Bachelor thesis

A TDMA Module for Waterborne Communication with Focus on Clock Synchronization

by

Anders Persson

LIU-IDA-SAS

2013-09-15
Final Thesis

A TDMA Module for Waterborne Communication with Focus on Clock Synchronization

by

Anders Persson

LIU-IDA/ LIU-IDA/LITH-EX-G--13/047--SE

2013-09-15

Supervisor: Amir Aminifar
IDA, Linköpings universitet

Examiner: Unmesh Bordoloı
IDA, Linköpings universitet
This bachelor thesis has been carried out at the company Didamus which is located in Mjärdevi, Linköping. The company is currently developing a dive-console which aims to take the scuba diving experience to a whole new level and also to make scuba diving more secure. An assembly of scuba divers that can communicate with each other during a dive session might be the difference between life and death. Many seas around the world have muddy water which means poor visibility. In each situation a computer providing a scuba diver with information about others connected to the network, hazardous accidents can possibly be avoided.

The network itself consist of 10 nodes that need a network protocol which provides stability and reliability for every participant. The nodes themselves have a distributed responsibility to make the network reliable. The type of network implemented was a regular Time Division Multiple Access (TDMA) network where different nodes were given permission to access the medium in different instances of time. A global reference of time is always needed in a TDMA network to make it function properly. In a typical TDMA network a GPS-service gives each and every node information about the global time. Unfortunately, GPS-services do not work well in water so a Master-Slave method was used instead. The master provides the rest of the nodes in the network with a global time reference. After a successful reception of a global time reference, the slave will be granted access to the network.

The communication between the nodes is based on ultrasonic waves propagating in the water. The velocity of ultrasonic waves in water is only 1500 meters per second, explained in *Discovery of Sound in the Sea* by University of Rhode Island, which is a relatively slow signal speed. With the slow velocity taken into account an efficient TDMA protocol was developed, to perform communication under water.
Abstract

This bachelor thesis has been carried out at the company Didamus which is located in Mjärdevi, Linköping. The company is currently developing a dive-console which aims to take the scuba diving experience to a whole new level and also to make scuba diving more secure.

An assembly of scuba divers that can communicate with each other during a dive session might be the difference between life and death. Many seas around the world have muddy water which means poor visibility. In each situation a computer providing a scuba diver with information about others connected to the network, hazardous accidents can possibly be avoided.

The network itself consist of 10 nodes that need a network protocol which provides stability and reliability for every participant. The nodes themselves have a distributed responsibility to make the network reliable. The type of network implemented was a regular Time Division Multiple Access (TDMA) network where different nodes were given permission to access the medium in different instances of time. A global reference of time is always needed in a TDMA network to make it function properly. In a typical TDMA network a GPS-service gives each and every node information about the global time. Unfortunately, GPS-services do not work well in water so a Master-Slave method was used instead. The master provides the rest of the nodes in the network with a global time reference. After a successful reception of a global time reference, the slave will be granted access to the network.

The communication between the nodes is based on ultrasonic waves propagating in the water. The velocity of ultrasonic waves in water is only 1500 meters per second, explained in Discovery of Sound in the Sea by University of Rhode Island, which is a relatively slow signal speed. With the slow velocity taken into account an efficient TDMA protocol was developed, to perform communication under water.
Acknowledgements

First of all, I want to say that I am really grateful that I was given the chance to do my thesis on this subject since I am very interested in embedded system communication.

I would like to thank the Computer and Information Science Department and Software and Systems Division at Linköpings University, especially Unmesh Bordoloi for being my examiner and Amir Aminifar for supervising me along the way. I also would like to thank Anders Brodin for supervising me at the company. I also want to give a commendation to all staff at Didamus for being friendly and supportive.

Linköping, Sweden, August 2013

Anders Persson
Chapter 1  Introduction ...........................................................................................................1
  1.1 Company Description .................................................................................................1
  1.2 Background .............................................................................................................1
  1.3 Purpose ..................................................................................................................1
  1.4 Objective ..............................................................................................................2
  1.5 Method ..................................................................................................................2

Chapter 2  Background Theory ..........................................................................................3
  2.1 TDMA Networks .....................................................................................................3
  2.2 Real-Time Systems .................................................................................................4
  2.3 Distributed Clock Synchronization .........................................................................4
  2.4 The Precision Time Protocol ..................................................................................4
  2.5 Clock Drifting ...................................................................................................... 5

