COMPARISON OF TDMA AND CONTENTION BASED MAC PROTOCOLS ON EMBEDDED SENSOR BOARDS

Masterarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Markus Anwander
2006

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

Contents

List of Figures

List of Tables

Abstract

With the goal of reducing energy consumption in Wireless Sensor Networks (WSNs), several energy-saving medium access control (MAC) protocols are in use today. In this thesis, the performance of two different energy efficient MAC protocols are compared on limited hardware. Both the TDMA-based LMAC and the contention-based TEEM protocol were implemented on ESB nodes developed by ScatterWeb. Experimental results on energy consumption, packet loss, packet delay variance and bit errors were used for the evaluation and the analysis of the energy saving capability of the two protocols. These results show that, within our test networks, TEEM was superior to LMAC in energy consumption, packet loss and average packet delay. By comparing the performance of two protocol types on limited hardware, the results should be useful to many real-world implementation of such protocols. Further, the results suggest areas for future research and implementation.

Acknowledgement

I wish to express my gratitude to all persons for helping me realizing this thesis. Special thanks go to Prof. Dr. Torsten Braun for supervising me and giving me the possibility of writing this thesis. I am also much obliged to all the people in the research group who supported me in different topics, especially Thomas Bernoulli and Thomas Staub for supporting me and proof reading. Furthermore, I want to thank Morten Anderson and Daria Spescha for proof-reading. Finally, I would like to thank my mother and my girlfriend Nadine Nigg for their support.

Chapter 1

Introduction

Wireless sensor Networks (WSNs) have a large area of applications: event detection, localization, tracking, building monitoring and many more. A WSN often consists of a huge number of nodes, randomly distributed in a large area. Normally, a sensor node has a micro-controller, some sensors, and a low-power radio for communication. Usually the nodes are battery powered, which makes energy efficiency an important issue.

The transceiver for wireless communication is one of most power consuming module of a sensor node. In order to reduce energy consumption, the research community has developed a number of energy saving medium-access control (MAC) protocols. The main strategy for saving energy is the reduction of the listen time and the shutdown of the radio module while unessential information is sent by other nodes.

The developed MAC protocols can be mainly categorized as either ”time division multiple access” (TDMA) based or contention based. TDMA based protocols allocate to each node an exclusive time-slot for communication. In these time-slots collision-free media access is guaranteed. This behavior allows, in comparison to contention based protocols, to reduce transceiver state switches and preamble transmissions to save more energy. Contention based protocols need usually a RTS/CTS exchange to enable collision-free media access. On the other hand, contention based protocols afford longer slots with a larger sleeping part.

During the work for this master thesis two energy saving protocols are implemented. The ”Lightweight Medium Access” (LMAC) [1] protocol as a TDMA protocol and the ”traffic aware, energy efficient” (TEEM) [2] protocol as a contention based protocol. For comparison with a non energy saving protocol, the ”ScatterWeb” [3] protocol, as a simple implementation of a carrier sense multiple access (CDMA) based protocol is used.

Hardware limits demand comparing different MAC protocols in real world experiments on real sensor hardware. Larger distances between nodes, as well as scores of radio interfaces which enhance background noises in medium leads to an unserviceable network behavior. Basically the low-power radio module necessitates some modifications for stable working protocol implementation. Especially the TDMA based LMAC protocol, which needs a very accurate synchronization, looses a lot of it’s energy efficiency for compensating effects of limited hardware. For instance at a low signal-to-noise ratio the network mesh tends towards building unconnected clusters, which must be avoid with longer radio listen ing times. For contention based protocols, the radio switch time is a problem. To ensure oscillator synchronization between individual

transceivers, a transmitter has to send always some preamble bytes before recognizable data bytes can be transferred. This requires several milliseconds at data rate of 19.2 kbps, which is often used in WSNs. Meanwhile a collision detection is impossible and forces to periodical energy wasting retransmissions.

The chapters of this master thesis are organized as follows. While in the first part of Chapter 2 some common energy saving MAC protocols are described, the second part contains a detailed description of the three implemented MAC protocols. Chapter 3 specifies the used hardware and describes problems occurred and experiences made during the implementation. Conducted experiments and achieved results are presented in Chapter 4. Finally the main conclusions of this thesis are drawn and possible future research is discussed in Chapter 5.

Chapter 2

Related Work

In this section, different energy efficient media access control (MAC) layer protocols for wireless sensor networks (WSN) are presented. They can be mainly categorized as either ”time division multiple access” (TDMA) based or contention based MAC protocols. A main strategy of these protocols to save energy is to kept the periods of active radio transceiver as short as possible and shut down unused radio transceivers immediately.

TDMA-based protocols allocate each node an exclusive time slot for data transmissions. The slot owner hold sole access rights on the medium during this time interval. This guarantees a collision-free communication without an energy wasting contention period. TDMA protocols require less transceiver state switches and are generally more energy efficient than contention based approaches. Disadvantages are a less flexible adaptation on actual network traffic load, the necessity of a precise time synchronization, a complex procedure to add new nodes to an existing network, and a limitation on the maximal number of nodes within a certain area.

In contention based protocols the slots are dynamically allocated to the node during a transmission preceding competition between nodes which want to send data. The random backoff time during the contention period can compensate the effects of small clock shifts or slightly imprecise synchronized time stamps. A *ready to send* (RTS) packet and *clear to send* (CTS) packet exchange as explained in Figure 2.1 is used to enable a collision free communication and to prevent the hidden node problem. Node A want to send data to node B and tries to allocate the medium by sending a RTS. The RTS includes a network allocation vector (NAV). During this NAV time interval other nodes should not initiated an own transmission. In case that node B is able to receive data, it send a CTS back to node A. For node C which is hidden from node A, the CTS contains also a NAV to prevent node C starting an own transmission and jamming node B. A disadvantage of this approach is an additional energy consumption for RTS and CTS packets. Furthermore, every packet needs several preamble bytes to tune the radio oscillator of a receiver before they can be transmitted. A node can not listen to the medium during switching from listening to sending. Therefore collisions are possible if a RTS or CTS packet arrives during a transceiver state switch.

The following sections explain the contention based protocols S-MAC [4], TEEM [2], T-MAC [5] and BitMAC [6]. There are many other approaches of energy efficient contention based MAC protocols for wireless sensor networks like WiseMAC [12], PAMAS [13], LEACH [14], DSMAC [9], ACMAC [10], and MS-MAC [11]. TEEM, T-MAC, DSMAC, ACMAC, and

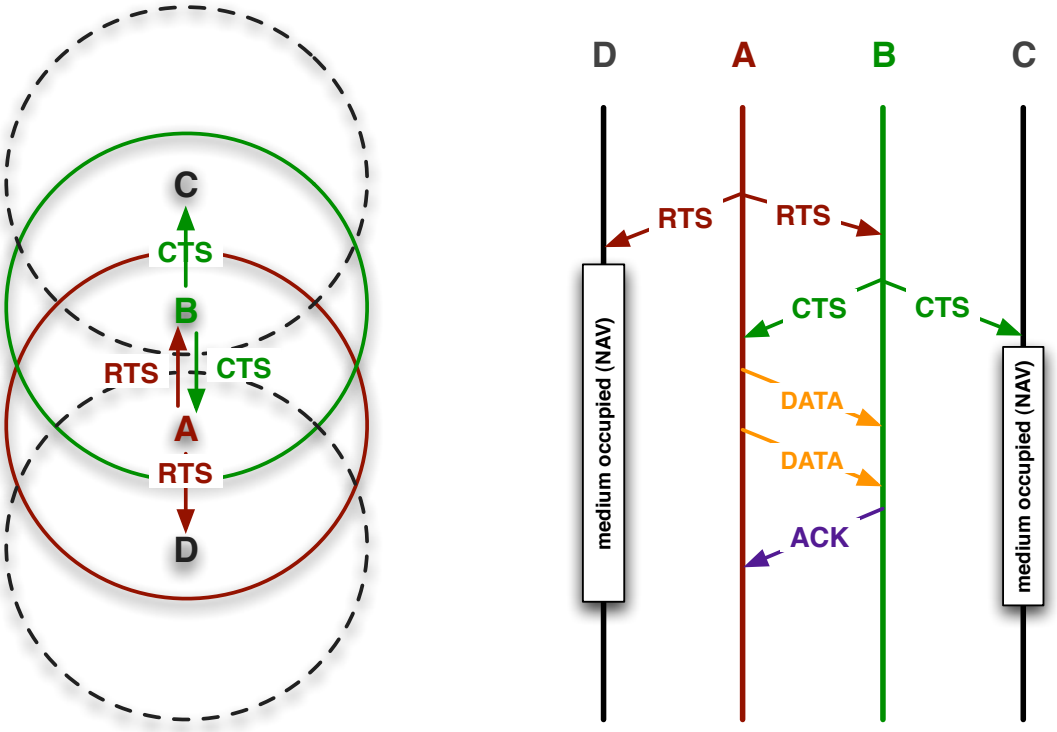


Figure 2.1: RTS/CTS exchange to enable a collision free communication.

MS-MAC are variants of the contention based S-MAC protocol.

LMAC, TRAMA [7], and B-MAC [8] are the TDMA based protocols presented in the following subsections. Additional examples for TDMA based protocols are AI-MAC [15], EMACS [16], DE-MAC [17], and Z-MAC [18].

There exist implementations of S-MAC and TEEM on Berkeley Mica Mote based on TinyOS [19]. But to our best knowledge, there are no implementations of TEEM and LMAC made on the scatterweb hardware platform.

2.1 S-MAC

Sensor-MAC (S-MAC) [4] is the first MAC layer protocol for sensor networks with considering energy limitations of battery driven sensor nodes. The main idea for saving energy is to turn off the radio transmitter during no relevant communication is taking place. The authors also try to reduce energy wasting collisions and control overhead. In periodical listen/sleep cycles, as illustrated in Figure 2.2, the nodes exchange synchronization messages (SYNC) during the listen period and shutdown the radio device during the sleep period. If a node wants to send data,

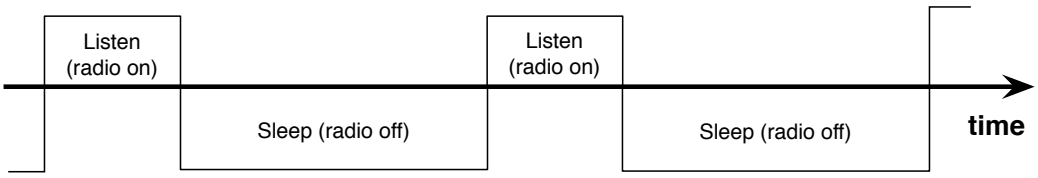


Figure 2.2: Listen/sleep cycles in S-MAC. During the sleep cycle the radio is completely to save energy.

it has to sent a RTS request during the receivers listening period. The longer the sleep periods are, the more energy can be saved. A sender must wait longer until the receiver wakes up. Every node has its own listen/sleep cycles. In order to determine an energy efficient listen/sleep cycle after engaging in a network area, the node first listens for a certain amount of time to its neighbors. In case of receiving a SYNC message the node follows the listen/sleep cycle specified by this SYNC message. Such a node is called *follower*. If the node does not receive a SYNC, it chooses its own listen/sleep cycle and starts broadcasting SYNC packets. Such a node is called *synchronizer*. A SYNC packet is quite short, and provides information about the sender's address and subsequent sleep length. Every node sets up a *schedule table* that holds the schedules of all known neighbor nodes. To ensure that all nodes can send a SYNC for transmitting their schedules to the neighbor nodes, the *follower* nodes waits after receiving a SYNC for a random delay and broadcast their own schedules transmitting a SYNC packet.

Figure 2.3 shows the timing relationship of three different possible scenarios in S-MAC. In scenario A the sender first broadcasts a SYNC. A RTS request for sending DATA is answered by the receiver with a CTS packet. Now the sender starts to send the DATA and waits for the ACK. Scenario B is equal, except the sender does not send a SYNC, it is sent by an other node. In scenario C the sender broadcast a SYNC without a data transmission. S-MAC avoids collisions by using a NAV and the RTS/CTS exchange scheme. An ACK for each DATA packet enables

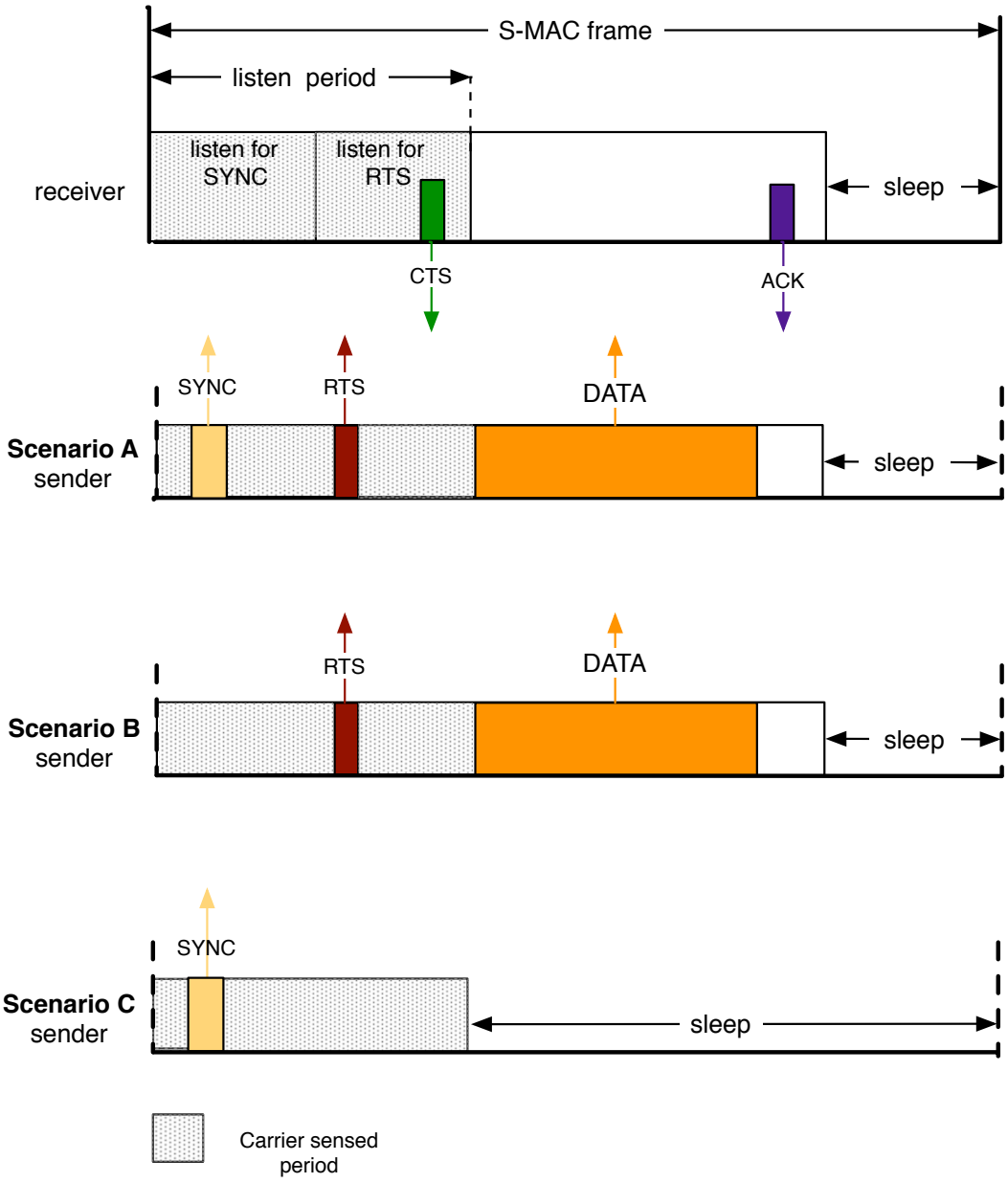


Figure 2.3: Three possible scenarios between a receiver and sender.

retransmissions for lost packets caused by interferences or collisions. After transmission failure, the node sleeps until the receiver will be ready again to receive data.

Using this scheme, nodes build virtual clusters on common schedules. Figure 2.4 shows that nodes at the periphery of a cluster, which have to send a packet to a node with a different listen/sleep cycle wake up at the listen period of the receiver and try to allocate the medium with a RTS/CTS exchange. These nodes have a higher energy usage caused by more listen intervals they have to follow and can run out of battery earlier.

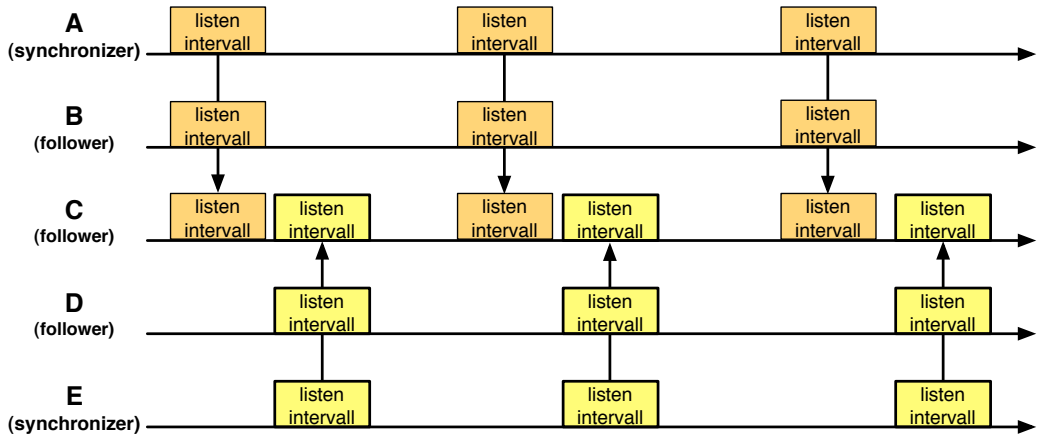


Figure 2.4: Node C in the periphery of two virtual clusters has more listen intervals and a higher energy consumption.

The authors of [4] implemented applied in there experiments a sleep time of 1 second and a listen time of 300 milliseconds. The *schedule table* was updated every ten listen/sleep cycles (13 seconds).

2.2 TEEM

The traffic aware, energy efficient (TEEM) [2] protocol is an enhancement of S-MAC with two improvements. First, nodes have no fixed duration of listen times and turn off their radio earlier when no data traffic is expected. Second, the transmission of a separate RTS control packets is avoided by sending a $SYNC_{rts}$. Figure 2.5 shows the merging of the SYNC and RTS packet to one $SYNC_{rts}$ packet.

Sending only one $SYNC_{rts}$ packet saves listening and sending time and several separate preamble bytes for the RTS packet.

