Collision Free Protocol for Ultrawideband Links in Distributed Satellite Avionics

Tobias Mikschl [(1)], Richard Rauscher [(1)], Sergio Montenegro [(1)], Klaus Schilling [(2)], Florian Kempf [(2)], Tristan Tzschichholz [(3)]

[(1)] Dep. of CS VIII: Aerospace Information Technology
University of Würzburg, D-97074 Würzburg, Germany
Email: tobias.mikschl@uni-wuerzburg.de
Telephone: +49 931 31 80031
[(2)] Dep. of CS VII: Robotics and Telematics
University of Würzburg, D-97074 Würzburg, Germany
[(3)] Zentrum für Telematik e.V.
D-97218 Gerbrunn, Germany

PAPER

We are working on a distributed and modular approach for wireless connected satellite avionics. In this paper, we want to present a simple, yet powerful protocol for Ultrawideband links, which was designed for intra-satellite communication, connecting computer, sensor and actuator nodes. The suggested protocol provides reliable communication and a fixed data-rate for every connected node, while still being re-configurable on the fly and reacting dynamically on connecting / disconnecting network members.

1 INTRODUCTION

In our project YETE we are working on a innovative solution for distributed data processing and control in fractionated spacecraft [1].

A common approach for on-board data handling (OBDH) in modern spacecraft is to use several specialized subsystem computers in parallel for the individual tasks, i.e. for sensor data post-processing and to hardwire the communication network of the individual subsystems of the spacecraft. These subsystems are then controlled by one or more redundant general purpose computing units (GPUs).

One drawback of this centralized approach is that computing resources of the specialized subsystem computers cannot be shared among other subsystems or other spacecraft in a mission, which results in wasted computing resources. Furthermore should all GPUs fail, all still working spacecraft subsystems are lost for the mission.

We are addressing these drawbacks by a distributed data processing concept with strong emphasis on modularity at hardware and software level.
On the hardware level all spacecraft subsystems (sensors, actuators, computing units, etc.) are treated as independent nodes in a distributed network. Most device specific computations are performed on general purpose computer nodes, which form a computing cluster.

On the software level all functional software units, i.e. I/O drivers or applications, are encapsulated into independent "Building Blocks (BB)". They can easily be added or removed allowing fast re-configuration of the software system to changing mission conditions. Intra-/inter vehicle communication, task distribution and task execution is handled by the middleware OS RODOS which runs natively on all nodes in the spacecraft.

Our system has been thoroughly tested in a network of sensors, actuators and computer nodes connected via CAN (Controller Area network). Now we are taking the next step to replace all connections within our system with low power short range wireless links. Later in the project we will add inter-satellite links, that will allow the space vehicle to use the computing resources, sensors and actuators of other space vehicles and to share its own. See figure 1 for a sample yete configuration.