Chapter 3  Hardware and Software ..................................................................................7
  3.1 Sam4s Xplained Evaluation Kit ...............................................................................7
  3.2 AM335x Starter Kit ................................................................................................ 8
  3.2.1 Hardware ........................................................................................................8
  3.3 Atmel Studio 6 ......................................................................................................9
  3.4 GNU Emacs ........................................................................................................9

Chapter 4  System Design ...............................................................................................11
  4.1 Brief Explanation of the Architecture ................................................................... 11
  4.2 Characteristics of the Network ..............................................................................11
  4.3 The TDMA MAC Protocol ....................................................................................12
  4.3.1 Unique Conditions for the TDMA Network .....................................................12
  4.3.2 The TDMA Network Structure .....................................................................13
  4.3.3 The Timeslot Size ..........................................................................................13
  4.4 TDMA and Clock Synchronization .......................................................................14
  4.4.1 The Different Messages ................................................................................14
  4.5 The Time Synchronization Protocol ......................................................................15
  4.5.1 The Startup Phase ..........................................................................................15
  4.5.2 The Slave Answering Phase .........................................................................16
  4.5.3 Distance Calculation ......................................................................................16
  4.5.4 Slave Answers Received ...............................................................................17
  4.5.5 Sync Packet Arrives .....................................................................................18
  4.5.6 The Clock Adjustment ..................................................................................18
  4.5.7 A Bound Medium Access Time for the Slaves .............................................19

Chapter 5  Results and Experiments ...............................................................................21
  5.1 Recognition of Different Messages .........................................................................21
  5.2 Tests of a varying delay .........................................................................................21
  5.3 Estimation of Processing Delay Time ....................................................................21
  5.4 Tests of clock drifting ............................................................................................22
  5.5 Tests below and over sea-level ...............................................................................22

Chapter 6  Discussion ......................................................................................................23
  6.1 A Valiant Attempt to Implement the TDMA-network in higher Network Layers ......23
  6.2 Testing the Functionality .......................................................................................24
  6.3 Errors caused by movement ..................................................................................24
  6.4 Different conditions when used under water .......................................................24

References ......................................................................................................................25
Chapter 1

Introduction

This chapter gives an introduction to the thesis.

1.1 Company Description

The thesis was undertaken at the company Didamus which was founded by Anders and Maria Brodin in 2012. The company is developing a dive-console which will connect scuba divers in a waterborne network. After connection has been established, data can be shared between participants of the network. The objective of the company is to give a better overall experience during scuba diving and to decrease scuba diving accidents. Examples of messages that will be sent over the network are different kinds of sensor data and alarm signals.

1.2 Background

There are many active scuba divers around the world, and with this wonderful experience comes a possible risk of being involved in an accident. The communication between scuba divers is limited to gestures in the water or physical contact when a possible accident occurs. In many places around the world the water is turbid and may as well be dark which gives a very short line of sight. Many accidents could be avoided if the participants in a dive session could send alarms and messages to each other. Thus, a dive-console with a secure and reliable network protocol is needed.

1.3 Purpose

The different nodes in an ad-hoc network have a distributed responsibility to make the communication reliable. Therefore a protocol is needed to organize the nature of temporal accesses to the medium. If, for example, two different nodes access the medium at the same time, two different sound waves would superimpose each other and the packets sent would arrive corrupted. Since no regulation exists at this level, a TDMA based Medium Access Control implementation is needed to control when different nodes should be granted access to the medium. A global time reference is also needed for a TDMA network to function properly, so a clock synchronization protocol for the particular circumstances is also needed.
1.4 Objective

The objective is to develop a TDMA based MAC protocol for a network consisting of 10 nodes so they are able to communicate with each other with a transfer speed high enough to present pertinent data between the users.

1.5 Method

A recess in previous implementations of TDMA protocols suited for similar scenarios will be a key factor to make a successful protocol for the specific unit. The implementation of the protocol will be made in a program called Atmel Studio 6, and quality tests will be performed to validate the functionality of the TDMA module.
Chapter 2

Background Theory

This chapter provides the background theory needed for the design of a TDMA module for waterborne communication.

2.1 TDMA Networks

A TDMA network is a network where different nodes broadcast their data over the medium at different instances of time, a so called timeslot [1]. The broadcast is done in this manner since every node is broadcasting their data at the same frequency. Every node is assigned a timeslot when it has the permission to broadcast data over the medium. Figure 2.1 shows an example of how it can look like in a TDMA network.

A TDMA network is well suited for networks where the dataflow is regular and large bursts of data are uncommon. If all the timeslots are used all of the time we will get the most out of a TDMA network. If for instance most of the timeslots are not used and suddenly a node wants to transmit a huge burst of data it will not be granted access to more timeslots and will therefore have to wait and patiently broadcast the data at the same bit rate as previously offered.