The listen period is shorter than in S-MAC and is not fixed. It is divided into an initial $SYNC_{DATA}$ and a succeeding $SYNC_{NODATA}$ part. If a node has data to send like shown in Figure 2.6(a), it tries to send the $SYNC_{rts}$ for allocating the medium quite after slot start in the contention based $SYNC_{DATA}$ period. All receivers of a $SYNC_{rts}$ stop trying to send an own $SYNC_{rts}$ or SYNC and use received ”sleep time” value for synchronization. Not addressed nodes turn their radios off. The receiver node addressed in $SYNC_{rts}$ sends back a CTS and

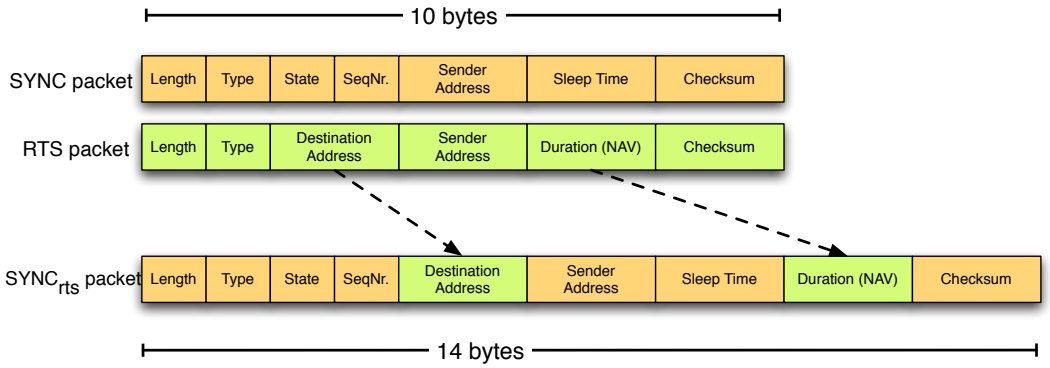


Figure 2.5: A combination of a SYNC and a RTS packet enables to save several redundant bytes.

waits for the incoming data packet. A successful transmission is approved by an ACK packet.

Alternatively, if none of the nodes has data to send, like illustrated in Figure 2.6(b), they try to send a SYNC in the succeeding $SYNC_{NODATA}$ period. This ensures that nodes with data have a higher priority to allocate the medium.

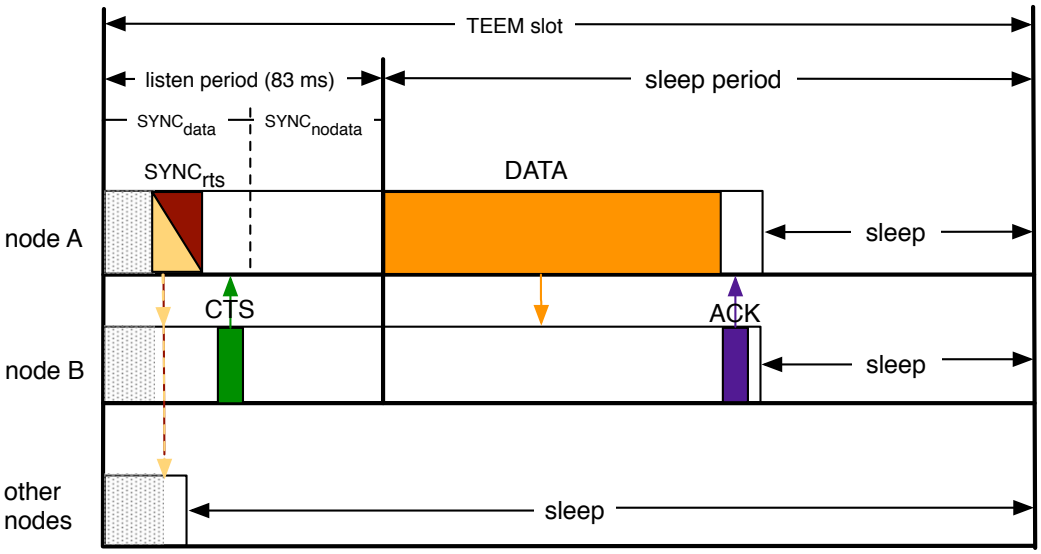
Collisions can occur if a node starts to send a RTS or a CTS during a neighbor node is switching its transceiver from listening to sending mode. In this case, the neighbor was not able to detect the transmission and starts with sending too.

2.3 T-MAC

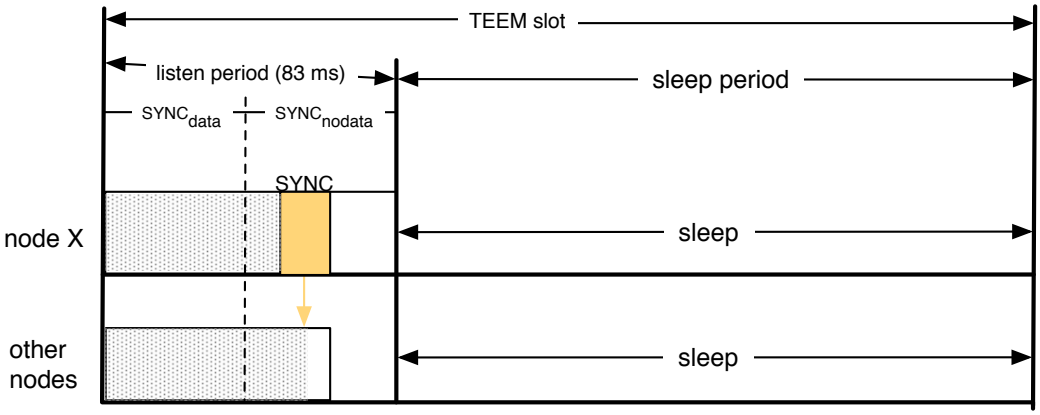
Timeout MAC (T-MAC) [5] is a contention-based MAC protocol. It tries to enhance S-MAC by having duty-cycles of adaptive lengths by dynamically terminating the listening period. After a given interval with no transmission, a timeout turns the radio off. It uses the same *schedule table* algorithm as S-MAC to get to know neighbors listen/sleep cycles. The RTS/CTS/DATA/ACK schemes are applied to avoid collisions and to ensure retransmission of corrupt or missing data. The sender retransmits a RTS if it receives no CTS. If the receiver still does not answer, the sender goes to sleep. In contrast to S-MAC, all data in queue is send in a burst at the beginning of a frame. Early timeout in cases of contention may decrease the throughput of T-MAC, i.e. a node may go to sleep too early if its neighbor lost contention, but still have data to send to this node.

Figure 2.7 illustrates the *early sleeping* problem in S-MAC multi-hop networks and a possible solution with a *Fist the future-request-to-send* (FRTS) approach. Node A sends a RTS packet to node B to initiate a data transmission. At the same time node C wants to send data to node D which is not in the transmission range of node A and B.

Node C can not sent a RTS to node D, otherwise a collision will occur at node B. Therefore, node D detects no activity and turn the radio off. This problem can happen in several next duty-cycles. With the FRTS node C informs node D that it has pending data for it, but is prohibited from using the medium. During this period node A sends a empty data-send (DS) to prevent any other node from taking the channel.



(a) Node A has to sends data to node B.



(b) None of the nodes have data to send.

Figure 2.6: Basic TEEM mechanisms.

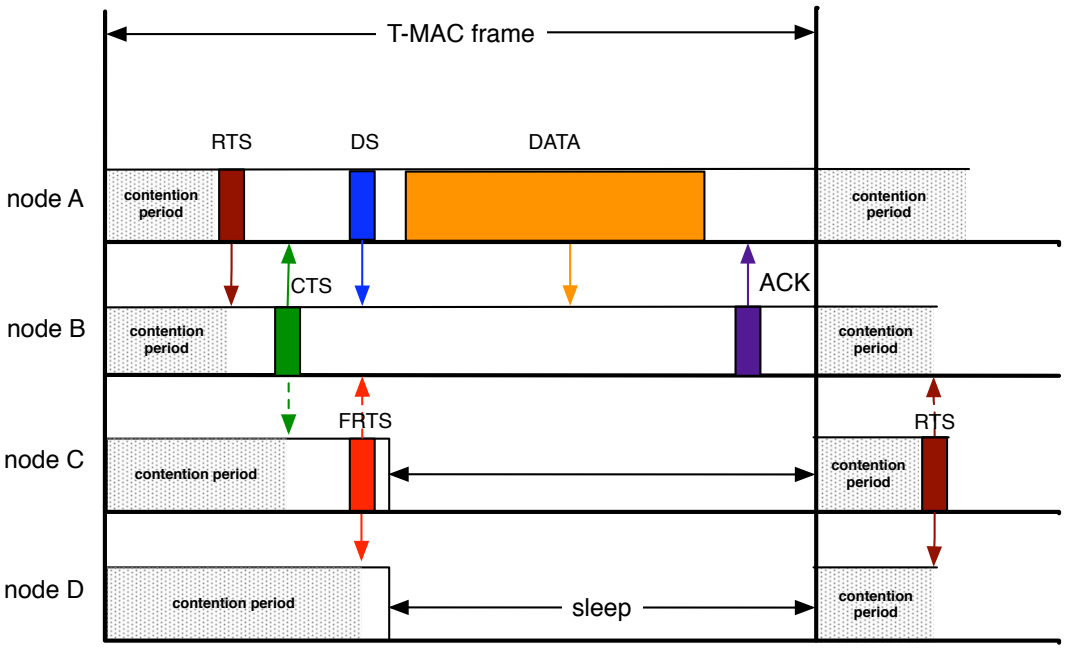


Figure 2.7: Basic T-MAC mechanism with listen/sleep cycles..

Another solution for this problem is a priority mode, which prompts nodes with full buffers to prefer sending to receiving. They ignore a received RTS packet and send an own RTS to initiate a data transmission. This behavior leads to the prioritization of the nodes with full buffers. This may work well in one-way communication to a sink.

2.4 BitMAC

BitMAC [6] is a collision-free, robust, TDMA based MAC protocol designed for dense networks with low mobility. The communication model supports concurrent transmissions by a ”bitwise” transmission scheme. Therefore, strong synchronization and an on/off-keying scheme are used. BitMAC builds a spanning tree consisting of hierarchical connected star networks. Within each star network the medium access is performed in time-slots. Neighboring star networks use different channels to access the medium. Time-multiplexing is used on demand, whenever data has to be sent. The parent node indicates each round with a beacon message and the child nodes answer with a bit-vector whether they have data to send or not. If not, they go to sleep. In order to allow concurrent operation of neighboring star networks, different channels are assigned to them by a graph-coloring algorithm. Drawbacks of BitMAC are its complexity and the performance of concurrent transmissions. Furthermore, the protocol is designed for applications in dense networks with few topology changes only. A main problem of BitMAC is its need for ”bitwise” synchronization which is hard to achieve.

2.5 ScatterWeb CDMA

The operating system ScatterWeb, especially developed for the ESB nodes, supports a native CDMA protocol for wireless communication. In contrast to other protocols in this chapter, *ScatterWeb CDMA* does not save energy, because the transceiver is permanently switched on. We used this protocol for comparison with energy efficient protocols.

Some mechanisms like a RTS/CTS exchange for collision avoidance are completely missing. Therefore the well-known hidden node problem can occur in this protocol.

Figure 2.8 shows the basic mechanism of this CDMA protocol. In case of a node has data to send, it listens for a 1-32 byte long random backoff time to the medium. If the medium is still free after this period, it starts transmitting the data. All neighbor nodes listen to the data transmission. The receiver sends a ACK packet to confirm a successfully received packet.

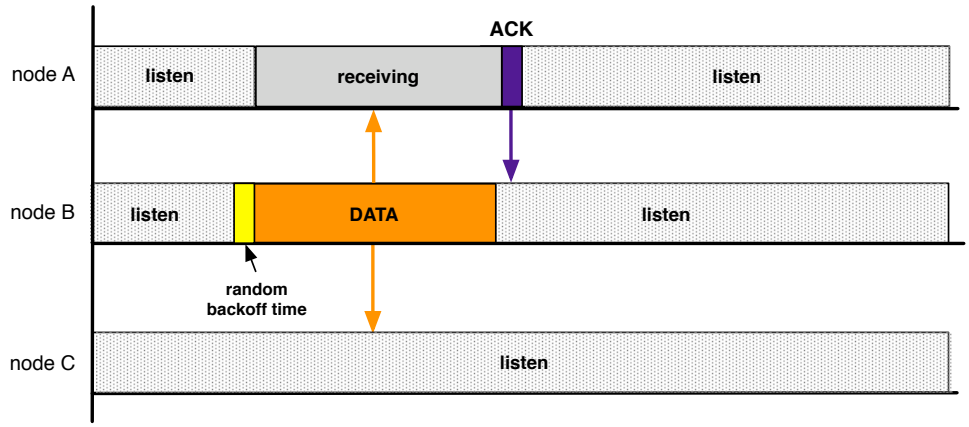


Figure 2.8: ScatterWeb data transmission.

2.6 LMAC

The lightweight medium access (LMAC) protocol for wireless sensor networks is TDMA based. It divides a frame into 32 equal slots as shown in Figure 2.9. Each slot can be allocated by only one node. Furthermore, a node can own only one slot in a frame.

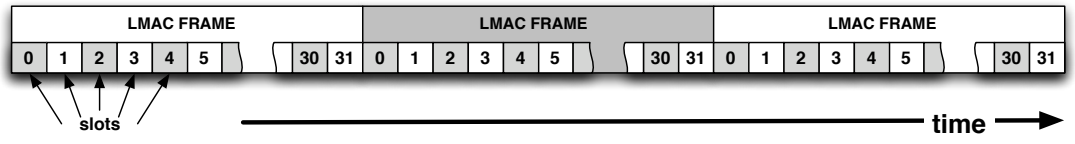
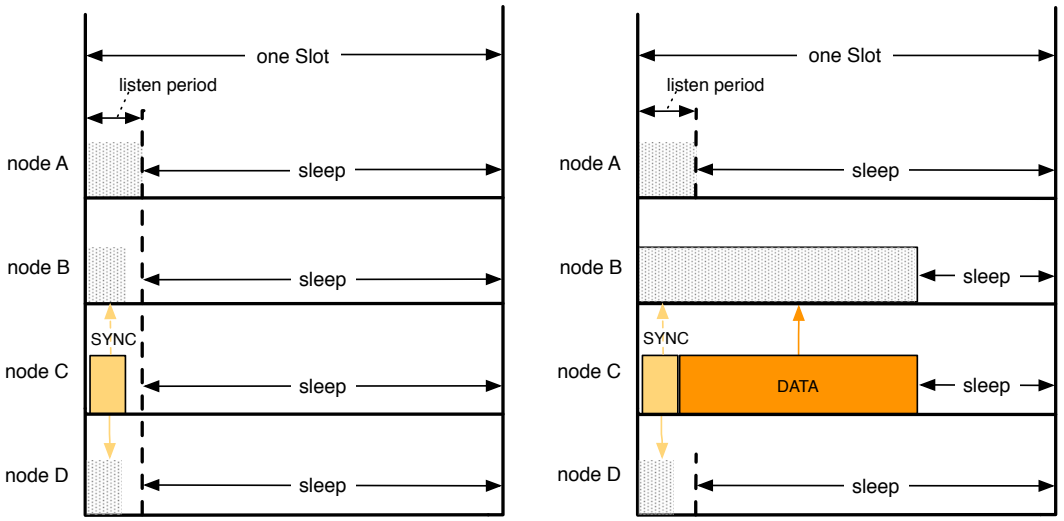


Figure 2.9: A LMAC frame consist of 32 single time slots.

The slot owner has exclusive transmission rights and can communicate collision-free. Figure 2.10 shows two different scenarios. In case of Figure 2.10(a) a slot owner has no data to send, it sends at the beginning of the slot a 12 byte control message (SYNC) and turns off the radio to save energy. In the other case a node has to transmit data, as shown in Figure 2.10(b), the SYNC and data part is sent as one packet. This makes it unnecessary to send preamble bytes for the data part. Other nodes that receive this transmission shutdown their radios after the SYNC part and do not listen to the data slice. The maximum data load is 256 Bytes to avoid an overlapping in next slot. An ACK packet for retransmitting corrupt or lost packets is not provided.



(a) Node C is slot owner has no data to send.

(b) Node B is slot owner and has to send data to node B.

Figure 2.10: Basic LMAC mechanisms.

All sensors nodes are awake at each slot for listening to a possible SNYC. Incoming control messages, shown in Table 2.1, are used for synchronizing and inform about a possible data transmission, shortest route to the gateway or detected collision. The control message is needed to find a free slot number at node initialization. The 32 bit ”occupied slots” contains all allocated slots numbers in one-hop neighborhood of the sender. This enables a new network member to know all free slot number in a two hop neighborhood after collecting all sent control messages in a frame. Slot numbers of three-hop neighbors can be reused without interfering with hidden nodes. Figure 2.11 shows a possible multi hop LMAC network. The new node can select the same slot number like the three-hop neighbor node in the right. All slot numbers allocated by the nodes in the two-hop neighborhood can not be selected by the new node. This setup algorithm do not need a central manager to divide the time slot among the sensor nodes. An one time selected number will be conserved until the power is down or an collision occurred. These collisions can occur, if two nodes make their setup at the same time and pick by accident the same slot number, or one node is moving inside the transmission of an other node with the same slot number. In

Description	Size (bytes)
Sender identification	2
Destination identification	2
Current slot number	1
Distance to gateway	1
Occupied slots	4
Distance to gateway	1
Collision in slot	1
Data part size (bytes)	1
Total	12

Table 2.1: Contents of the LMAC control message.

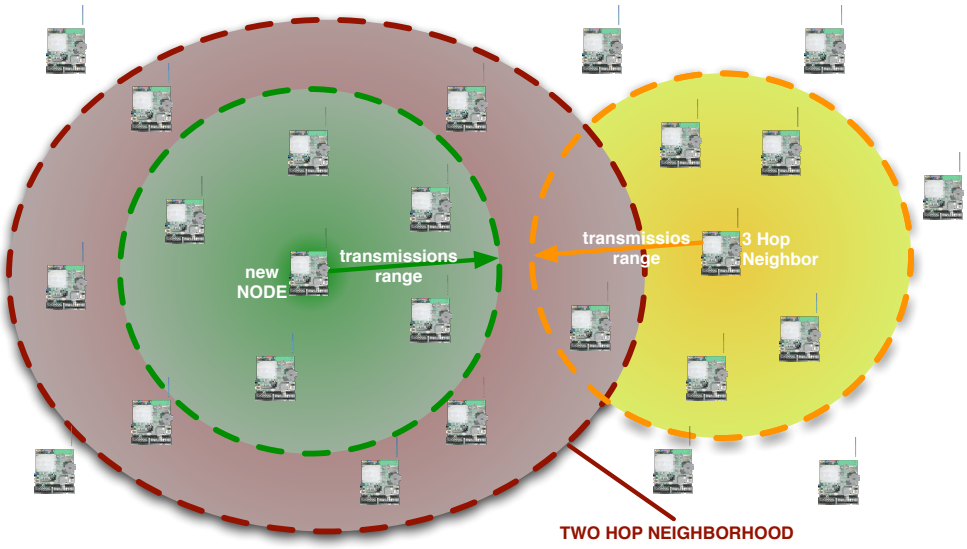


Figure 2.11: A possible LMAC sensor network. Nodes in a three-hop neighborhood can reuse the same slot number.

this case the sensor nodes become informed by their neighbors that a collision has occurred and they try to choose again a free slot number. The new slot number is chosen randomly out of the free slots, to minimize a possible collision if several nodes make there setup at the same time.

More than 32 slots each frame can be realized by adapting the control message also. Also more than 256 bytes during each frame are possible.

2.7 TRAMA

The traffic-adaptive medium access protocol (TRAMA) [7] is a collision-free MAC protocol that uses a distributed election scheme to determine particular time slots. TRAMA distinguishes between random access slots that are contention-based and used for signaling, and scheduled-access periods that are used for collision-free data exchange. Figure 2.12 shows the three components of TRAMA: The *Neighbor Protocol* (NP), the *Schedule Exchange Protocol* (SEP), and the *Adaptive Election Algorithm* (AEA). NP propagates one-hop neighbor information using signaling slots, thus information about the local two-hop neighborhood can be established. With the SEP protocol traffic-based schedule information is maintained among neighbors. SEP packets are exchanged during the scheduled-access periods. AEA selects transmitters and receivers according to the information obtained from NP and SEP. TRAMA supports multicast communication by using a bitmask vector containing the one-hop neighborhood. TRAMA has high delays compared to S-MAC and is therefore only suitable for applications that are not delay sensitive. However, the delivery ratio is much better than that of S-MAC.