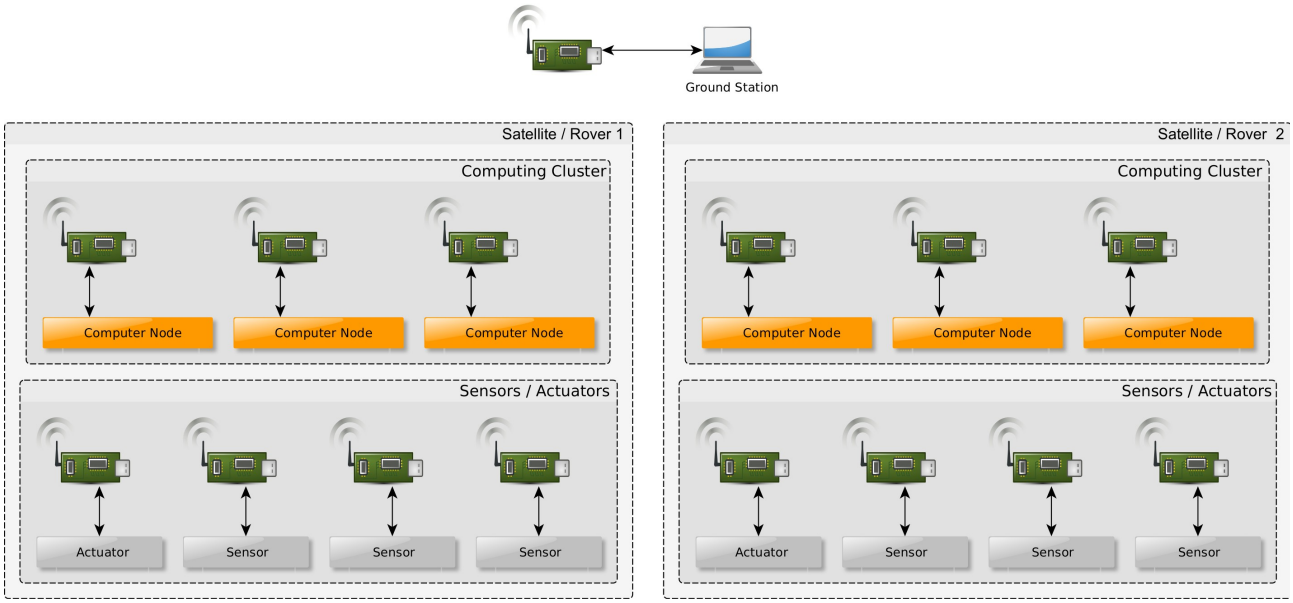


Figure 1. The YETE concept: Wireless links are used to connect a cluster of computing nodes to very simple sensors/ actuators. All sensor- or actuator-specific processing now takes place in one or more computing nodes.

2 ULTRAWIDEBAND

After studying several possibilities, we decided to use Ultrawideband (UWB) for short range intra-satellite communication.

UWB was standardized under IEEE 802.15.5 in the year 2011 [2] and brings several benefits compared to traditional narrow-band radio communication:

- ability to deal with severe multi-path environments
- low spectral density and therefor less disturbance for EM sensible devices
- robust to narrow-band disturbances

It has already been shown, that UWB looks like a promising and viable candidate for short-range radio communication in satellites [3].

On the hardware side we use *DW 1000* UWB modules by the Irish company *Decawave*. Mainly developed for local positioning systems, these modules are perfectly suited for data communication as well. Specifications include [4]:

- IEEE802.15.4-2011 UWB compliant
- maximum Data rate 6.8 Mbps
- Low power consumption

While we use these modules to develop our communication protocol, we implement it as hardware independent as possible.

3 THE PROTOCOL

Based on the Ultrawideband physical layer provided by the commercial of the shelf (COTS) modules, we implemented our own communication protocol.

Requirements for the protocol stack are:

- no separate time synchronization needed
- self initializing
- reasonable data rates
- dynamic detection of packet loss
- modular / highly re-configurable

The protocol features a slot based system, which avoids collisions in the network and ensures a fixed data rate for every network member. Initialization of the protocol is CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) based, as for the normal operating mode in TDMA (Time Division Multiple Access) first a synchronization of the connected nodes has to be established.

In figure 2 graphical illustrations are shown for the individual parts and structures. This will help to understand the explanatory graphics for the different phases of the protocol.