A clean and isolated frequency band is preferred for TDMA networks since it is sensitive to other transmitters in the frequencies around the used frequency band. This attribute makes TDMA-networks less preferable in unlicensed frequency bands according to [2].

![Figure 2.1 Example of how a TDMA network can look like.](image)
Outlines for a general TDMA-network

- Packet collisions are avoided since every node knows when and for how long it is allowed to access the channel.
- A global time reference for all the nodes in the network is needed, otherwise the nodes have no chance to know when it is their actual timeslot.

2.2 Real-Time Systems

Real-time systems are systems where the correctness depends not only on whether the computations are correct, but also on when in time the computations are completed [3]. The interaction between the real-time system and the environment is more or less time critical. If a system has a deadline which has to be met strictly, it is said to have a hard deadline. If a deadline requirement is not met in this case, it may lead to a catastrophic result. If a computation is delivered after deadline in a soft real-time system, the values relevance only degrades with time [4].

2.3 Distributed Clock Synchronization

There exist many different clock synchronization algorithms for real-time distributed systems. The synchronization algorithm used in our work is an algorithm adapted to the system architecture. It was inspired by The Precision Time Protocol and is discussed below [5].

2.4 The Precision Time Protocol

The precision time protocol provides a way to synchronize the clocks in distributed systems down to a precision of less than one microsecond. It is built around a master which is said to be the provider of global time and slaves which synchronizes according to the masters clock [6].

![Figure 2.2 Workflow for clock synchronization.](image)

1. The master sends a synch message at time T1.
2. Afterwards it sends a follow up message with the time when T1 was sent.
3. The sync message arrives at time T1' at the slave.
4. The slave then sends a message at time T2.
5. The message arrives at T2' at master.
6. The master then sends the value T2' to the slave.

After these steps the slave has the timestamps T1, T1', T2 and T2' and will now be able to synchronize its clock. A communication delay exists since the message sent at time T1 cannot arrive at the same time on the other end of the communication line. The delay has to be estimated. Combining Equation (2.1) and Equation (2.2) gives Equation (2.3) which lets us estimate the communication delay $\sigma$. $\Omega_{MS}$ is the offset between the slave’s clock and the master’s clock.
The slave clock can then be updated by subtracting the master slave offset according to Equation (2.4).

\[
T_1' - T_1 = \Omega_{MS} + \sigma \\
T_2' - T_2 = -\Omega_{MS} + \sigma \\
\frac{t_1' - t_1 + t_2' - t_2}{2} = \sigma
\]

The sync message containing T1 is usually sent every 2 seconds. This is done since the different clocks will drift apart as a function of time. Depending on the requirements of the specific network the resynchronization is done more often if the accepted deviation between the clocks is more strictly specified.

A new estimate of the delay is performed at greater intervals. This depends on how much the communication delay may deviate as a function of time, this can happen when switches and routers are in between a master and slave for example, or if a certain path becomes inaccessible and another route between the master and slave has to be found.

### 2.5 Clock Drifting

A computer clock running at a certain frequency can most often not be guaranteed to run at that exact frequency since external factors such as temperature variations have an influence on the frequency of the clock. This means that two crystal clocks potentially never run at exactly the same frequency which will make the clocks drift apart from each other. Let us say that C is the current time shown by the clock and that this time is a function of the real time t. In a perfect world the current time shown would be linear to the real time, which would give a derivative of 1, see Figure 2.3. This would be a perfect clock. But in reality, as shown in Equation (2.5) a maximum drift rate, that is specified by the manufacturer, exists. This means that 2 different processors could according to Equation (2.6) have drifted apart maximally 2 specified drift rates times the time interval from synchronization. Let us define a maximum acceptable difference between two clocks according to Equation (2.7). Then after some rearrangement which yields Equation (2.8), we see that depending on the requirements of T we will have to synchronize at smaller \( \Delta t \) to still fulfill the requirements on T since \( \rho \) is defined by the manufacturer.

\[
\frac{ac}{at} = 1 \pm \rho \\
S = 2\rho \times \Delta t \\
S_{max} = \Pi \\
\Delta t \leq \frac{\Pi}{2\rho}
\]
A TDMA Module for Waterborne Communication with Focus on Clock Synchronization
Chapter 3
Hardware and Software

This chapter provides a brief description of the hardware and software used during the development of the TDMA network protocol.