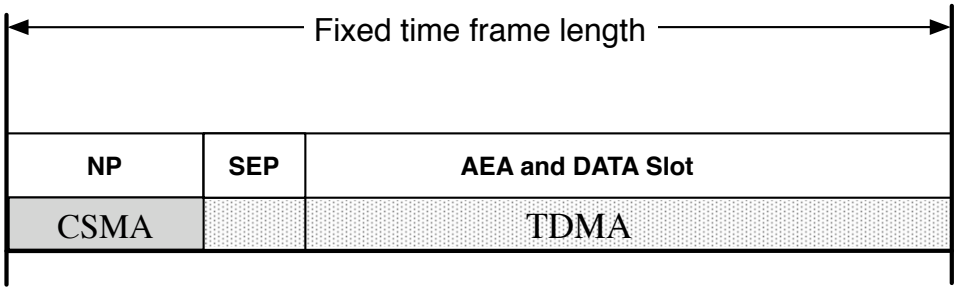


Figure 2.12: Different TRAMA periods.

2.8 B-MAC

B-MAC [8] is a carrier sense MAC protocol that employs an adaptive preamble sampling scheme to reduce duty cycle and minimize idle listening. B-MAC is a pure lightweight MAC protocol. It does not offer additional services like node organization, synchronization, and routing. It uses a Clear Channel Assessment (CCA) to determine whether the channel is clear. B-MAC duty cycles the radio through periodic channel sampling that is called Low Power Listening (LPL). The preamble length is matched to the interval that the channel is checked for activity. The

B-MAC protocol supports only minimal services and can be combined with other mechanisms to deliver tailored solutions for specific applications.

Chapter 3

Implementation of LMAC and TEEM

Section 3.1 presents the sensor hardware used with a main focus on micro-controller and transceiver. The remaining sections of this chapter describe the different protocol implementations and their problems.

3.1 Hardware

Figure 3.1 shows an Embedded Sensor Board (ESB) which provides several communication interfaces and some sensors for monitoring the environment. These nodes were developed by ScatterWeb GmbH [3], founded as spin-off company of the Free University of Berlin.

ESB nodes use the micro-controller MSP430F149 from Texas Instruments [20, 21]. A serial RS-232 standard interface allow the node to connect to a personal computer. In this thesis the serial interface is connected to a desktop computer with Gentoo Linux [23] as operating system. Minicom [22] is used as a terminal program. Default parameters for a connection are 115.2 kbit/s 8N1. The parallel interface of the ESB provides a JTAG interface (IEEE 1149.1) which allows flashing and real-time program debugging. Additionally, monitoring of variables and registers is possible. An essential component for this thesis is the wireless radio module. It is a low-power, short-range radio transceiver TR-1001 [24] using the license-free 868 MHz band. A detailed presentation follows in subsection 3.1.2.

As the name already implies, a sensor network needs sensors to collect data from their environment. The ESB node features the following sensors:

- A passive infrared (PIR) sensor allows the monitoring of movement in a range of approximately 8 m. It supports analog or digital PIR measurements.
- An infrared sensor measures the light intensity and can differentiate between natural and artificial light at 50 or 60 Hz.
- A temperature sensor with integrated real-time clock allows measurements of temperature.
- A tilt/Vibration sensor monitors tilt or vibrations.
- A microphone enables monitoring of noise level with adjustable thresholds. Unfortunately, the microphone seems to affect the measurements results of other sensors.

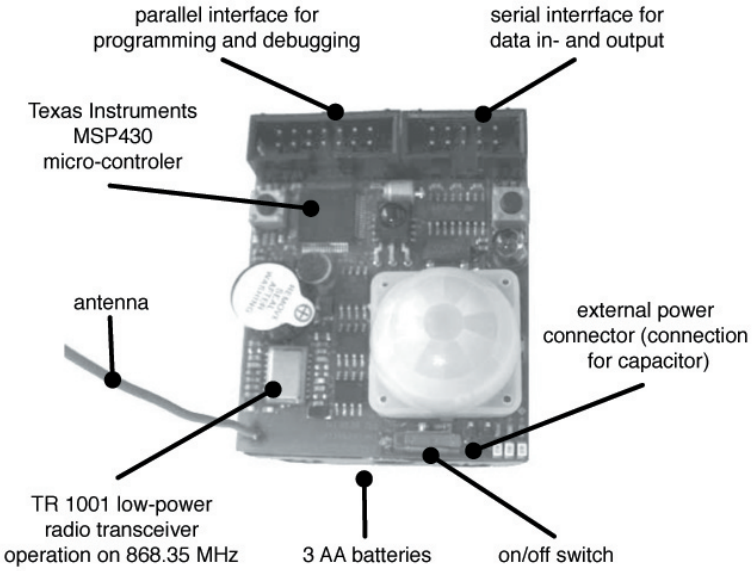


Figure 3.1: Embedded Sensor Board (ESB).

There is a battery case for 3 AAA batteries on the bottom of the ESB. The voltage controller MIC5201-3V of the ESB stabilizes the input voltage to 3 V. It is also possible to connect another power supplier like a solar panel or a capacitor. When using a capacitor without batteries, the power switch for the battery case must be powered on. Otherwise, the voltage is too low for unknown reasons. A beeper as well as red, green, and yellow LEDs and a switch can be used for application debugging or testing.

3.1.1 Micro-Controller MSP430F149

The micro-controller MSP430F149 from Texas Instruments has a low power consumption and can be switched into different low-power modes. It is possible to wake it up from a low-power mode via interrupts. The MSP430F149 has a 16-bit reduced instruction set computer (RISC) architecture. It consists of a 16 bit arithmetic unit, 16 registers, each 16 bit wide and a control unit. Four registers are reserved for use as *program counter*, *stack pointer*, *program status register* and *constant generator*. The RISC instruction set comes with only 27 basic instructions. The 16-bit address space allows the MSP430 micro-controller to address 60 kbyte + 256 byte flash memory and 2 kbyte RAM. A low supply-voltage range, between 1.8 V and 3.6 V, is adequate for the MSP430. A basic clock module generates three independent cycles for the different controller-functions:

1. The main clock system (MCLK) used for pulsing the central processing unit (CPU).
2. The sub system master clock (SMCLK) for pulsing the peripheral devices.
3. The auxiliary clock (ACLK) for pulsing the peripheral devices.

An integrated digitally controlled oscillator (DCO), generates the cycles used by the basic clock module. It is a ring oscillator which excites electrical oscillations of up to 5 MHz. It can be switched off by the *SCG0-bit*. Because the frequency depends on working voltage, it is necessary to synchronize the DCO periodical with a highly accurate quartz oscillator. It is a low-frequency oscillator at 32,768 kHz which monitors the DCO and adjusts it if required. This quartz oscillator can be switched off by the *OscOFF-bit*. The central processing unit (CPU) can be deactivated by the *CPUOFF-bit* and the MCLK can be turned off by the *SCG1-bit*. The controller can be switched in the different low-power modes shown in Table 3.1 by switching off the CPU and oscillators A useful feature is the watchdog timer (WDT) module. It performs

operating state	SCG1	SCG0	OscOFF	CPUOFF	power input
active mode (AM)	0	0	0	0	340 μA
LPM0	0	0	0	1	55 μA
LPM1	0	1	0	1	55
LPM2	1	0	0	1	17 μA
LPM3	1	1	0	1	1.6 μA
LPM4	1	1	1	1	0.1 μA

Table 3.1: Different MSP430 low-power modes (LPM) at 3 V.

a controlled system restart after applications crash or are locked in a loop. If the chosen time interval expires, a system reset is generated.

Interrupt vectors are located in the read only memory (ROM) unit with an address range of 0xFFFFh to 0xFFE0h. The vector contains the 16-bit address of the appropriate interrupt handler instruction sequence. The highest priority, 15, is occupied by power-up, external reset and watchdog. Figure 3.2 explains the interrupt architecture of the MSP430 controller. The MSP430 JTAG interface is used to program the flash memory and to debug applications. JTAG is an acronym for Joint Test Action Group, which is the common name used for the IEEE 1149.1 standard.

In order to communicate with the radio transceiver, one of the two universal synchronous-asynchronous receiver/transmitter (USART) of the MSP430 micro-controller is used. The other USART is used for the RS232 interface.

3.1.2 Transceiver TR 1001

For wireless communication with other nodes, ESBs use a RFM TR-1001 [24] radio transceiver. The TR-1001 hybrid transceiver is designed for short-range wireless communication with low power consumption. The module operates in the license free ISM band of 868.35 MHz and has a maximal transmitter power of 1 mW.

Because the on-off keyed (OOK) operation is used at 838,35MHz, which simply turns the HF signal on and off, the transmission of more than 20 - 30 equal bits must be avoided, i.e., if more than 20 - 30 zero bits are sent in a row, the receiver oscillator drifts to a different bit pulsing and is no longer able to identify incoming bits correctly.

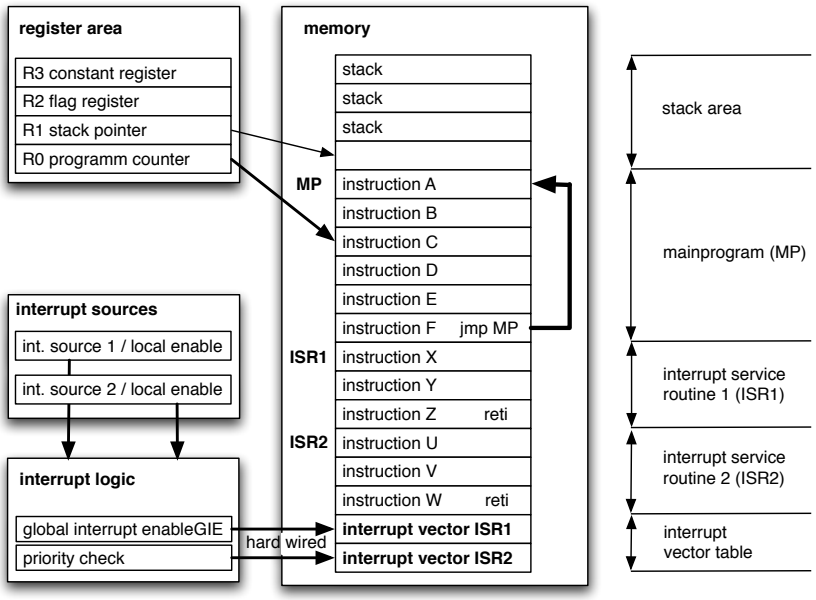


Figure 3.2: MSP430F149 interrupt architecture.

Transmission range depends heavily on the environment. An ideal outer field allow a transmission range up to 300 m. Walls inside a building or wireless devices like a GSM mobile phone, decrease transmission range considerably. During the protocol implementation it has been shown that nodes should be placed at least 1 m from the floor or walls, similar to recommendations of ScatterWeb [3]. This distance is needed by the electromagnetic field built by the dipole antenna with the fourth length of the wavelength to minimize interferences.

Also some home appliances or a nearby railway depots can decrease the transmission range. Table 3.2 shows the three different operation modes and their correspondent power consumption of transceiver TR-1001. There are three things to attend for radio transmission. First, the transmitter should send a preamble consisting of up to five 0xAA or 0x55 bytes. This should enable the receiver to synchronize itself. However, several experiments have demonstrated no difference when using this preamble. Second, the USART needs a 0xFF Byte to find the start of new byte sequences. Normally, the next two bytes are a frame preamble to identify a packet start. Lastly, something like manchester encoding should be used in order to avoid long sequences of 1 or 0 bits.

Transceiver state	power consumption
Transmitter sending (peak)	12mA
Receiver listening	4.5 mA
Transceiver sleeping	5μA

Table 3.2: TR-1001 transceiver energy consumption, powered with 3 V, at 19.2 kpbs.

3.2 Programming

The Free University of Berlin provides an operating system named ScatterWeb for the ESB nodes. It is written in C with MSP430 specific extensions. A SourceForge project page, named mspgcc [25], provides free compilation tools for the MSP430. It includes the GNU C compiler (GCC) with necessary language extensions, an assembler and linker (binutils), a debugger (GDB), and some other useful tools. The msp430-gdbproxy connects the micro-controller through the JTAG parallel port interface to the GDB. Unfortunately this program is available only as a binary file and requires the parallel port path *"/dev/parport0" (mknod /dev/parport0 c 99 0)*. On some personal computers it was not possible to arrange a successful connection over the JTAG interface. Table 3.3 shows the software used for development.

Software	Description	Version
Linux distribution Gentoo	Operation system	2.6.14-gentoo-r5
GCC	GNU C compiler	3.2.3
binutils	Assembler and linker	2.14
msp430-libc	Libc for MSP430	CVS, November 2005
libHIL	JTAG hardware access library	CVS, November 2005
msp430-gdbproxy MSP430	msp430-libc	CVS, November 2005
ScatterWeb	ESB operation system	2.3

Table 3.3: Software used for development.

The *ScatterWeb* software consists of an application and a system part. There are several applications with different functions available. The system part contains the main components for an operating system. Figure 3.3 shows the system parts most important to this thesis. System.c contains the main loop, which starts the watchdog. As soon as the program crashes or is locked in a loop, the watchdog triggers a reset. Next, a 16 bit *runModule* variable is checked for open tasks. Every bit in this variable represents an event handler, like the network receive, network transmit, data sensor or timer event handler. After all open handlers are called and finished, the watchdog is stopped and the micro controller goes into low-power mode 1 (LPM1), until the next interrupt starts the main loop again.

Net.c contains the network receive and network transmit handlers mentioned above. The network receive handler is called after a packet has been completely received, and sends the packet to the application if necessary. The network transmit handler is called after a packet has been completely sent by the transceiver. It deletes the sent packet from buffer or retransmit it if an error has occurred.

The most complex part of Net.c are the two interrupt functions. *radio_tx_isr()* is called every time the radio transceiver has sent one byte. *radio_rx_isr()* is called after the radio transceiver has received one byte completely. After the packet is completely sent or received, the interrupt function sets the corresponding bit in *runModule* and wakes up the main loop to activate the handler.

Because the transmitter used can either listen or send, it has to switch between the two states.

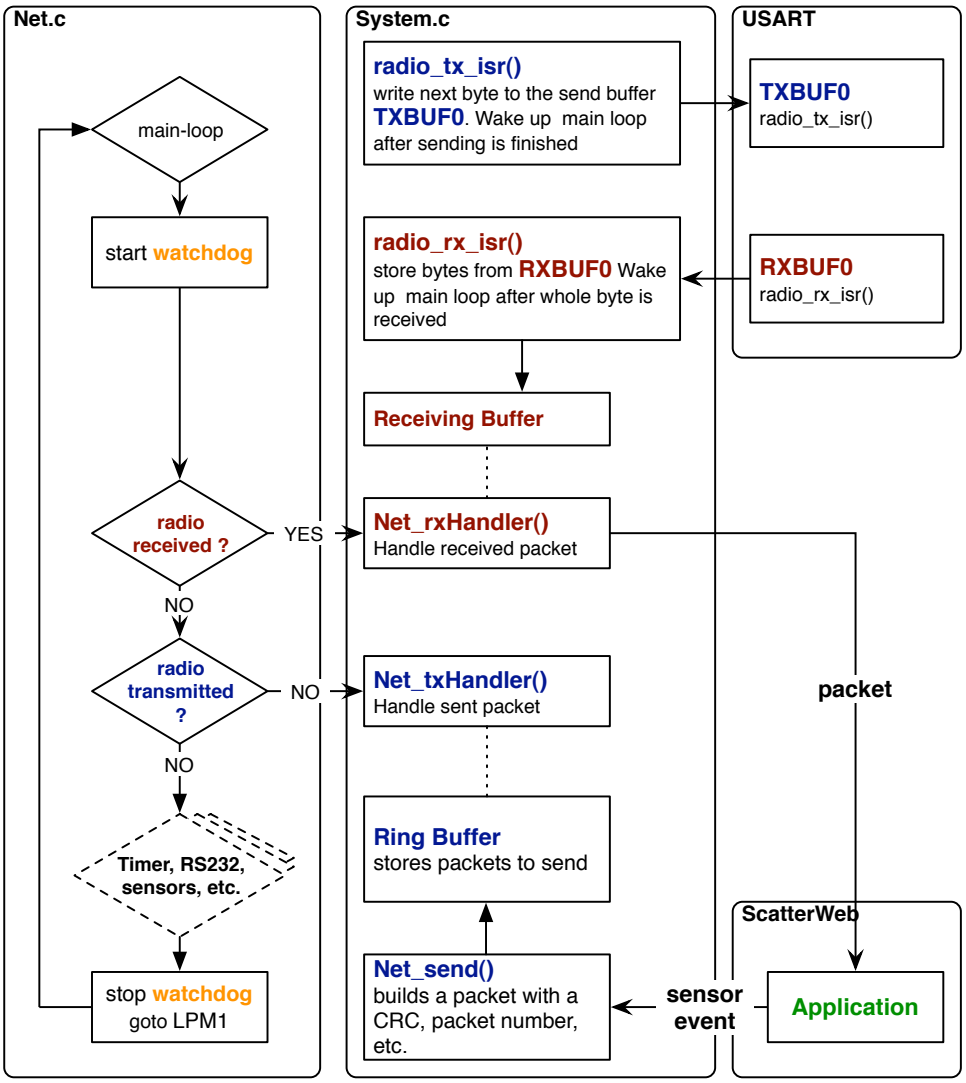


Figure 3.3: For wireless communication relevant parts of ScatterWeb operation system.

Especially, while the transmitter switches from the receive to the transmit state, it is not possible to listen to the medium. If during the switching time another node starts sending a packet, the occurring collision is not detectable by these two nodes. Usually, in contention based energy saving protocols, all nodes try to send a synchronization packet in a very short time slot. For energy efficient protocols it is very important to feature a short switch time during which the nodes can not listen to the medium. The TR-1001 data-sheet specifies a receive to transmit switch time of about 12 μS. Essentially, the switch-time is more than 300 times longer, mainly because of the necessary preamble bytes, as seen in Figure 3.4.

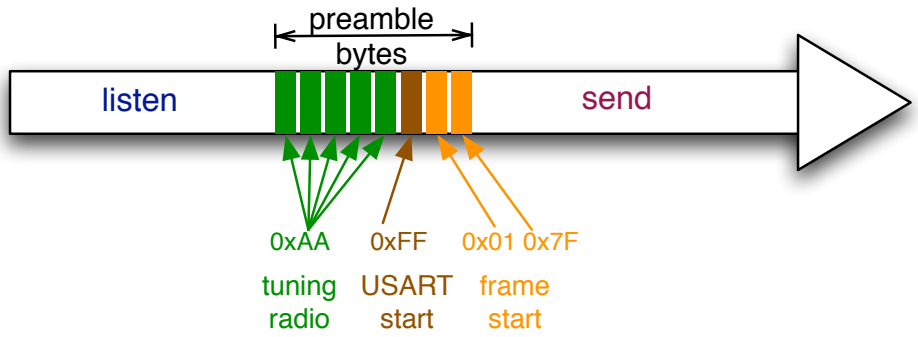


Figure 3.4: Preambles send by transmitter for tuning the receiver and frame start detection.

The Free University of Berlin suggests to use up to five 0xAA bytes. After that, the USART needs an incoming 0xFF byte to recognize a starting data transmission. Only at this point the other nodes are able to detect the next byte and stop a possible sending task. At 19.2 kbps, every byte needs 417 μS which results in 3.3 ms for the total 8 preamble bytes. Our own experiments further indicate that one 0xAA preamble byte is enough for transmitter tuning. Using one or five 0xAA bytes did not influence the error count of bit switching. Other interferences from known devices such as mobile phones result in the same error rate for both preamble lengths. Unfortunately, these experiments were rudimental because of the lack of professional measuring instruments.

