \cdot 10^{-2}$, $\theta_1 = 0.73 \cdot 10^{-3}$ and $\theta_2 = 1.4 \cdot 10^{-5}$. With those parameters the hypothesis function $h_\theta$ has a mean squared error as shown in equation (6), where $m$ is the number of samples, $x^{(i)}$ is the output power of sample $i$ and $y^{(i)}$ is the current consumption of sample $i$.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = 0.95 \tag{6}$$

Let $I(x) = h_\theta$ for the value of $\theta$ found above, then $I(x)$ in Ampere is the current consumption as a function of the output power. With function (7), where $x$ is the output power

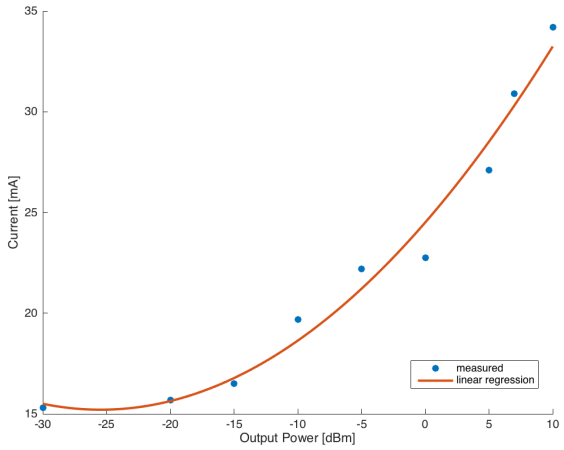


Fig. 11. Current consumption at different output power levels

in [dBm], $s$ the data rate in [bps] and $u$ the supply voltage in [V], we can predict the cost to transmit a single bit.

$$f(x, s, u) = I(x)\frac{u}{s} \tag{7}$$

This function was used in section III-A to estimate the cost to send a single bit over a certain range.

### F. Receiver Sensitivity

To estimate a continuous receiver sensitivity function we use linear regression (equation 8) on the receiver sensitivity dataset in the specification of TI CC1110 [1] [2] (Figure 12).

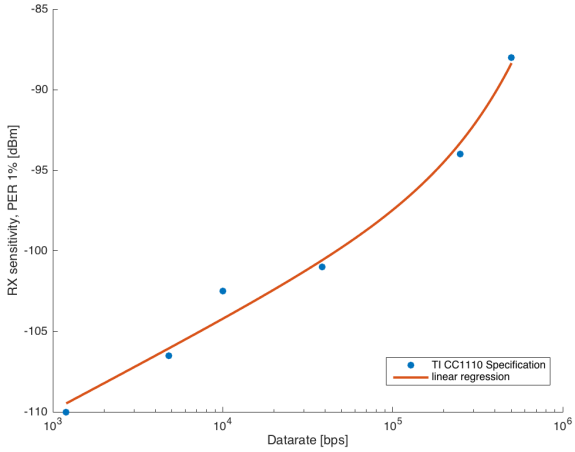


Fig. 12. RCV sensitivity approximation

$$h'_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 log(x) \tag{8}$$

For this hypothesis $h'_\theta$ function we found the parameters $\theta_0 = -1.267 \cdot 10^2$, $\theta_1 = 1.303 \cdot 10^{-5}$ and $\theta_2 = 2.423$. With these parameters the hypothesis function $h'_\theta$ has a mean squared error as shown in equation (9), where $m$ is the

number of samples, $x^{(i)}$ is the data rate of sample $i$ and $y^{(i)}$ is the sensitivity of sample $i$.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h'_\theta(x^{(i)}) - y^{(i)})^2 = 0.7 \tag{9}$$

## VI. CONCLUSIONS

This work contains the development and implementation of an ad-hoc wireless sensor network. We propose a protocol stack including a MAC and routing protocol for a low-power and lossy sensor network featuring long transmission ranges. Our design provides a way of transmitting sensor data with minimal costs over maximum range. We showed that an integrative approach where every aspect that accounts for the power usage can be very effective in terms of optimizing the overall energy consumption.

## REFERENCES


[1] Texas Instruments "CC1110Fx / CC1111Fx". "Low-Power SoC (System-on-Chip) with MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller". January 2006.

[2] Sverre Hellan "CC11xx Sensitivity versus Frequency Offset and Crystal Accuracy". August 2009.

[3] Geoffrey Werner-Allen et al. "Deploying a Wireless Sensor Network on an Active Volcano". IEEE Internet Computing. March 2006.

[4] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," Proc. 5th ACM/IEEE Mobicom Conference (MobiCom '99), Seattle, WA, August 1999.

[5] T. Winter, et al., "RPL: IPv6 Routing Power for Low-Power and Lossy Networks". RFC 6550. March 2012.

[6] Clausen et al., "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)" IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2011. October 2011.

[7] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol". 2011.

[8] P. Levis, et al. "The Trickle Algorithm". RFC 6206. March 2011.

[9] R. Braden "Requirements for Internet Hosts – Communication Layers". RFC 1122. October 1989.

[10] El-Hoiydi, A., and J. D. Decotignie. "WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks." Ninth International Symposium on Computers and Communications, 2004.

[11] Polastre, J., J. Hill, and D. Culler. "Versatile Low Power Media Access for Wireless Sensor Networks," SenSys'04, 95–107, 2004.

[12] Ko, JeongGil and Eriksson, Joakim et al. "ContikiRPL and TinyRPL: Happy Together." In: Extending the Internet to Low power and Lossy Networks (IP+SN 2011), Apr 2011.

[13] Buettner, Michael, Gary V. Yee, Eric Anderson, and Richard Han. "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks." In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, 307–20. SenSys '06. New York, NY, USA: ACM, 2006