3.1 Sam4s Xplained Evaluation Kit

The Sam4s Xplained from Atmel is a kit used for evaluating the capabilities of the processor AT91SAM4S16C for different applications [7]. The board is powered by the USB cable and can easily be programmed using the integrated JTAG emulator. The JTAG emulator can also be used for debugging the code when running the program in real-time.

Figure 3.1 Sam4s Xplained overview.

Downloaded from:
3.2 AM335x Starter Kit

The AM335x starter kit from Texas Instruments is an embedded system that allows you to choose between an Android or a Linux based operating system [8]. Figure 3.2 shows how the AM335x looks like unpacked. Figure 3.3 shows the Linux based operating systems which you navigate by using the 4.3” resistive touch LCD.

3.2.1 Hardware

- Sitara™ AM3358 Cortex™-A8 processor, up to 720 MHz
- TPS65910 power management I/C
- WL1271 Wi-Fi/Bluetooth Module
- 4.3" resistive touch LCD
- 256MB DDR3
- 2-port Gb Ethernet switch
- XDS100 USB JTAG

Figure 3.2 - AM335x Starter Kit from Texas Instruments
Downloaded from: http://processors.wiki.ti.com/index.php/AM335x_Starter_Kit (2013-06-20)

Figure 3.3 - The generic menu view from AM335x Starter Kit.

Downloaded from: http://processors.wiki.ti.com/index.php/AM335x_Starter_Kit (2013-06-20)
3.3 Atmel Studio 6

Atmel studio 6 is an integrated development platform for developing and debugging Atmel ARM microcontroller based applications [9]. The Atmel studio 6 is integrated with the Atmel Software Framework, which contains free source code with lots of examples which easily can be adapted to your projects, they are also useful to understand how different microcontroller peripherals are working. The Atmel Studio 6 also contains a great debugger to easily test programs, place breakpoints etc. If a JTAG is connected between a microcontroller and the PC with Atmel Studio 6, the program offers real-time debugging while running the program. Figure 4.1 shows how the integrated development environment for Windows looks like.

![Figure 3.4- Example of how the enviroment for Atmel 6 looks like.](image)

3.4 GNU Emacs

GNU Emacs is a customizable text editor that can be used to write for example C-code [10]. Emacs has a lot of hotkeys and is often preferred for writing programming code. Many people claim that GNU Emacs is more like an operating system than it is a text editor. The ways to customize Emacs to the specific way you want it, makes it very popular, and it also makes Emacs to a worthy challenger to the many integrated development environments available on the market.
Chapter 4
System Design

This chapter explains how the system was designed with the characteristics of the specific network in mind. Arguments are provided to clarify why certain decisions are made.

4.1 Brief Explanation of the Architecture

The dive-console is built around two different platforms which are communicating via USB as shown in Figure 4.1. The MAC protocol is implemented on the Sam4s platform while the AM335x is handling all the demanding computations.

![Figure 4.1 - The two platforms communicate via USB.](image)

4.2 Characteristics of the Network

The water based network consists of 10 nodes with half-duplex communication, which means that a node can send and receive data but not at the same time. When one node is broadcasting data all other nodes are listening for incoming data, as illustrated in Figure 4.2. Since the data is broadcasted at the same frequency a protocol is needed to control when medium access should be granted. The different nodes have a distributed responsibility to make the network reliable.

![Figure 4.2 - Half-duplex communication in the network.](image)
4.3 The TDMA MAC Protocol

The TDMA MAC protocol for the dive-console is a regular TDMA protocol where nodes broadcast their data in different timeslots. For the TDMA network to function properly a global time reference is needed by all the nodes, according to Figure 4.3. If no global time reference is available the network breaks down.

![Diagram of a TDMA network with a master and slaves](image)

**Figure 4.3 -** The master provides a global time to the slaves in the network.

4.3.1 Unique Conditions for the TDMA Network

The conditions for this network are not like an ordinary TDMA sensor network where each node gets a global time reference from a GPS service or equivalent. Since radio and GPS communication do not work well in water, ultrasonic communication is a better alternative. However, the propagation speed for sonic waves in water only is 1500 m/s the communication has to be adapted to these conditions. Another unique property that the network has to take care of is the lack of information regarding if the node is above or under water. Since the velocity is very different in air and water for ultrasonic waves [11] the protocol assumes that the unit is under water. This will introduce an offset error between the nodes. The error will be wiped out as soon as synchronization is performed with both nodes below sea level.