Another important condition for implementing energy saving protocols is a very accurate timer support. Scatterweb provides sufficiently accurate timer functions. However, the reliability of incoming synchronization packets is insufficient. An experiment demonstrates this problem. Node A, controlled by existing and accurate timer functions, sends a packet to node B every 1000 ms. Node A sends without any random backoff time or medium checking. If the nodes are near to each other, node B detects an incoming packet every 1000 ms. As soon as the nodes are far away from each other, a divergence of $\pm$ 5 ms is immediately measurable, i.e. interarrival time 995 - 1000 ms. This leads to longer listen times in protocol implementation and consequently to higher energy use.

3.3 LMAC

Two versions of LMAC have been implemented. Version 1 (*LMAC V1*) is done as close as possible to the LMAC paper [1] described in section 2.6. Interferences, bit errors and a working synchronization decrease energy saving effects substantially. Small, but unrealistic networks in a five meter range work smoothly with highly energy efficient timing parameters. But it was not possible to setup a network with lager distances between nodes using *LMAC V1*. To compensate this problems caused by interferences LMAC Version 2 (*LMAC V2*) has an additional CRC. Figure 3.5 shows the procedure of one node in one time slot.

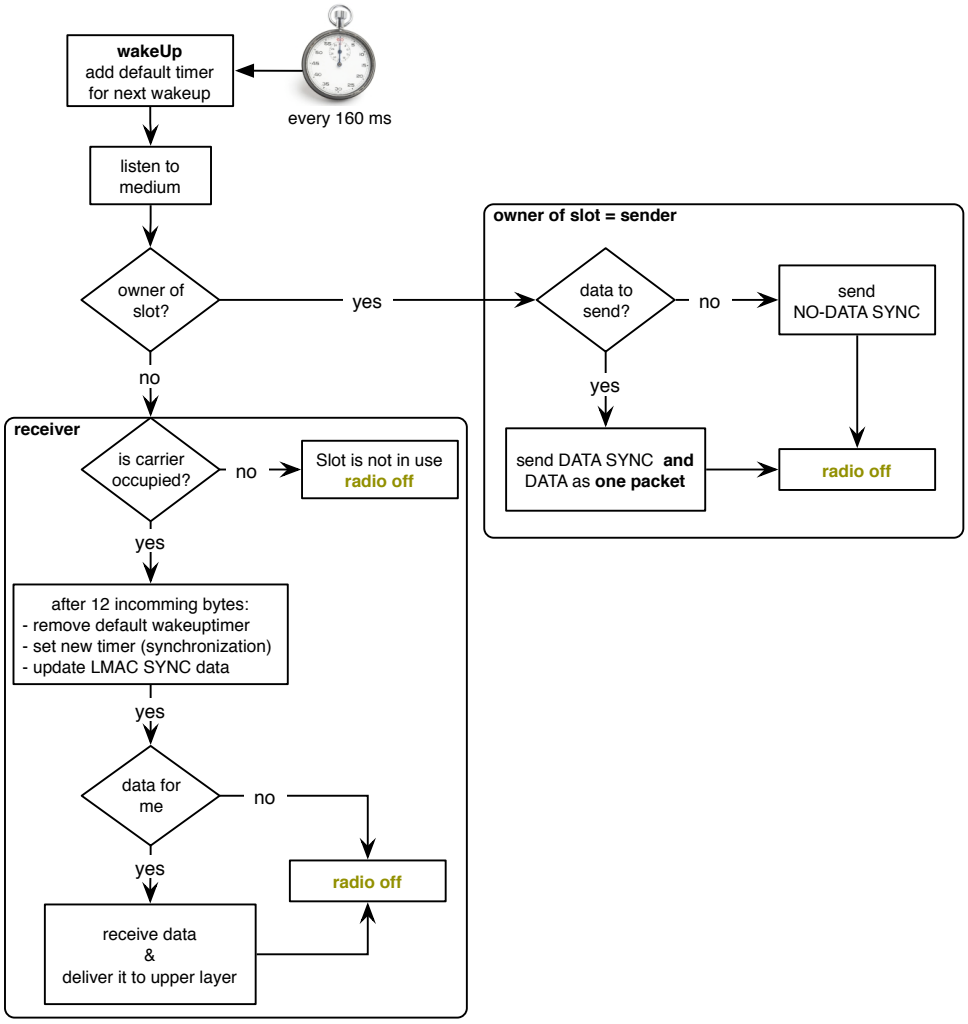


Figure 3.5: LMAC program logic summary.

3.3.1 LMAC V1

The authors of [1] provide no information about slot times or transceiver frequency. Depending on our hardware and a transceiver speed of 19.2 kbps each slot needs 160 ms to place the necessary parts (listen segment, preambles, control message, and data part). Thus one frame with 32 time slots takes 5.12 sec.

All nodes wake up at quite the same time, activate their transceivers in listening mode and set a default timer (160 ms) for the next wake up. To ensure that all nodes are listing, the current slot owner waits between 13 and 15 ms until it switches the radio to sending mode as shown in Figure 3.6.

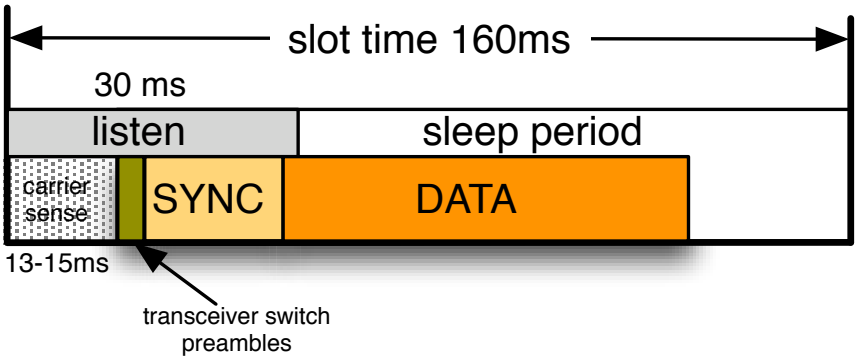


Figure 3.6: Different timers in LMAC implementations.

For synchronization, all visible neighbors remove their default timer directly after receiving the last control message byte and replace it with a new timer for wake up. Even the receiver of a following data part is fast enough to do that. The slot owner and the data receiver turn their radio off after the data transmission finishes. Visible neighbors turn the radio off directly after the last control message byte. Nodes, which are hidden from the sender, turn their radio off 30 ms after wake up.

3.3.2 LMAC V2

In the *LMAC V2* several things are adapted. Primarily a 16 bit CRC is added to the control message for getting a more stable wide range network on the ESB platform. Some parts of the control message are compressed to add new information without increasing the control message size. Figure 3.7 shows the structure of the implemented control messages.

The following subsections explain the main LMAC problems and give a detailed characterization and explanation of the motivation following changes in *LMAC V2*.

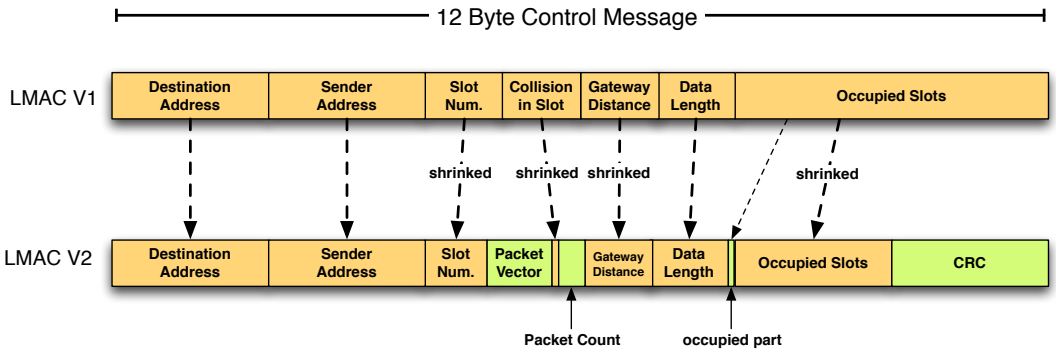


Figure 3.7: Control messages of *LMAC V1* and *LMAC V2*.

3.3.3 Data Rates

One frame with 32 time slots needs 5.12 sec for one cycle. Therefore, a node can send a maximum of one packet each 5.12 s, something which is not feasible. Nodes close to a sink which forward a lot of data packets would drop most packets. To solve this problem, in both LMAC versions a node can send up to 8 packets in a row in one time slot. A disadvantage of this feature is that every data packet contains its own checksum and length declaration as is shown in Figure 3.8. Thus, if one checksum is corrupt, the length of following packets can not be calculated and must be dropped. To reduce this problem, the control message of *LMAC V2* adds a length vector to the packet in the middle of data part.

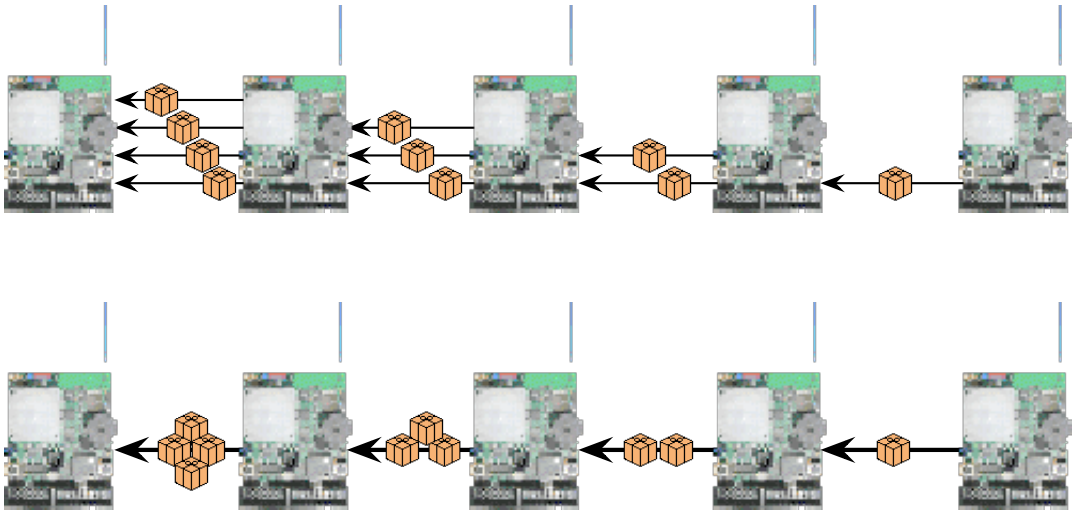


Figure 3.8: LMAC traffic improvement. Up to 8 packets can be sent in one slot.

3.3.4 Bit Errors

If nodes are placed far away from each other, interferences reduce the signal-to-noise ratio to a critical level. Signal-to-noise ratio, as shown in Equation 3.1, is a term for the power ratio between a signal (meaningful information) and the background noise.

$$SNR = \frac{Power_{signal}}{Power_{noise}} = \left(\frac{Amplitude_{signal}}{Amplitude_{noise}}\right)^2 \tag{3.1}$$

Originally LMAC does not come with a checksum for ensuring that the control message is correctly received. Bit errors lead to wrong slot-numbers, or wrong destination or sender addresses while evaluating received information. Corrupt control messages appear to have a bigger time shift than correctly received bytes. To clarify this issue, the control message of *LMAC V2* has a 16 Bit CRC field for checking information integrity.

3.3.5 Synchronization

The authors of [1] assumed that the clock drift is negligible in a single frame. But the already mentioned time shift of a control messages is quite bigger than a the time shift of the micro-controller basic clock module.

Nodes with different wake up times do not receive each others SYNC messages. In practice the outcome of this is a clustering effect. Figure 3.9 shows five nodes in a row which have built two clusters. Communication between the two clusters is no longer possible. If each cluster consists of a minimum of two nodes, the clusters can synchronize themselves and keep disconnected. To avoid a clustering, the listen window at wake up must be long enough to ensure that all nodes can hear each other. However, long listen windows increase energy consumption considerably. Thus, an effective synchronization is important for energy efficiency.

A *schedule table* like SMAC is using for communication between virtual clusters is unusable. The whole time in a slot is reserved for the 256 byte packet load. An overlapping with an other neighbor slot result in a collision.

In our implementation *LMAC V1* and *LMAC V2* start scanning for neighbors if a node does not receive a SYNC message for more than 5 frames. To do this, a node makes a new network setup and scans for the last known gateway node during three frames. If this gateway can not be detected, it searches for another node. Searching first for the gateway node reduces network sensitivity for clustering.

The most effective adaptation is to drop every control message with a wrong CRC for synchronization, because the time stamps of corrupted packets mostly have larger shifts than correctly received packets. Only LMAC V2 with this CRC feature can avoid a network clustering.

Nodes that do not receive a SYNC message have to listen for 30 ms to the medium before they can turn off the radio. Theoretically a node can turn off the transmitter earlier because the control message is already overdue since several milliseconds. However, shorter listen intervals lead to faster clustering.

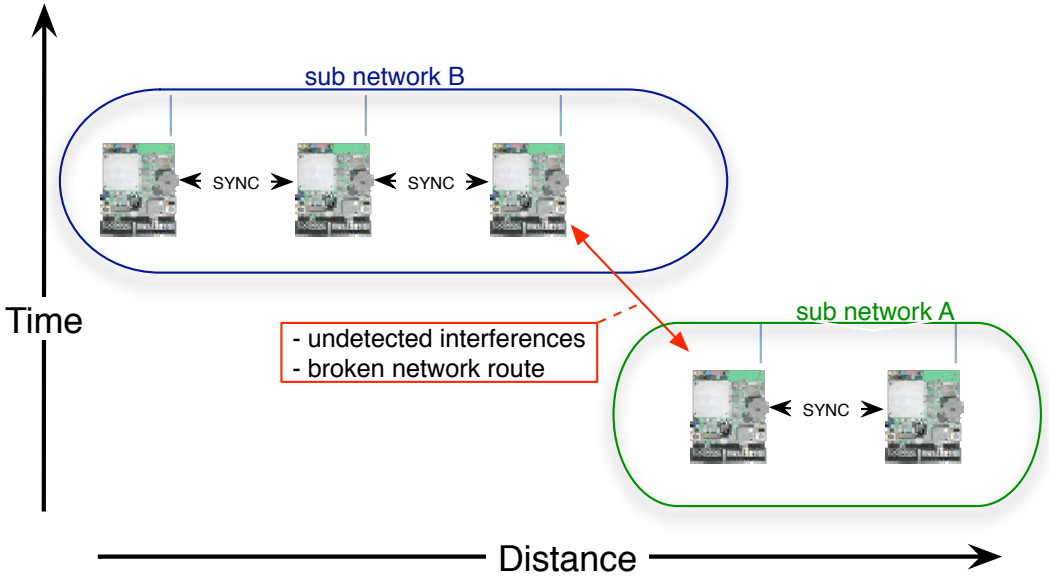


Figure 3.9: LMAC nodes with different timed slot starts. The sub networks can not interconnect for synchronization or data transmission.

3.3.6 Pre- And Postamble

Figure 3.10 shows additional pre- and post-amble bytes for LMAC used all in our experiments. Each byte represents one interrupt call. Every time a byte from the send-buffer completely has been sent, the USART interface calls the programmable *radio_tx_isr()* function. To ensure that *radio_tx_isr()* is invoked constantly, it is required to write a byte in the send-buffer each time. This is also necessary if the radio is in listening or sleeping mode. During this time, the radio sends nothing and does not waste energy. The first byte (0x00) in Figure 3.10 shows one of these bytes invoking constantly this function also in listening mode. For the next two cycles the radio is switched to sending mode. Then, four preamble bytes are sent. The first byte (0xAA) is for tuning the radio oscillator, the second (0xFF) is to allow the USART to detect the sending start. The next byte is the first byte which makes it possible for the programmer to detect a packet start. Hence every sending try would be stopped. The fourth byte ensures the packet start. The control message and data part follow. The data part assumes the packet structure from *scatterWeb*. It includes 16 bit addresses for receiver and sender, packet type, packet number, packet length and a 16 bit CRC for a integrity check. After two post-ambles for termination, the radio is set back to listen-mode and the transmit handler is called, which turns the radio off. Until next sending, *radio_tx_isr()* will be invoked constantly by writing 0x00 to the send buffer.

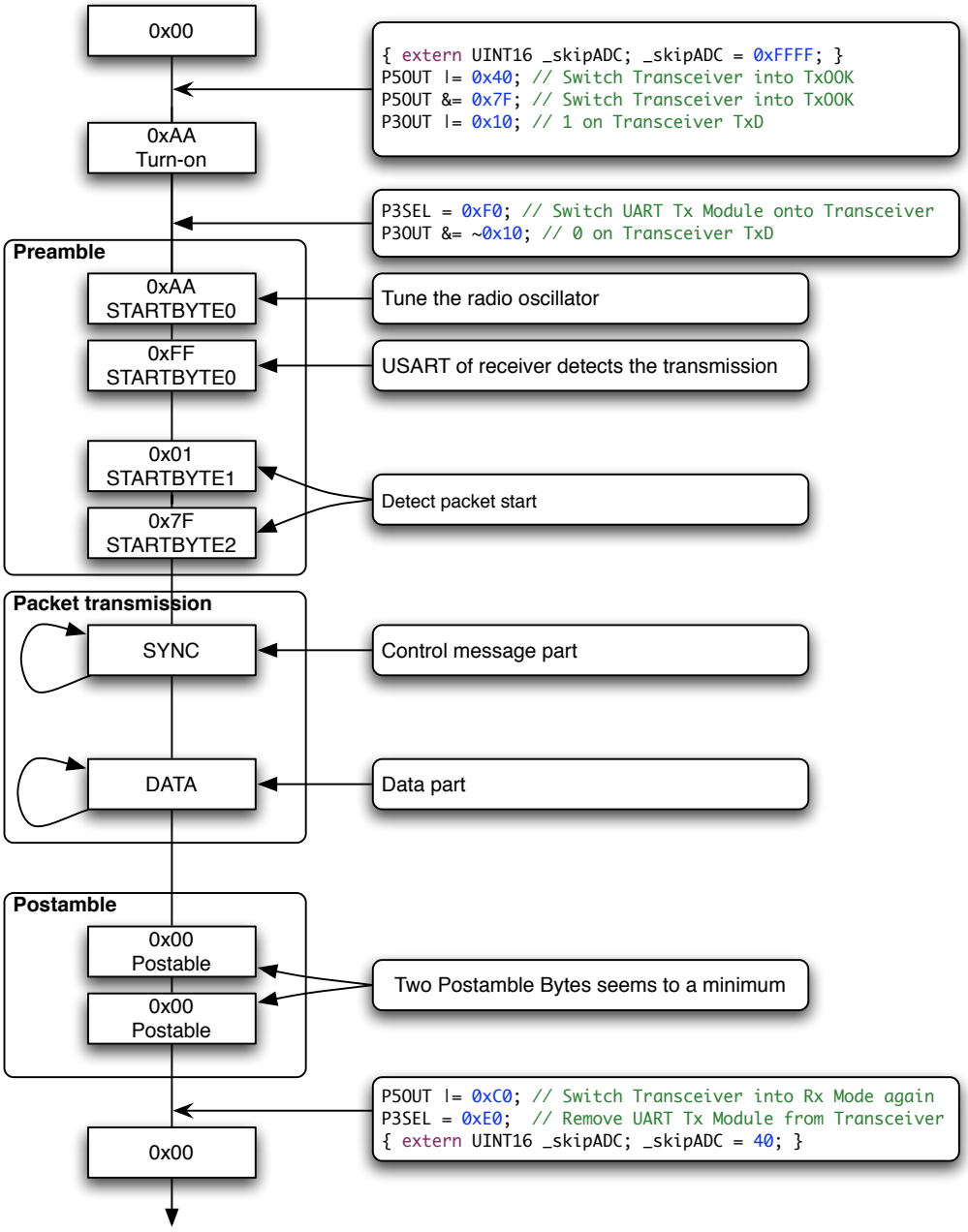


Figure 3.10: All bytes send by LMAC transmission interrupt.