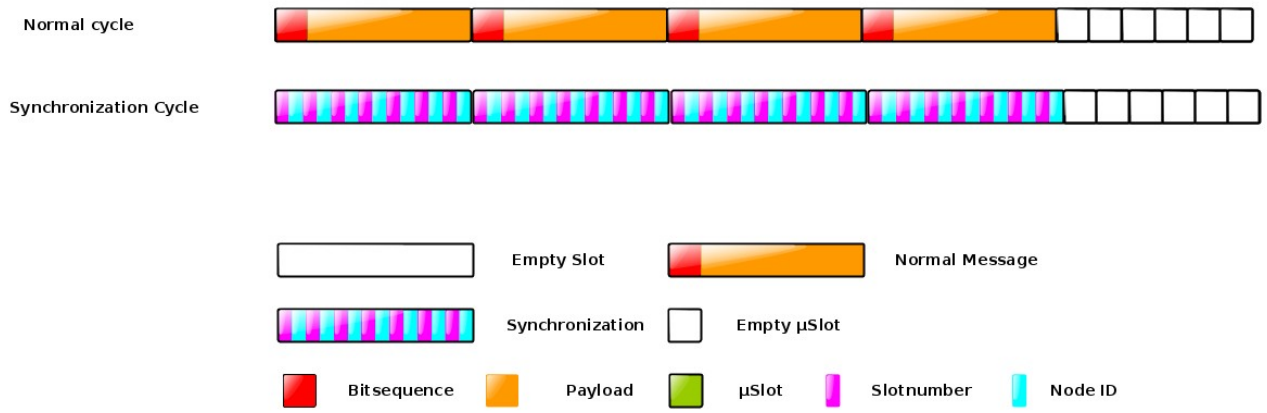


Figure 2. Graphical illustration of the different components of the protocol.

The network can be in one of three phases. Either the Hello cycle, reordering cycle or in the normal cycle (see figure 3). After the reordering and normal cycle so called µSlots are added, which allow registration of new nodes in an already established network.

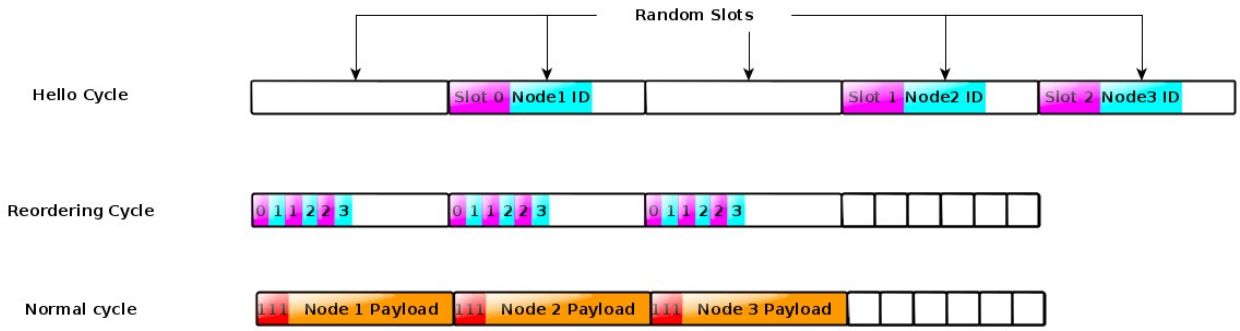


Figure 3. Different cycles of the protocol.

3.1 Simplified protocol sequence

In the beginning the protocol starts with the Init Phase. If there is no network, then the nodes discover its neighbors and form a new one. The access to the media is similar to CSMA/CA.

The Sync Phase is started after the formation of the new network, in case of a node's failure or when a of a new device registers itself via the µSlots. In this stage, the nodes synchronize its local lists with the known neighbors and assigned slots. This list serves each node as schedule. The used technique to access the media in the Sync and Data Phase is TDMA in combination with a dynamic slot and frame size.

The exchange of data is the objective of the Data Phase and induced after the Sync Phase. In the Data Phase, the header contains acknowledgements to every message. These are necessary to detect the failure of a node. In this case, the node and its assigned slot is removed from the schedule so that the throughput is increased, and the delay is decreased. Afterwards, the known nodes must synchronize its local lists. A registration in the Data Phase is possible analog to the Sync Phase via µSlots.

The registration of a failed or new node takes place via the µSlots. At the end of the frame, there is an empty slot which is divided further in smaller slots. These µSlots help to reduce collisions if more than one node wants to register at the network. The specific µSlot is chosen randomly by the new node. A registration leads to a followed Sync Phase in which a new slot is added to the frame and the new schedule is distributed.