The number of nodes in the network is not constant and each node is in motion. This results in a need of an adaptable clock synchronization protocol that are able to handle if nodes join or leaves the network in real time. A balance between the time it takes to join the network and how much data a node can send per second has to be taken into consideration. This is explained in depth in Section 4.5.1. A master-slave method is a great solution which only limits the network to work when a master is present in the network. Preferably when a group at a scuba diving center is scuba diving the instructor uses the master unit, since the instructor has to be present under water when he instructs his class.

With these characteristics in mind, the clock synchronization protocol was developed.
4.3.2 The TDMA Network Structure

A well-defined structure explains when different nodes in the TDMA-network will have their chance to broadcast data over the network. We propose that a frame is divided into 12 timeslots as illustrated in Figure 4.4. The first two timeslots (TS0-TS1) are special slots used for the clock synchronization between the master and the slaves, the other 10 timeslots are each assigned to one user in the network. 10 frames form one superframe as depicted in Figure 4.5, and two superframes form a megaframe as shown in Figure 4.6.

![Figure 4.4](image)

**Figure 4.4** - One frame divided into 12 timeslots.

![Figure 4.5](image)

**Figure 4.5** - One superframe divided into 10 frames.

![Figure 4.6](image)

**Figure 4.6** - One megaframe divided into ten superframes.

4.3.3 The Timeslot Size

The timeslot size is set to 140 milliseconds to be able to handle distances up to 100 meters between the nodes in the network. 100 meters is specified as the requirement from Didamus. The maximum time of flight is approximately 66.7 milliseconds. In Equation (4.1) this is denoted with $T_{MAX}$ where the numerator is the distance and the denominator is the speed of sound in water [11]. The broadcast time for a message is 65 milliseconds, the broadcast time was specified by Didamus. In Equation (4.2), this is denoted with $T_{BROADCAST}$. The total time before a complete message arrives is denoted with $T_{TOT}$ which is the maximum travel time added up with the broadcast time as shown in Equation (4.2). A guard time is used in case of an offset error between the nodes global time reference. In Equation (4.3), $T_{GUARD}$ stands for 8.3 milliseconds. $T_{SAFE}$ is the complete timeslot size with $T_{TOT}$ and $T_{GUARD}$ summed up together.

\[
T_{MAX} = \frac{100}{1500} s \approx 66.7 \text{ ms} \quad (4.1)
\]

\[
T_{TOT} = T_{MAX} + T_{BROADCAST} = 131.7 \text{ ms} \quad (4.2)
\]

\[
T_{SAFE} = T_{TOT} + T_{GUARD} = 140 \text{ ms} \quad (4.3)
\]
Figure 4.7 shows a scenario where the distance between node 1 and node 2 is 100 meters and between node 2 and 3 is 0 meters, the maximum distance the network has to handle and 2 nodes very close to each other. In this scenario node 1’s message sent at timeslot 1 will arrive at node 2 and node 3 after approximately 131.7 milliseconds, then 0.3 milliseconds later it is time for node 2 to broadcast his message. Due to the carefulness in timeslot size, the last part of node 1’s broadcast and the first part of node 2’s broadcast will not superimpose each other, which also is shown in Figure 4.8.

4.4 TDMA and Clock Synchronization

The TDMA protocol needs a common time reference shared between all the nodes in the network to function properly. The precision time protocol offers extremely accurate clock synchronization but its efficiency would be poor if the signal processing delay deviates too much from one sent message to another, since new estimates of the signal processing delay would be needed often, this would cause critical redundancy to the network. Especially since the signal speed is slow, too many timeslots occupied for clock synchronization purposes, would give an extremely poor network bandwidth.

According to Figure 4.9 the signal processing for the decoding module takes approximately 100 milliseconds and may deviate with as much as 20 milliseconds. This was measured with the help of 2 nodes sending messages to each other, timestamps was logged to calculate the round-trip time.

To avoid the deviation in signal processing and to give a better bandwidth for the network two special synchronization messages were used, which is explained in detail in section 4.4.1.

4.4.1 The Different Messages

To be able to identify a message in an efficient manner, which is to avoid deviation in signal processing delay, two new message types were implemented, a slave synch message and a master synch message. The efficiency lies in the way to decode a message without a deviating signal processing delay.
These new messages only need to be sampled by the analog to digital converter in order to be identified, which makes the signal processing delay approximately constant. Figure 4.10 shows the structure of the different messages. A probe is commonly used to tell the receiver that a message is incoming, a predetermined pause time is then followed by the actual message which is shown in Figure 4.10.