3.4 TEEM

Two versions of TEEM have been implemented. Version 1 (*TEEM V1*) is kept as close as possible to the TEEM paper [2], described in section 2.2. Version 2 (*TEEM V2*) differs by the possibility to send up to 8 packets in one slot for expanding bandwidth. Both versions work stably and can deal with interferences and resulting bit errors in long range networks. Figure 3.11 shows the basic program logic at a slot start of a TEEM node on the ESB platform. Every node tries to send a SYNC message. If there is data to send a node tries to send a $SYNC_{data}$ else a $SYNC_{nodata}$. If the medium is occupied by a incoming SYNC packet the program logic switches to Figure 3.12 and evaluate the received information.

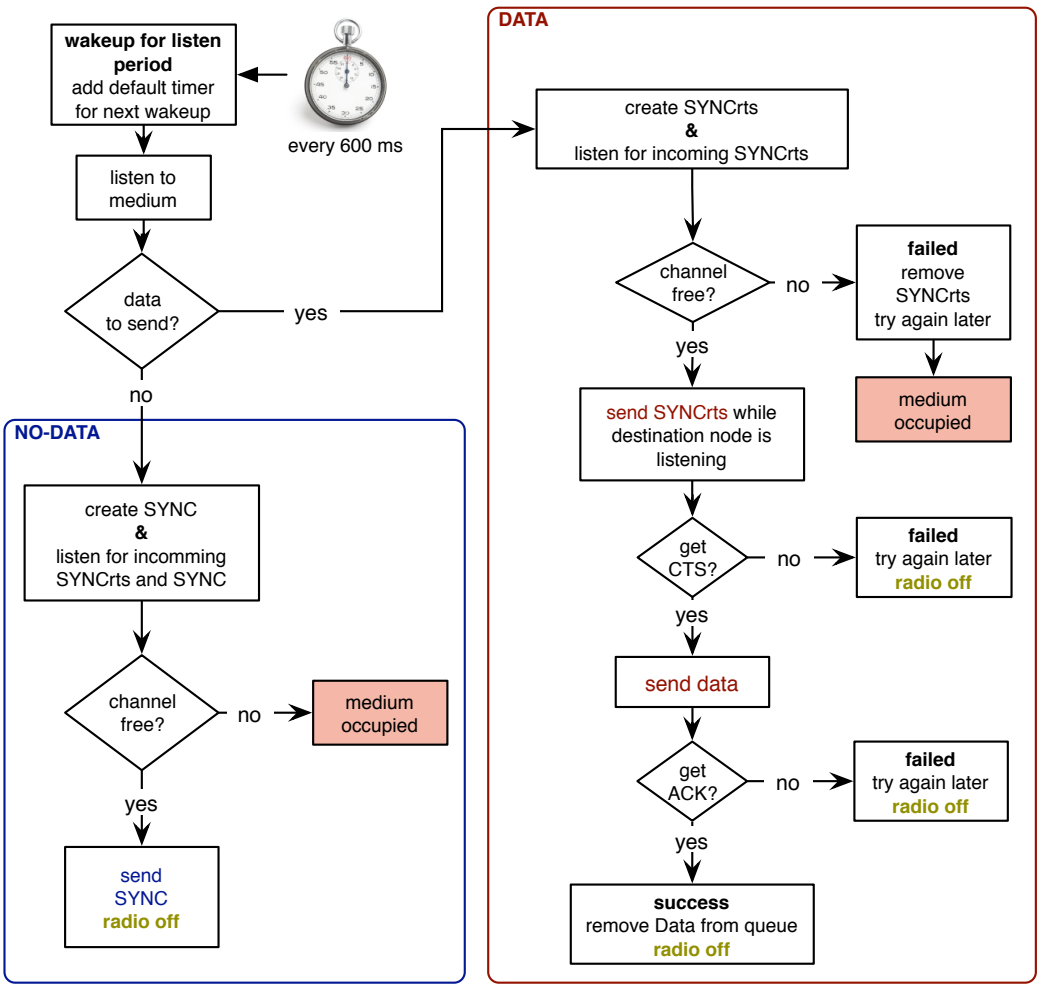


Figure 3.11: TEEM node as sender.

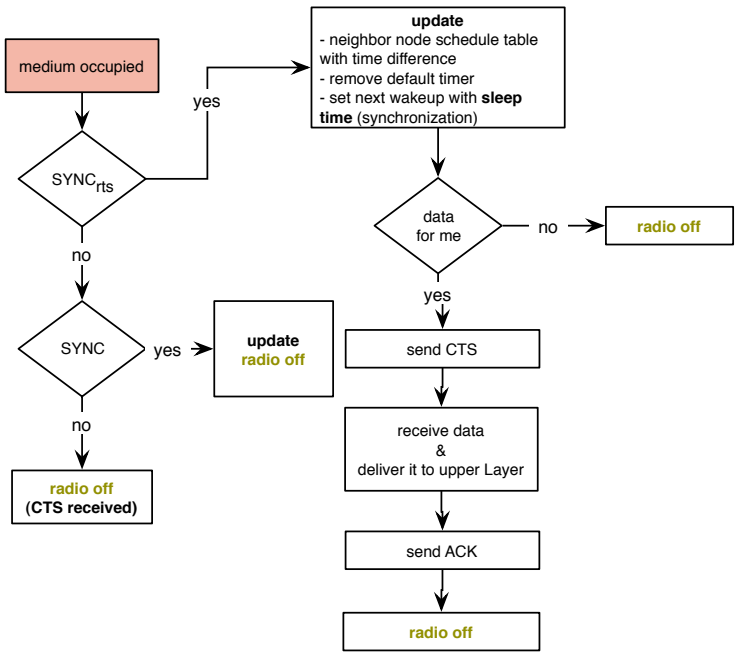


Figure 3.12: TEEM node as receiver.

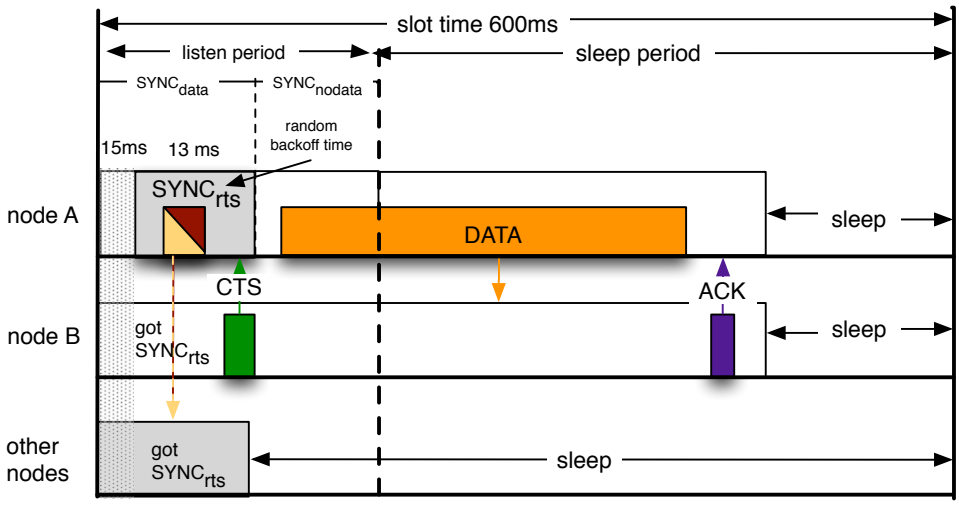
3.4.1 TEEM V1

The authors of [2] only make a statement about the listen period length of 83 ms. There are no time information about the listen period segmentation or the used transceiver frequency. Figure 3.13 shows the used timers to realize TEEM on the ESB nodes.

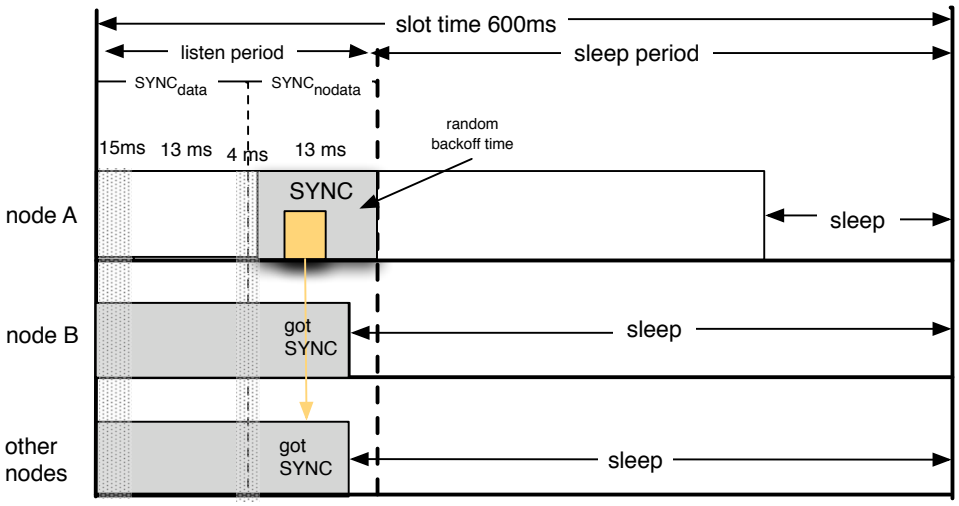
At slot start, first the default timer is set for the next wake up in 600 ms and the radio is turned on. In case a node has data to send, a 15 ms listening time interval ensures that every node is listening to the medium. After that, the random backoff time (0 - 13 ms) starts. If the medium is still free the node sends its $SYNC_{rts}$ and waits 8 ms for a CTS packet. If no CTS arrives, the nodes tries to send the packet in the next slot. Generally, a node drops a packet after five unsuccessful transmitting attempts. After the incoming CTS starts the transmission of one data packet, it waits up to 25 ms for the ACK. Then the sender turns the radio off.

A data receiver drops the 600 ms default timer after a incoming $SYNC_{rts}$. The new wake-up timer is set with the received "sleep time" information. This enables to compensate the random backoff time for ensuring that every node has a 600 ms slot. After sending the CTS packet, it waits up to 25 ms for the incoming data packet. If the CRC of this packet is correct, it sends the ACK packet and turns the radio off immediately.

If a node has no data to send, it starts the random backoff time 50 ms after wake up. This ensures that nodes with data are preferred to allocate a slot. Having sent the *SYNC* packet, the node turns instantly the radio off. Further, every neighbor node which receive a $SYNC_{rts}$/*SYNC* stops any own sending trials and replaces the wake-up timer for synchronization.



(a) Data transmission in TEEM.



(b) No data to transmit.

Figure 3.13: Different timers in TEEM implementation. A 15 ms time interval at the beginning guarantees that every node is listening during the random backoff time in the $SYNC_{data}$ period. The $SYNC_{nodata}$ period starts after 28 ms.

3.4.2 TEEM V2

TEEM V2 is able to transfer up to 8 packets in a slot to support a lager bandwidth. The $SYNC_{rts}$ is modified as shown in Figure 3.14. Length, state, sequence number and NAV are removed and replaced with four 10 bit vectors and a packet count. Length can be compensated by the type variable, because the $SYNC_{rts}$ has a fixed length. A NAV is not necessary in our implementation, because all nodes use the same listen/sleep cycle. The mechanism S-MAC applies to handle multiple clusters can not be used on the ESB platform as it results in in network overload, as described in subsection 3.4.4. Using RTS/CTS is adequate to avoid the hidden node problem. The additional vectors point to the different packet starting-points of a multiple packet transmission. If a packet has its erroneous CRC, the next packet start point can nonetheless be calculated. In case of more than four packets in one transmission, not each packet has its own start vector.

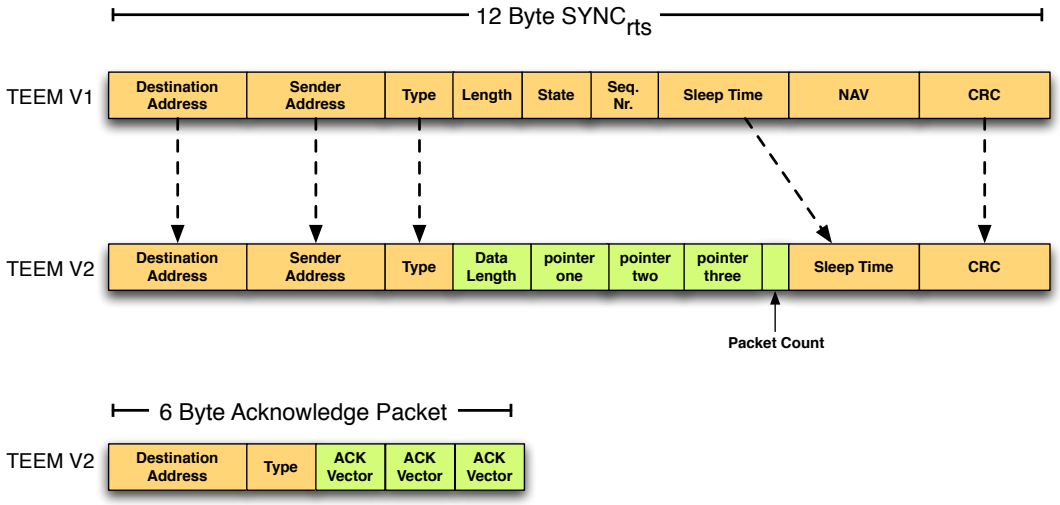


Figure 3.14: $SYNC_{rts}$ of *TEEM V1*, *TEEM V2* and ACK for *TEEM V2*

In the ACK packet, the 16 bit source-address and the 8 bit packet number are replaced three times by the same 8 bit ACK-vector. Every bit in this vector represents a sent data packet. If the CRC check for a packet failed, the bit for this packet is set to one, otherwise to zero. The data sender retransmits every unsuccessfully sent packet in the same slot. To verify the ACK, the data sender makes a voting with the three ACK-vectors. If more than two bits differs, the whole data transmission is repeated.

3.4.3 Bit Errors

As long as the network bandwidth is not overloaded almost every packet reaches the target node. Five retransmissions are only rarely unsuccessful and the packet is dropped. Further a ten-time retransmission rate does not eliminate dropping. Sometimes a long distance transmission is not

possible for several seconds. Interferences occur often in burst, like the periodic mobile phone paging of GSM providers.

3.4.4 Synchronization

In contrast to LMAC, TEEM does not tend to build clusters. During the measurement, even wide-range networks are stable and every node starts at the same time. It is essential to compensate the random backoff time. For that purpose every node transmits its own backoff time used in the $SYNC_{rts}$ respectively *SYNC* value "sleep time".

Because TEEM is inspired by S-MAC we try to realize the clustering algorithm, which allows communication between nodes with different wake up times. Several programming tries ends always in a complete network desynchronization. As long all nodes have a high signal-to-noise they stay synchronized. But as soon as the distance and interferences between the nodes grows, they start to build clusters. Attempts to synchronize their schedules were unsuccessful. Changing the scan interval for neighbor nodes, or modifying the algorithm to calculate the own listen/sleep cycle were only able to retard the desynchronization process.

Now all nodes have the same listen/sleep cycles in our implementation. Figure 3.15 shows that under these circumstances the RTS/CTS exchange is sufficient to avoid the hidden node problem and the NAV can be saved. Nodes in a two-hop distance can detect a CTS about 10 ms after the $SYNC_{rts}$ sender has started the transmission. This is quite a long time and not really practical.

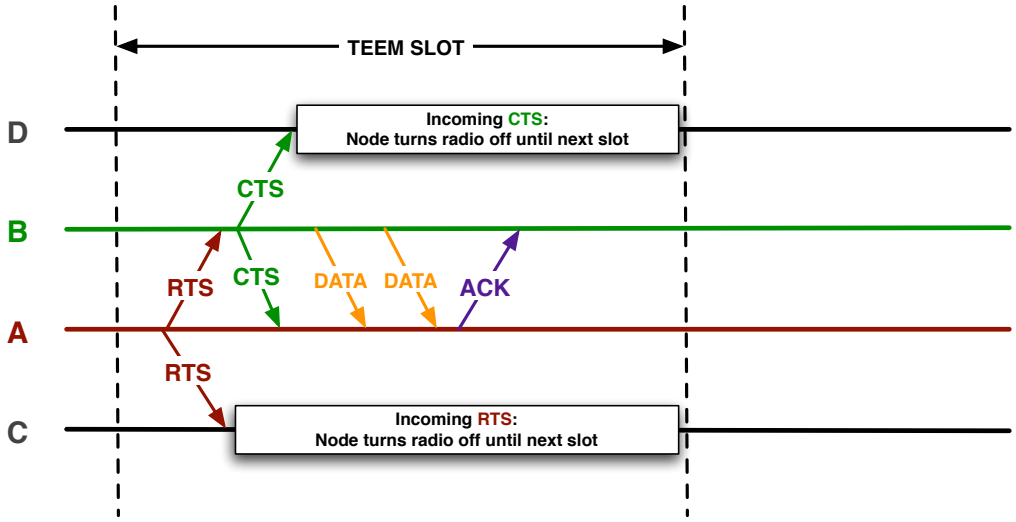


Figure 3.15: Node D is not in the transmission range off node A. But both nodes are in the transmission range of node B. Therefor all nodes receiving an CTS turn their radio off during the actual slot.

3.4.5 Post- and preamble

If a node has data to send, TEEM needs four packets for a successful transmission. Every packet needs additional pre- and postambles. The control and data part are divided into to separate packets. In combination with the additional CTS and ACK packets, TEEM needs a lot more additional bytes for a data transmission than LMAC. If there is no data to send, also TEEM needs only one packet to guarantee a synchronous network. Figure 3.16 shows all additional bytes for sending a data packet. Pre- and postambles are the same as in LMAC. After sending $SYNC_{rts}$ the second start byte for frame packet start detecting is economized.