3.2 Initialization phase - the Hello cycle

The starting point is the Init Phase and there are four challenges, namely the decrease of collisions, the election of the first node which starts the next phase, a mechanism if the first node fails and a stable state as fast as possible. The Init Phase is used to form the network in the case that all nodes fail or a new network is started. At the beginning, the nodes do not know each other and communication to discover the network is necessary. However, the nodes are not ordered and unknown, so collisions can not be excluded. However, with the technique of CSMA/CA the number can be reduced.

Only exploration packets shifted by random listening times are send. This exploration packets contain the sending nodes ID and between these the node is listening for exploration packages of foreign nodes. As the sending times are distributed with randomly varying listening time in between, sooner or later two different node will receive each other, if they are in range.

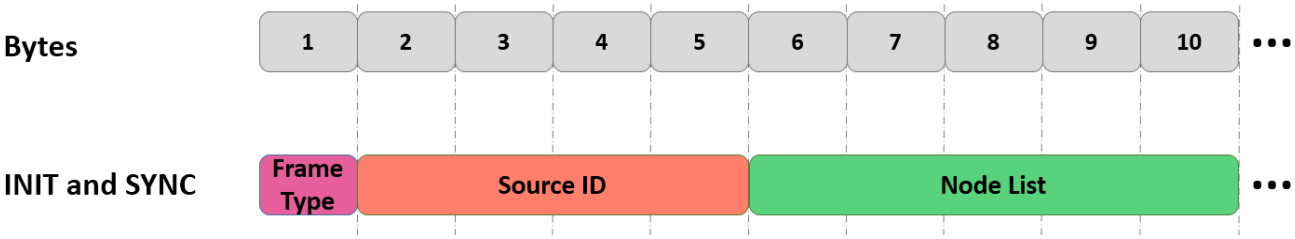


Figure 4. Protocol header of the Init Phase.

Figure 4 shows the header of the protocol in the Init Phase. The first byte is the frame type, which is in accordance with the different stages of the protocol. The different types are Init, Sync, Data and Micro. In the initial stage, the type of a received message determines the further action. The type is followed by the id of the message source. Finally, a list with the known neighbor nodes is contained. This list is essential for discovery and synchronization.

3.3 Sync phase - Reordering cycle

The Sync Phase aims at the synchronization of the node lists of all registered devices. This list is used as schedule for the TDMA and every node can determine its slot from this schedule. The header of the Sync Frames are identically with the Init Frame apart from the frame type. The Sync Phase is started after the Init Phase or when changes in the list occur due to failed or new nodes.

In the Init Phase the first node is determined, called Leadoff, which starts the Sync Phase. In the case that all nodes but the Leadoff fail, the Init Phase is restarted. However, if the first node fails, the other ones remain in the Init Phase since no stable state was reached so far.

If a Sync Message is received, the contained node list overwrites the local list. With the overwriting a global consistent list state on every active node is reached. The next slot node redistributes the received list because of two reasons. The first one is, that the reception of the list is guaranteed in case that several nodes did not received the Sync Frame. A second reason is that new arrived or failed nodes receive the list and can determine the µSlots. Because the list contains the slot numbers, the unregistered node waits until the last slot. After them, the empty slot with the several µSlots is appended. Then the node can register and a further Sync Phase is started.

The timers of the nodes are synchronized at the reception of a message. This means as soon as a message is received, the node determines the last slot number and computes the number of remaining slots until its own. In Figure 4, Node i sends the message in Slot i. Node i + 1 sends its message as soon as the message of Node i is received. So the slots are compressed and the successor does not wait until the end of the current slot. In the case of an later node as shown in b), Node i + j shifts the beginning of its frame forward by the remaining slot time. Also, the reception of a message is the reference point to adapt the nodes timer. This is also valid in case of the µSlots, whose beginning is also set earlier. With these dynamic size, the slot is as large as necessary and the throughput is increased. Simultaneously, the logical order is respected. If nodes fail, the slot remains empty and the following nodes do not change its beginning.