![Diagram](image)

**Figure 4.10** - Different messages that the unit can send.

### 4.5 The Time Synchronization Protocol

#### 4.5.1 The Startup Phase

When the units are booted, no global time is available for the units, thus no unit can begin to send, except the master, since his local clock is the global clock that the slaves will adjust their local clocks to. Therefore the master starts sending a special message that the slaves can identify, which asks if somebody wants to join the network, this message is always sent at some specific instant of time, according to Figure 4.11.

![Diagram](image)

**Figure 4.11** - When the master asks if somebody wants to join the network.
4.5.2 The Slave Answering Phase

When a slave recognizes that a master synch message has been received, the slave will send a slave synch message to the master at a unique dedicated timeslot and frameslot. The time the slave will wait will differ, depending on what slave id the unit got, according to Figure 4.12 and Equation (4.4). The subtraction of the processing delay is to compensate for the time it takes to process the master synch message.

\[ T_{\text{Wait}} = (\text{FrameSize} \times \text{SlaveID}) - T_{\text{ProcessingDelay}} \]  

(4.4)

**Figure 4.12** - Shows when the master will send a message to ask if somebody wants to join the network and when each slave will answer.

4.5.3 Distance Calculation

When the master receives a slave synch message it will timestamp it and save the value in an array, as depicted in Figure 4.13. The saved timestamps can then be used to calculate the distance to respective slave which Equation (4.5) clarifies. Here, \( \Phi_x \) is the time difference between the start of frame \( F_i \) and when the actual slave synch message arrived at the master, as Figure 4.14 shows. \( \Lambda \) is the speed of sound in water. It is divided by two because \( \Phi_x \) is the round trip delay. The different messages include no particular information, the ability to identify different messages and perform actions depending on the type of messages is enough to be able to calculate the distance with sufficiently high precision.

\[ \text{SlaveDistance} = \frac{\Phi_x \times \Lambda}{2} \text{ m} \]  

(4.5)

**Figure 4.13** - Array containing timestamps.  
**Figure 4.14** - A varying delay depending on the distance between the nodes will exist.
The distance calculation step by step:

1. The master sends a master synch message, as illustrated in Figure 4.15, at time \( \tau \).
2. Slave \( S_i \) receives the master synch message somewhere in Frame 0, depending on the distance \( \rho \) between the specific slave and the master, as illustrated in Figure 4.15.
3. Slave \( S_i \) then answers after a number of frames depending on its unique slave ID at time \( \chi_i \) as illustrated in Figure 4.15.
4. Finally the answer from the slave to the master arrives at time \( \Psi_i \), depending on which frame the answer came in, the master knows which slave the actual message belongs to.
5. The value of \( \phi_i \) is obtained by measuring the time from the start of frame \( F_i \) and when the actual message arrived at the master, which actually is \( \Psi_i + \rho_i \) as shown in Figure 4.15, which will be the round trip delay subtracted with a number of frames depending on the slave id.

![Figure 4.15](image)

**Figure 4.15** - Shows an example of when slave 1 and 2 will receive and transmit synch messages, depending on the distance.

### 4.5.4 Slave Answers Received

By the time all 9 slaves have had a chance to answer according to Figure 4.16, the master will send the array to the target machine, which is the Linux platform referred in section 3.2, for distance calculation. When the platform is ready it will immediately send it down to the MCU platform mentioned in section 3.1, which Figure 4.17 describes. The MCU platform then sends the synch packet at time \( \tau_4 \) which is equal to the start of the second super frame.

![Figure 4.16](image)

**Figure 4.16** - At time \( \tau_1 \) all slaves have had a chance to answer, and at time \( \tau_4 \) the master will broadcast a sync packet over the network.
4.5.5 Sync Packet Arrives

The sync packet will have a special header so the slaves are able to identify if a sync packet has arrived. Every slave unit has a special user-id which is used to know which index in the array that is their actual distance. After the slave have estimated how many clock ticks it should compensate for to be in synch with the master, it will send the value to the MCU, according to Figure 4.18.

4.5.6 The Clock Adjustment

Every time a slave receives a master synch message it sets its local clock to TS0, F0 and SF0 if it is not synchronized yet. At this moment an offset error exist which is equal to the distance between the nodes, which Equation (4.6) and Figure 4.19 clarifies.

\[ Slave_{CLOCK} - Offset_{ERROR} = Master_{CLOCK} \]  \hspace{1cm} (4.6)
After a master synch message has arrived the slave is not fully synchronized with the master, one last important detail is missing, the distance between the master and the slave. When a slave recognizes that a message containing the distance to the master has finally arrived the slave will adjust its clock and finally be synchronized with the master, according to Figure 4.20 and Equation (4.7). The slave will then get access to the medium at his unique timeslot.

\[
\text{Slave}_{\text{Clock}} - \text{Offset}_{\text{ERROR}} + \text{Offset}_{\text{ERROR}} = \text{Master}_{\text{Clock}}
\]  

(4.7)

**Figure 4.20** - When the distance to the master is known the slave adjust its clock and is finally synchronized with the master.