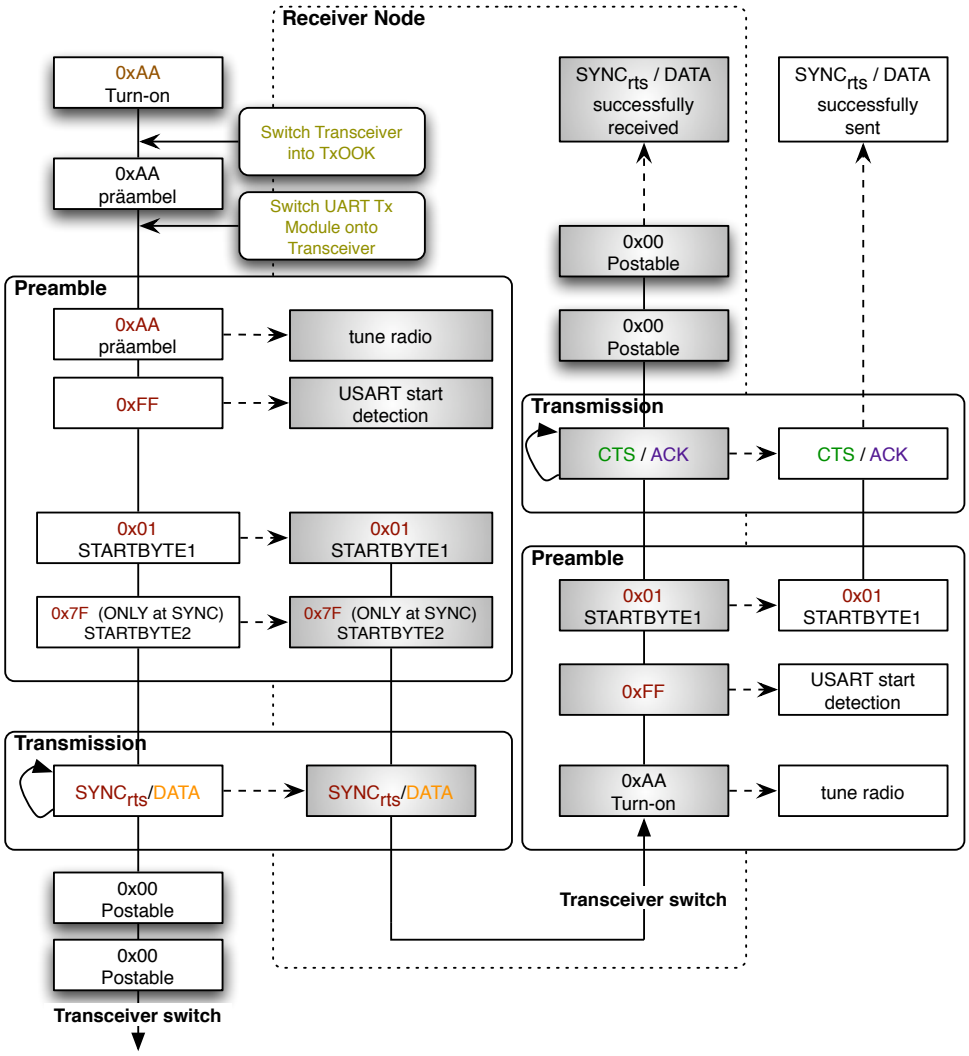


Figure 3.16: All bytes send by TEEM transmission interrupt. A RTS/CTS exchange is like shown quite equals to a DATA/ACK exchange.

Chapter 4

Experiments and Results

Subsection 4.1 explains the evaluation methodology for measuring the energy consumption of the implemented protocols. Then, subsection 4.2 specifies the experimental network setup. The results are shown in subsection 4.3.

4.1 Evaluation Methodology

The evaluation focuses on measuring energy consumption and analyzing the network behavior. Different possibilities to determine the energy usage are investigated. A simple amperemeter is unserviceable because the current conduction is alternately extremely fast. Further an amperemeter, which is able to record the energy consumption in correlation to time for calculating an integral, was not available at the institute of Physics of the University of Bern.

Therefore, we used another method for measuring the energy consumption, presented in [27]. We determine it directly by using the lifetime of a node with an exact amount of energy at the start of the experiment. Because the capacity of batteries is highly varying as well as the suffer from the battery relation effect, we use 1F GoldCap capacitors with a very high capacity of 1 Farad at 5.5 V. A capacitor can store an exactly known amount of energy in the electric field between a pair of closely spaced conductors. The capacitor's capacitance (C) depends on the charge (Q) which is stored in the conductors and the impressed voltage (U):

$$Power_{storage} = \frac{1}{2}CU^2 = \frac{Q^2}{2C} = \frac{1}{2}UQ \tag{4.1}$$

Equation 4.1 shows that our 1F GoldCap can store up to 15.1 Joule (J) at a charging voltage of 5.5 V. A customary AA battery can store about 8.3 kJ. To ensure that in our measurements the capacitor has always an almost identical load, a series of measurements analyzing the correlation of GoldCap charging time and the node lifetime has been done. Figure 4.1 shows the different lifetimes of a node with a charge time of 0.5 to 10 min. We have decided to use a five minutes charge time at 5.5 V for our lifetime measurements. There are two possibilities to define the endpoint of lifetime. An ESB can report the actual battery working voltage over the RS- 232 interface or a *Battery Warning* if reaching a specified voltage level. Unfortunately, the reported voltage alternates too fast for an accurate gauging. It is more accurate to log the breakdown of the RS-232 interface to get uniform results. At breakdown, the connected minicom terminal

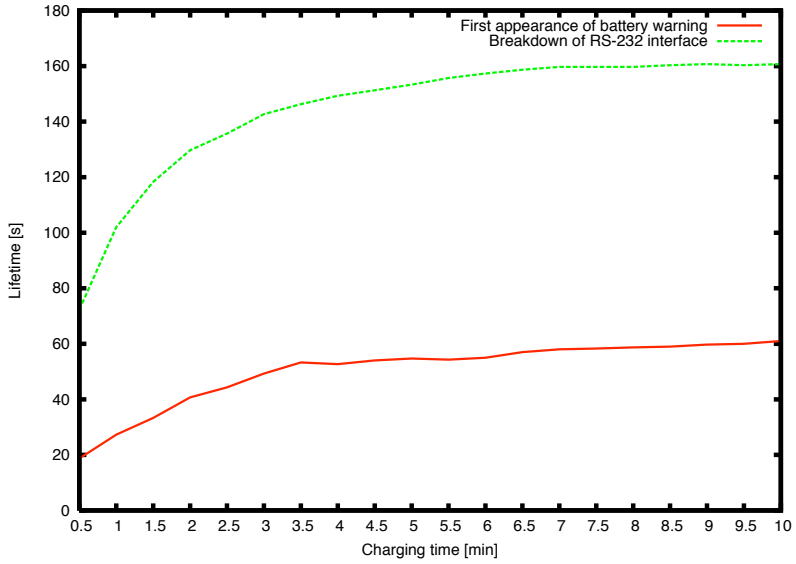


Figure 4.1: Correlation of GoldCap charging time and ESB lifetime using *Battery Warning* messages and RS-232 shutdown as indicators.

program starts to reports a lot of unreadable characters. In order to minimize possible side effects in the test network, only one node is powered with a capacitor, the others are powered by battery. We elect the node next to the gateway (*node 1* in Figures 4.2 and 4.3) as the measuring-node, because this node has to forward all other packets in the test network. Every network setup is measured ten times to get an average value. The next subsection contains detailed information about network setup and the conditions.

In order to analyze network characteristic data like packet loss, packet delay or bit errors, all nodes are powered by battery. This eliminates side effects of a quickly decreasing power level and makes it possible to collect information during 20 minutes. Every node collects information of all in- and out going packets, occurred errors and status information. This data is written into a packet and sent over the test network to the gateway. The gateway logs all incoming information with a timestamp. It provides us with informations on lost packets, jitter, quantity of retransmissions and checksum errors.

4.2 Experimental Setup

The nodes are tested in a *short-range* and a *long-range* network to analyze effects of interferences and bit errors. Figure 4.2 shows the *short-range* network, which consists of six ESB nodes. Every node receives every synchronization packet with a high signal-to-noise ratio. *Node 5* sends all data packets to *node 4*, that forwards all received packets to *node 3* and so on. Fi-

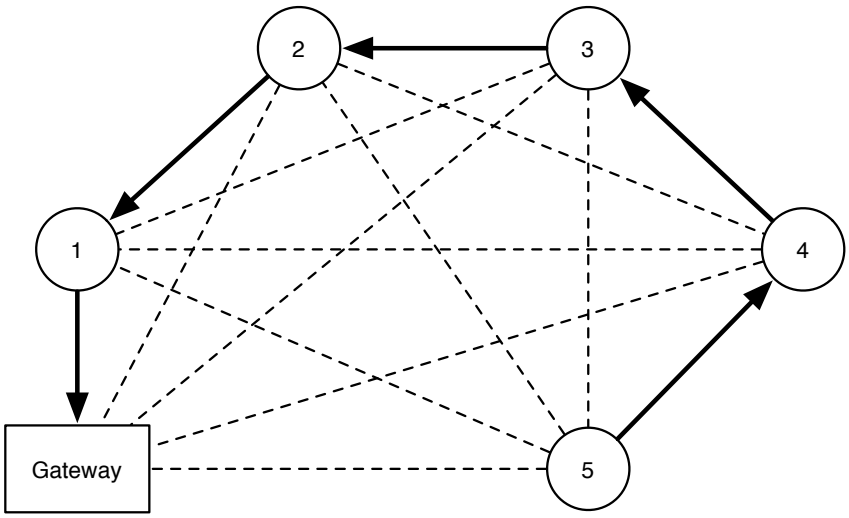


Figure 4.2: *Short range* network: The nodes build up a full mesh.

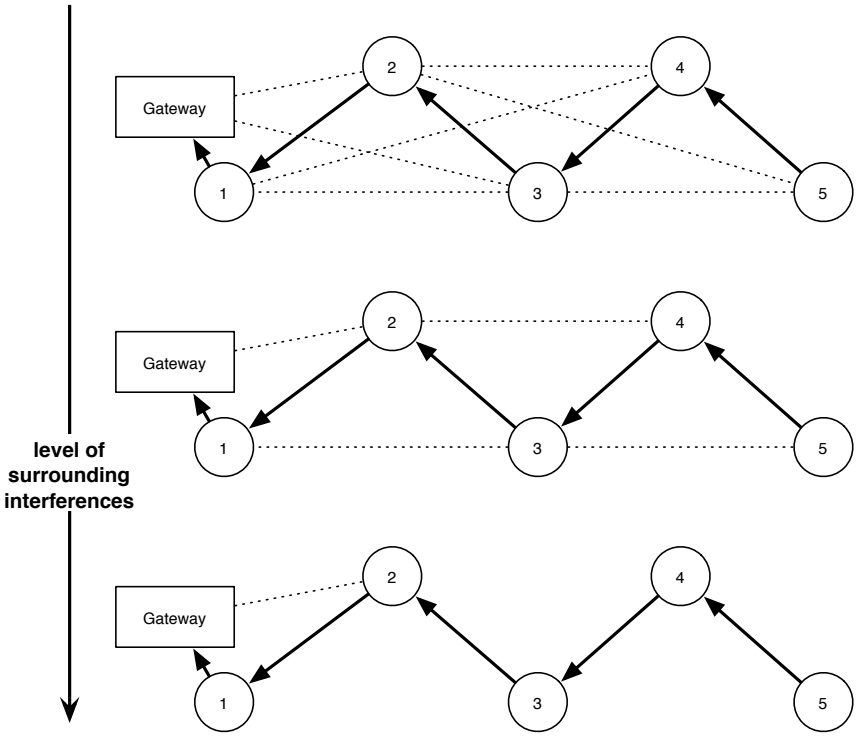
nally, the gateway node receives all messages. Small logic modifications and a static routing table disable the ability of LMAC of finding the shortest way to the gateway. The static routing table also guarantees the defined network structure for TEEM and *ScatterWeb CSMA*.

For the *long-range* network, as shown in Figure 4.3, the same nodes are distributed in the whole building. *Node 1* is located very close to the gateway. The other nodes are allotted in various rooms on different floors and receive in correlation to actual interference synchronization packets of one, two or three neighbor nodes.

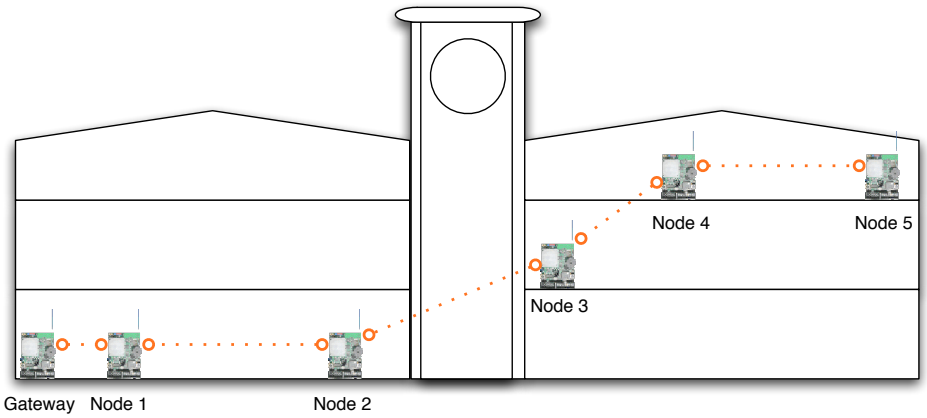
In both network scenarios, *Node 1* has the biggest load and is powered with one GoldCap capacitor to perform the lifetime measurements. The capacitor is kept continuously connected to the ESB node to guarantee a equal energy load for each measurement. After the RS-232 interface has been collapsed, the capacitor is immediately recharged for five minutes to start the next measurement. Two patterns imitate different traffic load in a sensor network and analyze dependency on network behavior. Therefore, each node tries to send 12 bytes of data every 10 seconds in the first and every 20 seconds in the second traffic pattern. Table 4.1 gives an overview of all scenarios.

scenario	network	packet load	protocols
1	*short-range*	packet each 10 s	*TEEM V1, V2*; *LMAC V1, V2*; *ScatterWeb CSMA*
2	*short-range*	packet each 20 s	*TEEM V1, V2*; *LMAC V1, V2*; *ScatterWeb CSMA*
3	*long-range*	packet each 10 s	*TEEM V1, V2*; *LMAC V1, V2*; *ScatterWeb CSMA*
4	*long-range*	packet each 20 s	*TEEM V1, V2*; *LMAC V1, V2*; *ScatterWeb CSMA*

Table 4.1: Different scenario for lifetime measurements.



(a) The network mesh depends on surrounding interferences.



(b) The nodes are distributed in the building.

Figure 4.3: *Long range* network: The network mesh depends on surrounding interferences.

Figure 4.4 shows an additional scenario. Four nodes with TEEM and four nodes with LMAC are running at the same time in the same transmission range. In this scenario, the only interest is the network behavior in order to analyze possible side when combining between the two protocols.

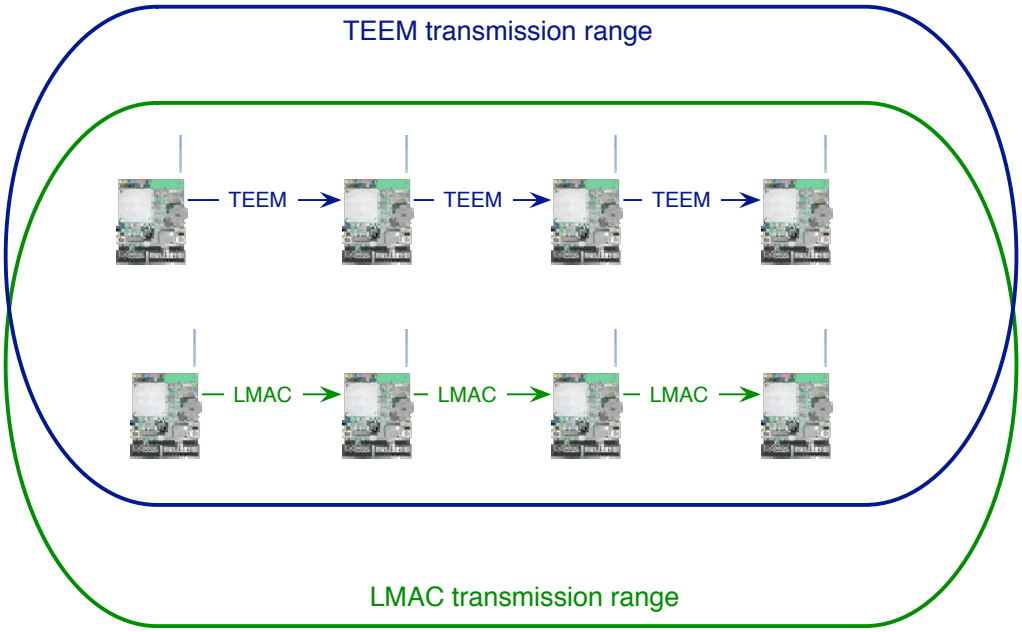


Figure 4.4: TEEM and LMAC working at the same time.

4.3 Measurement Results

The following subsections present measured results of TEEM, LMAC, and *ScatterWeb CSMA*. Measured values are: Energy consumption, packet loss, packet delay properties, checksum errors, and data rates. Some results have already been published at REALWSN 2006 [26].

4.3.1 Energy Consumption

To estimate the potential of an energy efficient MAC protocol, the average energy consumption of several ESB nodes are measured with an ampere-meter. Figure 4.5 shows six ESB nodes with radio and RS232 interfaces switched on and off. The RS232 interfaces is switched off by disconnecting the connection cable. All sensors were disabled for this measurements. With disabled RS232 interfaces and disabled sensors, the radio interface in listening mode is responsible for almost half the energy consumption. This is only an indication of the potential for energy saving, as power consumption depends on many other factors, such as activated sensors or how often the MSP430 can go into a low-power mode. Sensors can consume up to 2.9 mA. The

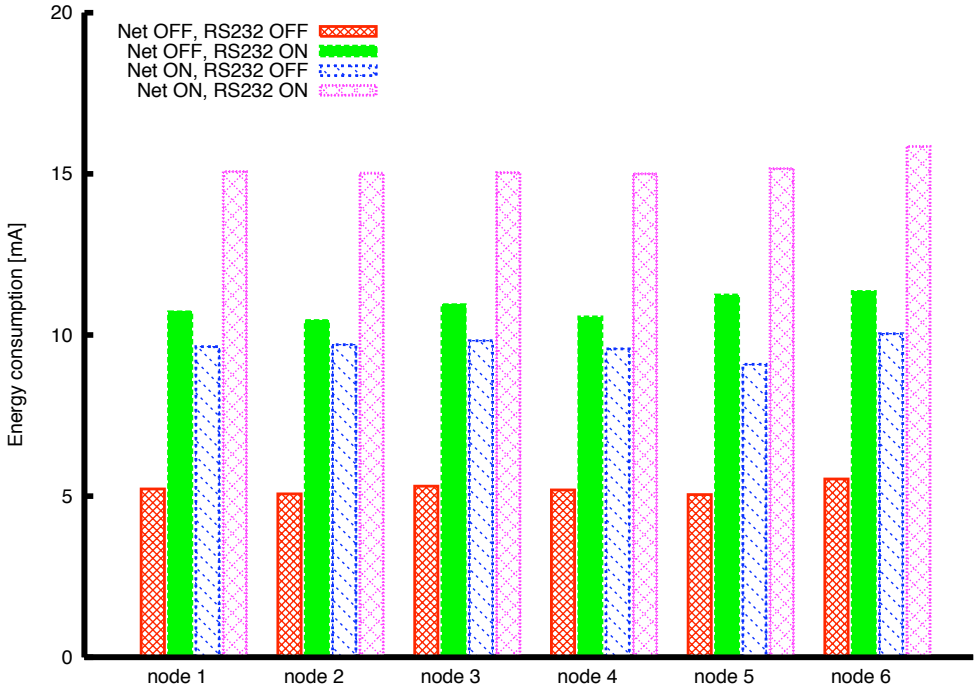


Figure 4.5: ESB energy consumption with and without enabled radio and RS232 interfaces at 5.5 V.

biggest consumer with 80 mA is the built-in IR-transceiver. Writing the EEPROM needs 3 mA, reading 0.4 mA.

The red line at 182.5 seconds shows the lifetime of a node with an always turned off radio. This represents the longest possible lifetime and visualizes the potential of TEEM and LMAC compared to a protocol with no energy efficiency. *ScatterWeb CSMA* has the highest energy usage which results in the shortest lifetime. A higher packet rate and more retransmissions in *long-range* scenarios have no clear identifiable effect in lifetime. Protocol *TEEM V1* can save 62 - 73 % energy in comparison to *ScatterWeb CSMA*. The amount of transmitted packets has a detectable effect of energy usage. The radio requires about 2% more energy in *long-range* scenarios. This effect is much smaller in *TEEM V2*. The up to eight time higher bandwidth allows *TEEM V2* to save additional preambles and to increase the sleeping part proportion. In all *LMAC V1* scenarios the radio consumes about 2 - 3 % more energy than in *LMAC V2* scenarios. The network must be reset several times between the individual lifetime measurements for *LMAC V1 long-range* because of the periodical building of subnetworks. LMAC loses a lot of energy to support a long enough listen window for compensating the time shifts in *long-range* scenarios. An additional LMAC version constructed only for *short-range* scenarios is tested in order to probe the potential of LMAC architecture. This version is able to save more the 90 % of radio power with a average lifetime of around 175 seconds.

Figure 4.6 shows the aggregation of all lifetime measurements.

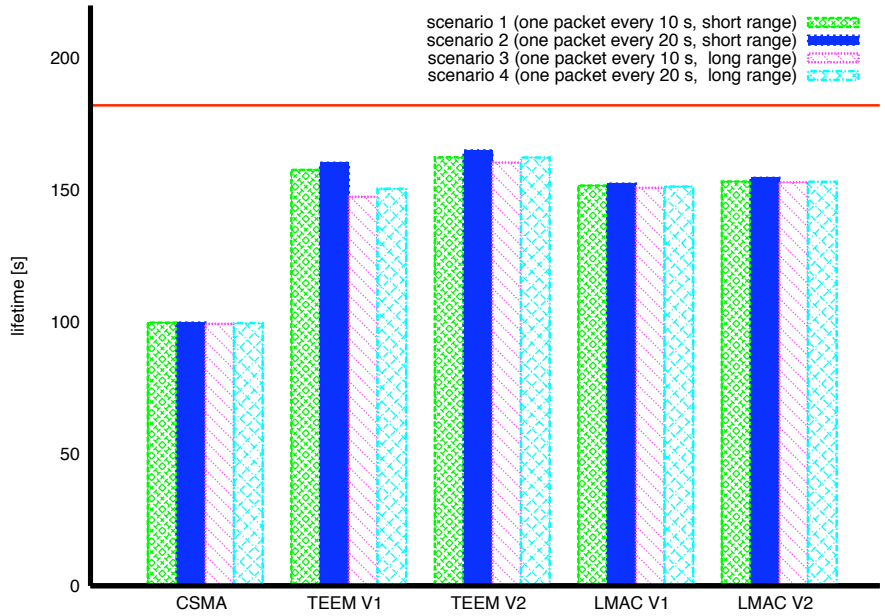


Figure 4.6: Aggregation of all lifetime measurements.

4.3.2 Packet Loss

Figure 4.7 shows the different packet losses in periods of 20 minutes. In the CSMA protocol in *scenario 2* has an unexpected high packet loss of totally three packets. They are all lost successively, which indicates there was high interference which has jammed the transmissions. The *long-range* scenarios all packets are lost in a row, too.

Both TEEM protocols have the smallest average packet loss except *scenario 3* of *TEEM V1*. The huge packet loss in *scenario 3* is based on a network overload because of additional retransmissions in *long-range* scenarios. The network log files show that *node 5* is responsible for 94% of the packet loss.

Because the *LMAC V1 long-range* scenario tends to build clusters, which disconnect the packet route, there are no results for the two scenarios. *Long range* scenarios of *LMAC V2* have huge packet losses. An extra period of 6 hours was recorded for a superior analysis which shows the same high packet loss. The missed packets are not evenly distributed over the whole time period. There are periods with a lot of lost packets and other ones with only few dropped packets.

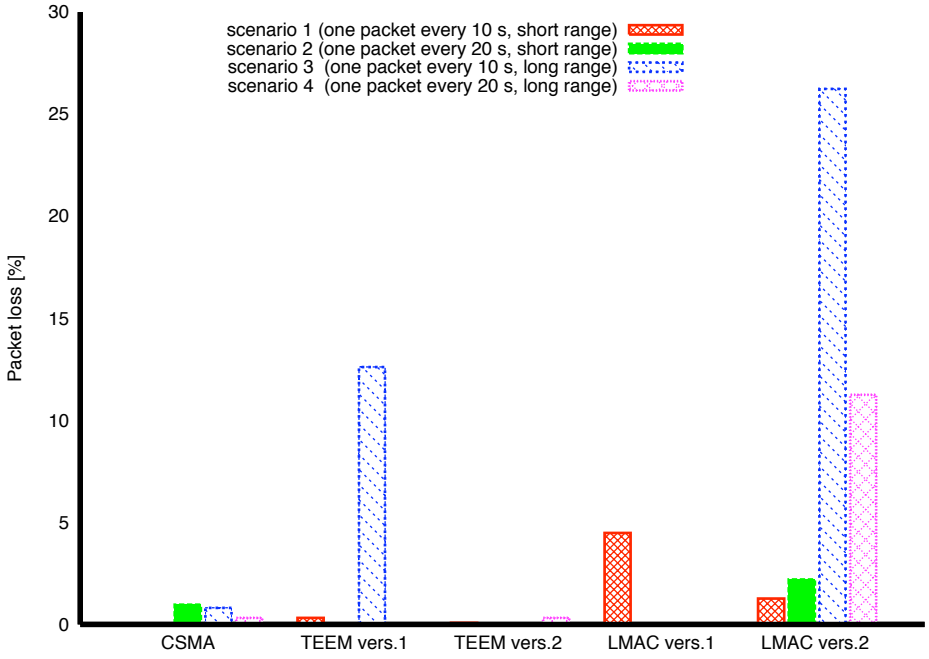


Figure 4.7: Packet loss of all scenarios and protocols.

4.3.3 Packet Delay Variance

The network log files makes it possible to calculate the delay variance (jitter) of incoming packets. Jitter consists of two different factors, notably delay time (latency) and variance. Latency is the time interval which a packet needs to be sent from sender to the receiver.

Like variance it depends on several factors like the hop count, the actual network load, the hardware used or the protocol implementation. The variance is the average time difference of each incoming packet of the average packet delay. We use the better understandable *95% confidence interval* in order to display the variance. It describes the interval in which the next packet will arrive with a 95% probability.

Figure 4.8 shows the different confidence intervals. Figure 4.9 illustrates the different packet delays of all nodes in *ScatterWeb CSMA* protocol scenarios. Both *short-range* scenarios have the smallest latency and variance of any other readings. In comparison to the *short-range* scenarios, where the nodes build a full mesh, the remote nodes in *long-range* scenarios have a bigger latency and a lot of outliers. The average latency and confidence interval in *long-range* scenarios illustrate a very disordered packet latency. Figure 4.9(c) and 4.9(d) visualize the *long-range* scenarios. *ScatterWeb CSMA* responds clearly to a lower signal-to-noise ratio even though this protocol has all recommended preamble bytes.

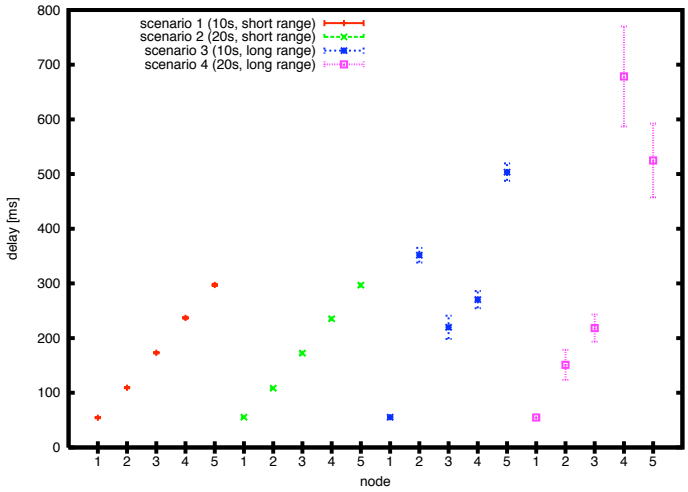
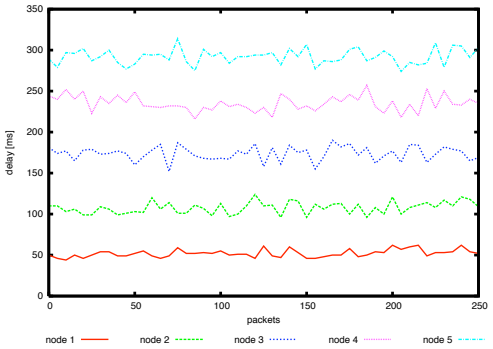
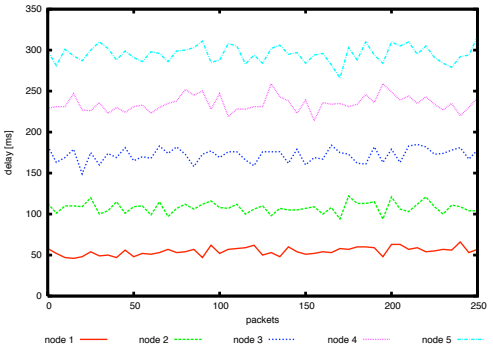


Figure 4.8: Average packet latency with a 95 % confidence interval for *ScatterWeb CSMA* scenarios.

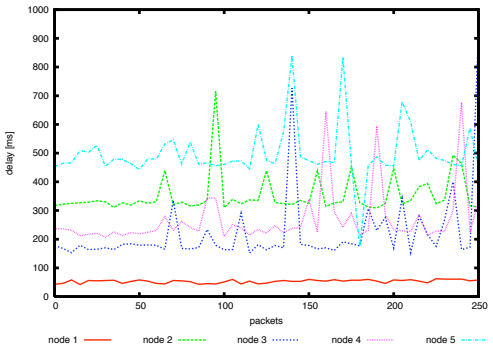
All LMAC scenarios get more or less the same packet loss diagrams. Responsible for that is the frame length of 5.12 s and the periodical packet generation of each node. It results in a ”sawtooth” look as shown in Figure 4.11. The sporadic irregularities in the diagrams arise from packet loss. The huge frame length result in a enormous latency. An important factor for the latency is the slot number allocation of each node. In an optimal case, where the next node also



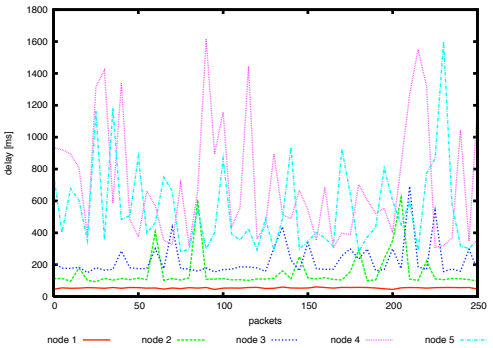
(a) *ScatterWeb CSMA* scenario 1.



(b) *ScatterWeb CSMA* scenario 2.



(c) *ScatterWeb CSMA* scenario 3.



(d) *ScatterWeb CSMA* scenario 4.

Figure 4.9: *ScatterWeb CSMA* packet delay for all scenarios.

holds the next slot number every hop generates a 160 ms delay. In the worst case, where the next node owns the last slot number, a hop produces a 31 times higher latency of 4960 ms. As long as every node can send its produced and received data in the next time slot, the estimated packet latency could theoretically be calculated.

The 95 % confidence intervals for estimated latency is shown in Figure 4.10. Every node has quite the same confidence interval in each scenario. It mainly depends on the non-continuous arrival times based in the TDMA slot architecture. Considering this behavior, the 95 % confidence interval for next incoming packet decreases for example for *LMAC V1* from about ±400 ms to ±4 ms. Unlike *ScatterWeb CSMA* and TEEM, *LMAC V2* shows no conspicuous difference between *short-range* and *long-range* scenarios.

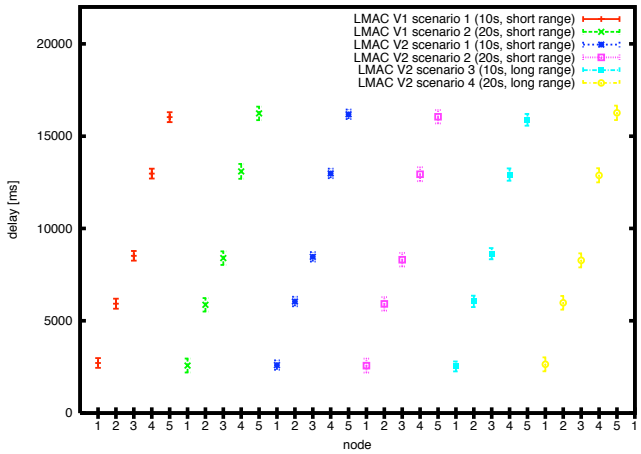
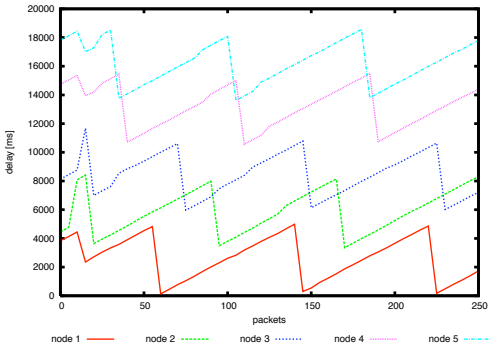
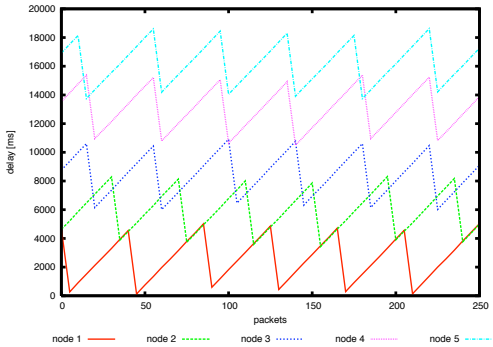


Figure 4.10: Average packet delay with a 95 % confidence interval for LMAC scenarios.

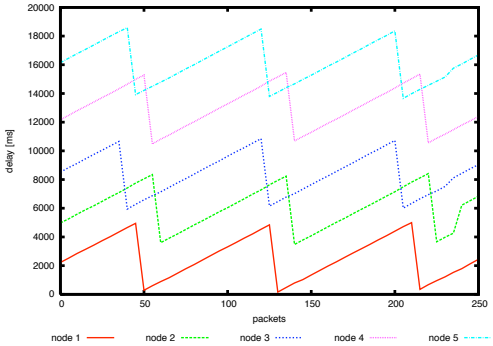
The latency of TEEM is smaller in comparison to LMAC. Figure 4.12 illustrates that latency and confidence interval depend heavily on the traffic load in *TEEM V1*. Figures 4.13 and 4.14 show that latency is very unsteady in all TEEM scenarios. Especially in scenarios with high traffic load, the confidence interval is bigger and the average latency of the different nodes is snatchier. *TEEM V2* scenarios do not have such a clear difference between the different traffic loads as *TEEM V1*. But the average latencies of small traffic load scenarios are a little higher. The bigger bandwidth of *TEEM V2* seems to be the only positive effect on packet delay variance which avoids capacity overload effects in high traffic load scenarios.



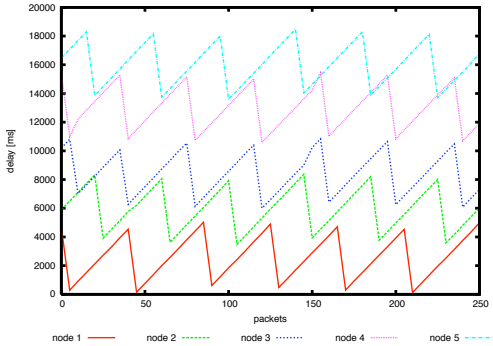
(a) *LMAC V1* scenario 1.



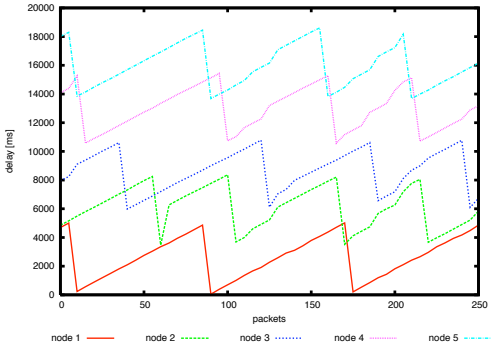
(b) *LMAC V1* scenario 2.



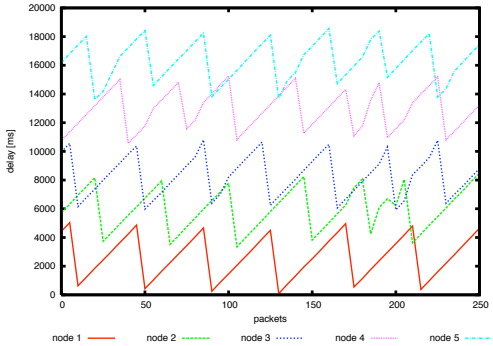
(c) *LMAC V2* scenario 1.



(d) *LMAC V2* scenario 2.

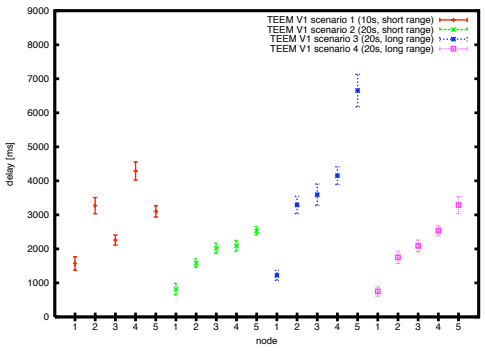


(e) *LMAC V2* scenario 3.

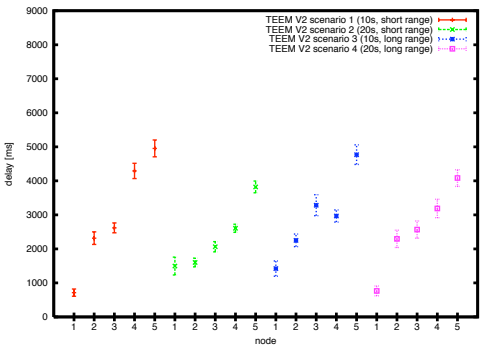


(f) *LMAC V2* scenario 4.

Figure 4.11: Packet delay variance of all LMAC nodes in different scenarios.

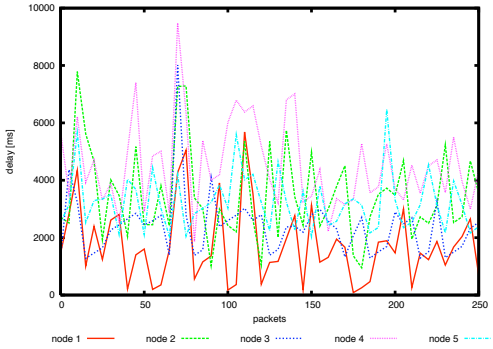


(a) *TEEM* scenarios.

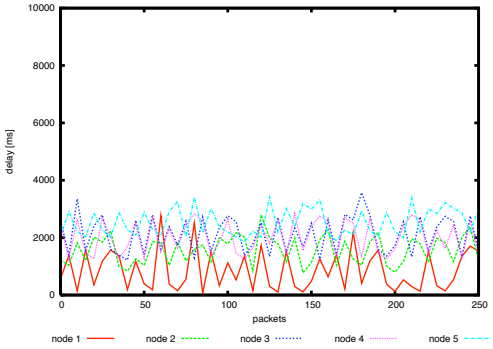


(b) *TEEM long-range* scenarios.