### 4.5.7 A Bound Medium Access Time for the Slaves

After a slave has successfully synchronized with the master it will only be granted access to the medium for a limited amount of time, more precisely 70 seconds. This is due to some error which might have been introduced during the synchronization phase. Such as incorrect distance, false synch messages etc.

If a node with an inaccurate clock is broadcasting over the network it might overlap with another node’s timeslot and introduce errors in the network communication. But if a node is only synchronized for a limited amount of time the damage will not be fatal and the unit will resynchronize with hopefully a better outcome.

To be more precise a unit will be able to send data over the network approximately 41 times before it gets ordered to resynchronize. Since the synchronization messages are broadcasted continuously the distance between the master and slaves will be updated each megaframe period. Since the latest most up to date distance to the master probably is available for the slave it will resynchronize as soon as a new master synch message arrives.

A limited medium access time for the slave is also used to avoid the phenomena called clock drifting.
Chapter 5

Results and Experiments

In this chapter results and experiments are presented.

5.1 Recognition of Different Messages

Tests were made concerning the recognition of the master synch message and slave synch message, since these messages contained 10 and 5 probes, respectively, the limits had to be tuned to give a as high probability of correct detection as possible. After the tuning, the units seemed to decode correct messages most of the time. A wider testing with logging performed during a real dive trip would have been more realistic to confirm the functionality of the decoding of the messages. Due to a limited amount of time no tests of this kind were performed during the thesis.

5.2 Tests of a varying delay

A loop-back message was sent between two different units to see if the time of arrival of the message was varying between each sending. The tests performed showed that the variation was at an acceptable level which was about one to two milliseconds. It is acceptable since the timeslots are 140 milliseconds long and with this variation a relatively accurate distance could be calculated.

5.3 Estimation of Processing Delay Time

To estimate how long it took to process a master probe message and a slave probe message two nodes were placed with a known distance between them. Then a master probe message was sent from the master at its given timeslot according to the TDMA structure. The slave then answers at its dedicated timeslot. When the slave probe message arrives at the master, it saves a timestamp which can be used to calculate the processing time according to Equation (5.1). When an answer arrives, a delay exists depending on the distance between the nodes, but since the nodes are placed at a known distance, the distance delay is constant. After compensation of the distance, a new equation can be used to determine the actual processing time, since the master probe message is twice the size of a slave probe message, Equation (5.2) and Equation (5.3) gives the processing delay time for the master and the slave.

\[
T_{\text{timestamp}} = T_{\text{processing}} + T_{\text{distance}} \tag{5.1}
\]

\[
T_{\text{master}} = 66,66\% \times T_{\text{processing}} \tag{5.2}
\]

\[
T_{\text{slave}} = 33,33\% \times T_{\text{processing}} \tag{5.3}
\]
5.4 Tests of clock drifting

A few tests were performed to see if the messages sent arrived at approximately the same time even after running for a few minutes. Time stamps that belonged to each message were analyzed and a noticeable change were never detected. Although a more realistic test in water would have been preferred since the temperature in water can vary with various depths, and a change in temperature will change the frequency of the embedded systems clock [12].

5.5 Tests below and over sea-level

When the development was ongoing, tests were performed in the sea with two scuba divers to verify if the communication also worked below sea-level. The tests outcome was positive and it seemed to work well. Due to the fact that only two functional units existed, when the deadline was reached for the thesis the functionality of the TDMA-module could not be validated with more than two units, however in the moment of this writing, tests have been done with three units properly synchronizing with each other with an error of about 1 millisecond at most.
Chapter 6

Discussion

This chapter provides a few insightful thoughts about the thesis.

6.1 A Valiant Attempt to Implement the TDMA-network in higher Network Layers

When the need for a network driver concerning the medium access control was brought up from the company Didamus, the first idea was to make a kernel driver for Linux taking care of the sending and receiving packets for the network. But since the whole system was built in user-space and an event-handler was used to communicate between different layers of the network, this solution were never driven really far.