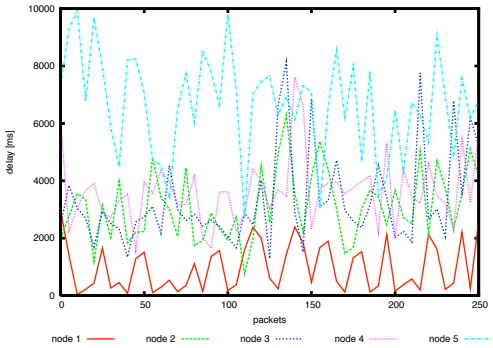
Figure 4.12: Average packet latency with a 95 % confidence interval for TEEM scenarios.



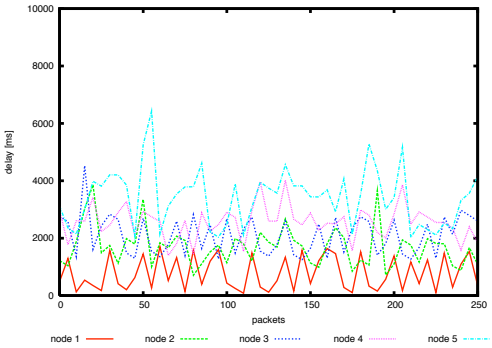
(a) *TEEM* scenario 1.



(b) *TEEM* scenario 2.

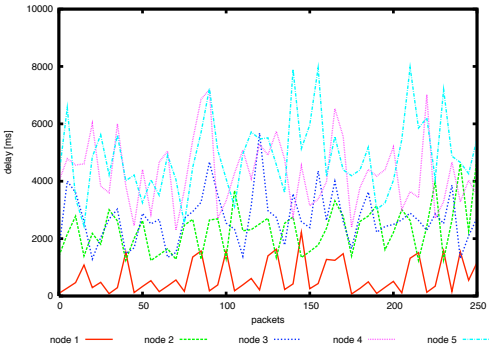


(c) *TEEM V1* scenario 3.

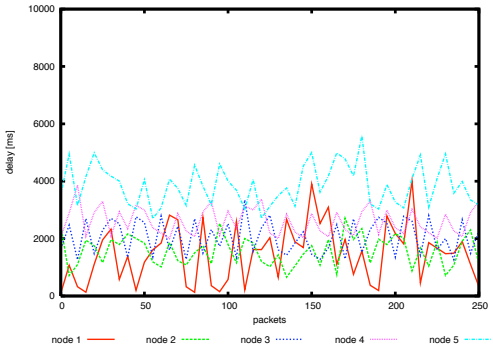


(d) *TEEM V1* scenario 4.

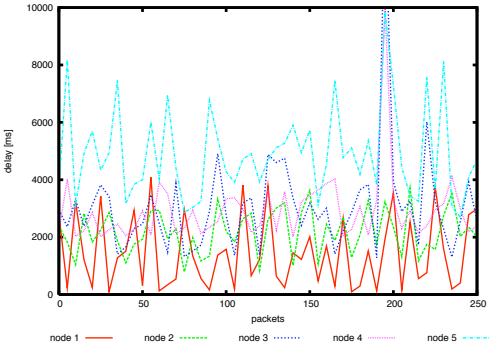
Figure 4.13: Packet delay variance of *TEEM V1* nodes in all scenarios.



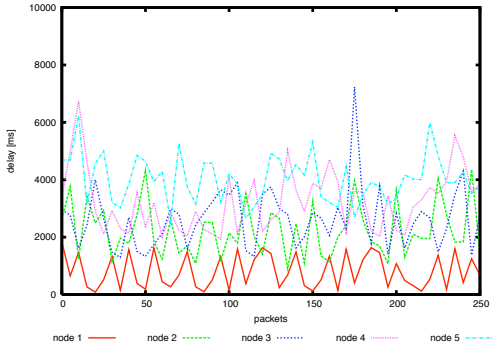
(a) *TEEM V2* scenario 1.



(b) *TEEM V2* scenario 2.



(c) *TEEM V2* scenario 3.



(d) *TEEM V2* scenario 4.

Figure 4.14: Packet delay variance of *TEEM V2* nodes in all scenarios.

4.3.4 Checksum Errors

Hardly any checksum error occurs for CSMA and LMAC in *short-range* scenarios, as shown in Figure 4.15. The collisions of $SYNC_{rts}$ packets are the main responsibles for the checksum errors in TEEM. This is caused by the switch time problem described in section 3.2. In contrast to *long-range* scenarios in Figure 4.16, no $DATA$ packet of TEEM and LMAC have a checksum error.

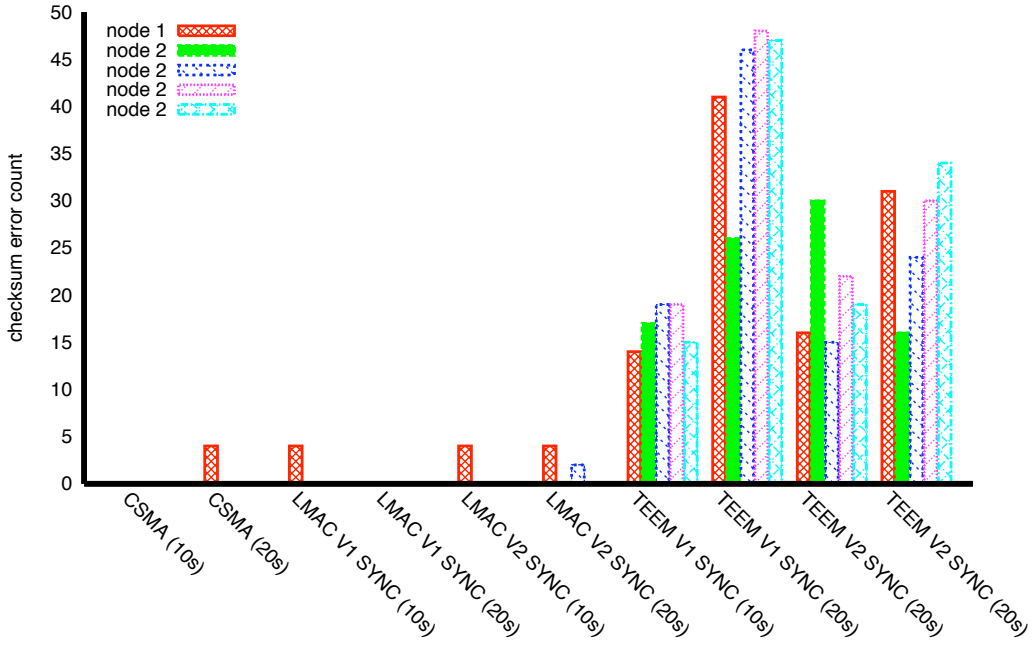


Figure 4.15: Checksum errors in all *short-range* scenarios.

In *long-range* scenarios all protocols suffers from bit errors. In spite of a longer preamble in *ScatterWeb CSMA*, a lot of data packet checksum errors occurred. Unlike TEEM, where collision during the contention based period can occur, LMAC should be collision free. Therefore, the occurred checksum errors in LMAC have to be a result of interferences. Generally, jamming interferences often occur in a burst, during several times longer than a packet transmission. Hence, most checksum errors in TEEM and *LMAC V2* occurred during the data transmission foregoing control message.

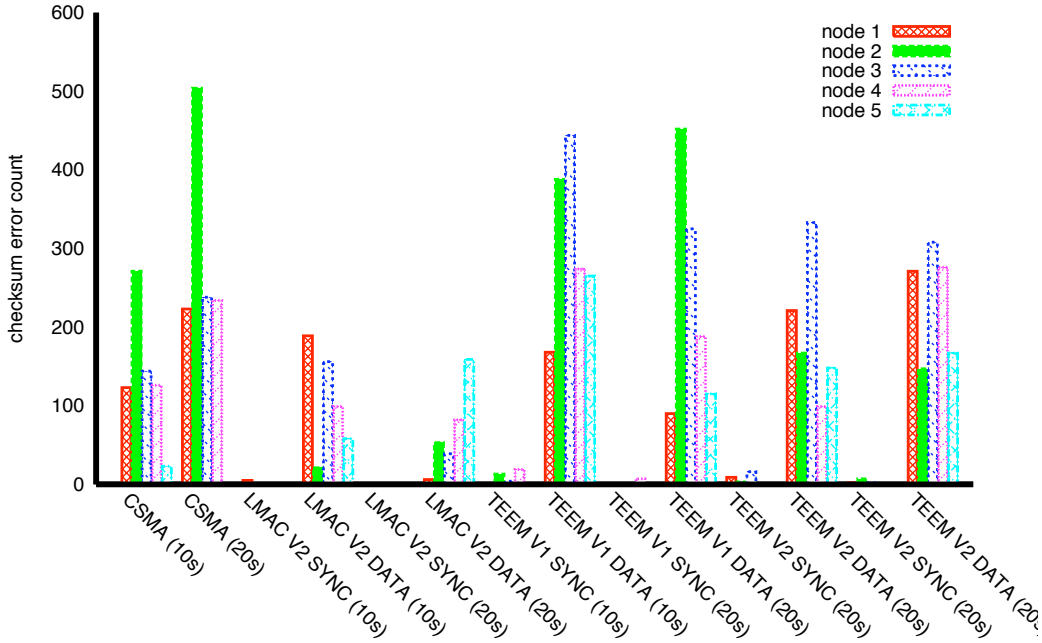


Figure 4.16: Checksum errors in all *long-range* scenarios.

4.3.5 TEEM and LMAC Simultaneous Scenario

The following setup describes the simultaneous deployment of TEEM and LMAC. The four tested scenarios are shown in Table 4.2. Every node has a strong signal from all other nodes as shown in Figure 4.4 (section 4.2). Only *short-range* scenarios are tested. It was not possible to

scenario	protocols	packet rate
A	*TEEM V1* versus *LMAC V1*	one packet each 10 s
B	*TEEM V1* versus *LMAC V1*	one packet each 20 s
C	*TEEM V2* versus *LMAC V2*	one packet each 10 s
D	*TEEM V2* versus *LMAC V2*	one packet each 20 s

Table 4.2: Different scenario for lifetime measurements.

setup a LMAC network, while a TEEM network was running. First, the LMAC network must be established and afterwards the TEEM nodes can be activated. Shortly after the TEEM nodes starts to send the packets, the network route of *LMAC V1* collapses because of cluster building. *LMAC V2* is more stable, but loses more than 60 % of all data packets. TEEM has no packet loss, but has about twice the numbers of CRC errors than in *long-range* scenarios. This cuts theoretical maximal bandwidth in half.

4.3.6 Data Rate

Table 4.3 shows the maximum data rate of all analyzed protocols at 19.2 kbps with data packets of 28 bytes in a one hop scenario. *LMAC V1* and *LMAC V2* can send in one slot up to eight

	max. data rate [kbit/s]			
protocol	28 byte packet	28 byte packet	28 byte packet	28 byte packet
LMAC V1, V2	0.350	0.400	0.400	0.400
TEEM V1	0.373	0.853	1.71	3.41
TEEM V2	2.99	6.83	13.7	17.1
ScatterWeb CSMA	3.19	4.94	6.14	7.11

Table 4.3: Maximum bit rate over one hop at 19.2 kpbs with data packets of 28 bytes.

packets if they are combined not bigger than 256 byte. The data rate is 0.4 kbps using all 256 bytes in a slot otherwise smaller. A multi hop scenario would have no effect on the maximum data rate as long as every node can send in the same frame. *TEEM V2* can send up to eight packets in a slot as long as they are smaller than 1.6 kbytes. The effective data rates are lower because of bit errors which result in packet loss (LMAC) or retransmissions (TEEM). In order to calculate the bit rate of *ScatterWeb CSMA*, we used Equation 4.2 for CSMA protocols described in [28]. The maximum data quota (S_{max}) is derived from the empty channel detection delay t

(3.33 ms), raw channel bit rate C (19.2 bits/ms) and bits per packets L.

$$S_{max} = \frac{1}{1 + 2 * \sqrt{\frac{C*t}{L}}} \tag{4.2}$$

The results for *ScatterWeb CSMA* are calculated in consideration of the additional bytes of ScatterWeb and an average backoff time of 16 bytes.

Chapter 5

Conclusions and Outlook

5.1 Conclusions

Our measurements show that the implemented energy efficient LMAC and TEEM protocols substantially decrease the sensor nodes' energy consumption. In comparison to *ScatterWeb CSMA*, the energy consumption of the radio transceiver can be reduced by 60% - 75% using the implemented energy efficient protocols.

The measurements show TEEM to be superior to LMAC in energy consumption, packet loss and average packet delay.

Hardware limitations of the ESB nodes made additional energy consuming adjustments necessary. Especially LMAC is loosing capability to compensate the hardware disadvantages.

It was not possible to set up a stable LMAC in a widely distributed network with the control messages described in [1]. Only an additional CRC to detect corrupted control messages enables LMAC on the ESB platform to build a synchronized network in a *long-range* scenario. Further, the missing retransmission possibility was responsible for a substantial packet loss in the *long-range* scenarios. The long packet delay of LMAC is caused by the frame length of 5.12 seconds. LMAC shows no recognizable difference in energy consumption between the different scenarios and packet loads.

TEEM suffers from a long transceiver switch time on the ESB platform of 2 ms, which is responsible for scores of $SYNC_{rts}$ collisions. The necessary retransmissions decreases bandwidth and increases energy consumption and packet delay. Higher packet load in TEEM result in higher energy consumption and jitter. However, as long as TEEM provides enough bandwidth to every node, packet loss is still quite low.

Interferences and small signal-to-noise ratios in long-range scenarios pose a challenge for both protocols. Because *short-range* scenarios have a generally high signal-to-noise ratio from all nodes in the actual network, interferences of other devices are negligible. In *long-range* scenarios the receivers have a low signal-to-noise ratio from one-hop neighbor nodes and interferences produced by two-hop or three-hop neighbor nodes. Also interferences by other devices have more effect because of the lower signal-to-noise ratio.

In comparison to the time shift of received messages in *long-range* scenarios ($\pm$ 5 ms) the clock shift of the MSP430 micro-controller is negligible. The long time shift is responsible for longer listen times at each slot start and increases the energy consumption of TEEM and

especially LMAC.

Therefore analyzing hardware limitations and network behavior of energy-efficient sensor MAC protocols do not yield significant results in *short-range* scenarios.

5.2 Outlook

This final section outlines some of the additional ideas and measurement scenarios which could not be tested within the scope of this work:

- A higher transmitter speed of 115.2 kbps with an amplitude-shift keying (ASK) modulation could enable shorter listen times. The ASK modulation could also have a positive effect on interference robustness .
- A spectrum analyzer may be useful for detecting interferences and analyzing network behavior.
- The slot number allocation could be optimized. In most scenarios the nodes do not move and they always send data to the same sink. Therefore, the packets always flow in the same direction and the coordinated slot numbers increase packet delay.
- LMAC control messages provide information about gateway distance to minimize the hop count of a packet. This can result in long transmission distances between nodes with high bit error rates. Shorter hop distances would enable more stable transmission.
- Further results of network scenarios incorporating a higher number of nodes distributed over a larger area, as well as bigger packet sizes, would permit us to draw more conclusions.

Bibliography

[1] L. van Hoesel and P. Havinga. A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In *INSS*, Japan, 2004.

[2] C. Suh and Y.-B. Ko. A traffic aware, energy efficient mac protocol for wireless sensor networks. In *IEEE 2005 International Symposium on Circuits and Systems (ISCAS'05)*, May 2005.

[3] Scatterweb. Platform for self-configuring wireless sensor networks. http://www.scatterweb.net. Last visit September 2006.

[4] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004.

[5] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, Los Angeles, California, USA, 2003.

[6] M. Ringwald and K. Römer. Bitmac: A deterministic, collision-free, and robust mac protocol for sensor networks. In *Proceedings of 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 57–69, Istanbul, Turkey, January 2005.

[7] V. Rajendran, K. Obraczka and J. J. Garcia-Luna-Aceves. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. *Conference On Embedded Networked Sensor Systems*, pages 181–192, 2003.

[8] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, Baltimore, Maryland, USA, November 2004.

[9] P. Lin, C. Qiao and X. Wang. Medium access control with a dynamic duty cycle for sensor networks. *IEEE Wireless Communications and Networking Conference (WCNC)*, Vol. 3, pages 1534–1539, 2004.

[10] J. Ai, J. Kong and D. Turgut. An adaptive coordinated medium access control for wireless sensor networks. *International Symposium on Computers and Communications*, Vol. 1, pages 214–219, 2004.

[11] H. Pham and S. Jha. An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC). *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 214–226, 2004.

[12] A. El-Hoiydi and J. -D. Decotignie. Wisemac: An ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. *ISCC2004, the 9th IEEE International Symposium on Computers and Communications,*, Alexandria, Egypt, June 2004.

[13] S. Singh and C. Raghavendra. PAMAS - power aware multi-access protocol with signaling for ad hoc networks. *SIGCOMM Computer Communications Review 28*, pages 5–26, 1998.

[14] W. B. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, An applicationspecific protocol architecture for wireless microsensor networks *IEEE Transactions on Wireless Communications 1*, pages 660–670, 2002.

[15] S. Chatterjea, L. van Hoesel and P. Havinga. AI-LMAC: An adaptive, informationcentric and lightweight MAC protocol for wireless sensor networks. *Intelligent Sensors, Sensor Networks, and Information Processing Conference*, pages 381–388, 2004.

[16] L. van Hoesel, P. Havinga, Poster abstract: A TDMA-based MAC protocol for WSNs. *International Conference on Embedded Networked Sensor Systems (SenSys)*. pages 303–304, 2004.

[17] R. Kalidindi, R. Kannan, S. S. Iyengar, and L. Ray. Distributed energy aware mac layer protocol for wireless sensornetworks. In *International Conference on Wireless Networks*, pages 282–286, 2003.

[18] I. Rhee, A. Warrier, M. Aia, J. Min, TinyOS is an open source operating system for wireless sensor networks. *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 90–101, 2005.

[19] TinyOS. Z-MAC: A hybrid MAC for wireless sensor networks. http://www.tinyos.net/ Last visit September 2006.

[20] MSP430. Texas instruments MSP430 microcontroller family. http://focus.ti.com/docs/prod/folders/print/msp430f149.html Last visit September 2006.

[21] Texas Instruments. User's guide for the micro controller MSP430F149 from Texas Instruments. http://focus.ti.com/lit/ug/slau049f/slau049f.pdf.

[22] Minicom. Minicom is a menu driven communications program. It emulates ANSI and VT102 terminals. http://alioth.debian.org/projects/minicom. Last visit September 2006.

[23] Gentoo. Gentoo Linux Project. http://www.gentoo.org. Last visit September 2006.

[24] RF Monolithics. Datasheet for TR1001 868.35 MHz hybrid transceiver. http://www.rfm.com/products/data/tr1001.pdf.

[25] mspgcc. GCC toolchain for the Texas Instruments MSP430 MCUs. http://mspgcc.sourceforge.net/ Last visit September 2006.

[26] T. Staub, T. Bernoulli, M. Anwander, M. Waelchli and T. Braun. Experimental Lifetime Evaluation for MAC Protocols on Real Sensor Hardware. In *ACM Workshop on Real-World Wireless Sensor Networks REALWSN 2006*, pages 25–29, Uppsala, Sweden, June 2006.

[27] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso. Experimental evaluation of lifetime bounds for wireless sensor networks. In *European Workshop on Wireless Sensor Networks ESWN 2005*, Istanbul, Turkey, January 2005.

[28] C. Wan, S. B. Eisenman and A. T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks In *ACM SENSYS 2003*, November. 2003.