Instead the TDMA-protocol was implemented on the AM335x Starter kit platform. The general thoughts about this were that it would never be a problem even if the part controlling the medium access was higher up in network layers. Besides the AM335x processor was far more powerful than the ARM processor on the Atmel evaluation board. The ARM processor was already handling a couple of sensors, Analog to Digital Sampling and down-conversion of the signal. The guidelines were to not expose the ARM platform for more load than needed, with the risk of getting overburdened. Figure 5.8 shows where the TDMA-protocol was intended to be placed. After the implementation was done, tests were made to see if the delay was approximately constant. The problem here was that the delay was varying with about 10 to 20 milliseconds. The largest impact on the varying delay came from the decoding module which uses complex and CPU-demanding algorithms to compensate for bounces if the ultrasonic wave hits something on the way. An increased guard time between timeslots could have solved the problem, but since the velocity in water for ultrasonic waves is only 1500 m/s an inefficient TDMA-protocol would have a large negative impact on the unit as a whole. Another important thing to take into consideration is that a general Linux OS with multithreaded programs is not the ideal place to implement real-time modules on, since no guarantee exists concerning when a specific thread will get execution time.

After some deliberation, the TDMA MAC protocol and TDMA time-synchronization protocol were moved to the evaluation platform from Atmel where no operating system existed.

![Figure 5.8 TDMA Implementation at first try.](image-url)
6.2 Testing the Functionality

To test if the TDMA network is working as intended, tests can be performed for several hours collecting data of when messages arrives to different nodes in the network. Every message that arrives at a node becomes time stamped. This logged data can later be analyzed with the help of a computer. Unfortunately, when the thesis at the company finished, there were only 2 nodes available for testing purposes.

6.3 Errors caused by movement

Let us take an example. Let us say that on average a scuba diver is moving with the speed of 0.44 m/s. This varies of course depending on the specific scuba diver. So let us say that two different scuba divers are swimming in opposite directions in the water. Imagine that one has the master unit and the other one a slave unit. The master starts to broadcast a master synch message followed by an answer from the slave. The slave will then receive the global time reference after 16.8 seconds, which is at the start of superframe one when all slaves have had a chance to answer. The largest error can occur between the master and the slave that answers last of all slaves. The time between these messages are approximately 15.1 seconds which is shown in Equation (6.1). This would introduce an error to the estimated distance with a worst case error according to Equation (6.2). The two in the equation appears since they swim in the opposite direction. The extra guard time needed in this scenario is given by Equation (6.3). The additional guard time between timeslots is needed to prevent this error from ruining the TDMA network when this scenario arises.

\[ T_{\text{delay, max}} = T_{\text{super, frame}} - T_{\text{frame}} = 16 - 1.68 s \approx 15.1 s \quad (6.1) \]

\[ \text{Distance}_{\text{error}} = 15.1 \times 0.44 \times 2 \text{ m} \approx 13.2 \text{ m} \quad (6.2) \]

\[ T_{\text{guard}} = \frac{13.2}{1500} \approx 9 \text{ ms} \quad (6.3) \]

6.4 Different conditions when used under water

There is a slightly difference of the velocity of sound in water which depends on if it is fresh- or saltwater [13]. This has to be taken into consideration when the guard-time between time slots is estimated. There also exists salt-layers where the concentration of salt is very high, this will probably have a great impact on the velocity [14]. Tests should be performed to test the ability of the sound-wave to penetrate through a realistic salt-layer. The speed of sound in water is also dependent on the current depth it propagates within [15]. But down to about 50 meters depths it’s approximately constant [15] and the unit only got the requirement to work down to 40 meters depth, this property can hence be ignored.
References


(Taken 28-05-2013).

(Taken 07-06-2013).

http://www.seas.upenn.edu/~lee/10cis541/lecs/kopetz_chap_1and2-ilx2.pdf
(Taken 15-06-2013).

(Taken 15-07-2013).

[6] Introduction to Distributed Clock Synchronization and the IEEE 1588 Precision Time Protocol.
(Taken 25-07-2013).

[7] Sam4s Xplained.
http://www.atmel.com/tools/SAM4S-XPLD.aspx
(Taken 25-06-2013).
http://processors.wiki.ti.com/index.php/AM335x_Starter_Kit#What_is_the_AM335x_Starter_Kit
(Taken 24-06-2013).

http://www.atmel.com/tools/ATMELSTUDIO.aspx
(Taken 08-08-2013).

http://www.gnu.org/software/emacs/
(Taken 24-06-2013).

http://physics.info/sound/
(Taken 13-05-2013).

(Taken 07-07-2013).

(Taken 20-07-2013).

http://www.oregon.gov/dsl/ssnerr/docs/efs/efs33mix.pdf
(Taken 08-07-2013).

http://www.arc.id.au/UWAcoustics.html
(Taken 06-06-2013).
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Uphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Anders Persson