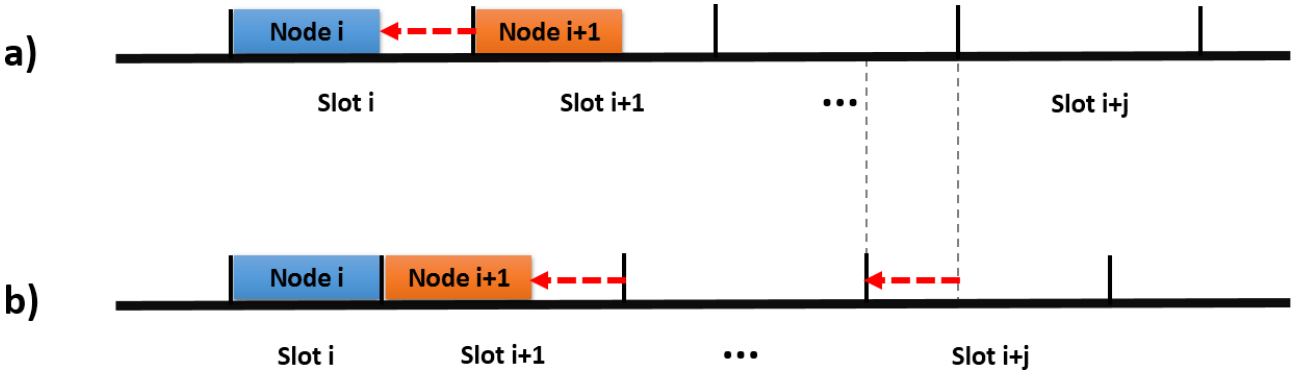


Figure 4. Visualization of the slot shift to increase throughput.

However, the distributed application of the protocol hinders the backup of the first node. Because if a node changes its list, it is possible that the modification is only local e.g. this node can be the only one, which did not receive the message. Therefore, the device is in an unstable state and has to wait for the synchronization. In the next frame, a Sync Phase is expected to ensure that all nodes share the same node list. Otherwise, if the next frame is not a Sync Phase the node deletes its list and has to reregister via µSlot, because only its list has changed. This leads to a Sync Phase, which synchronizes the node lists. Because of the design, the nodes have to wait until the first node starts the Sync Phase. Alternatively, the node with the changed list can start the Sync Phase. However, the Sync Phase starts in the midst of a frame and if the µSlot is used additionally, a further Sync period has to start. In the case of a dedicated sync frame, the µSlot registration and the node list change can be handled at once.

After the Sync Phase the next stage starts, provided that no node uses the µSlot. In the next phase, Data Frames are sent.

3.4 Data phase - Nominal (synchronized) operation

In the Data Phase, the goal is the exchange of packets with data and as keep-alive packets via broadcast to recognize nodes, which failed. A challenge in this phase is the recognition of a downtime of a node. Because there is no central device, each node has to detect a failure of a node by itself. However, a node which failed has to be distinguished from a message, which only a single node did not receive. Furthermore, the failure of the Leadoff must not lead to a breakdown of the network. The last challenge is finding the right time for updating the local neighbor list.

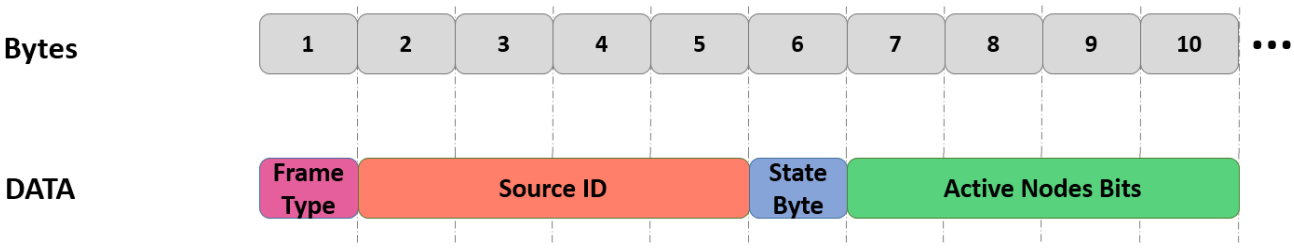


Figure 5. Protocol header of the Data Phase.

The header of the Data Frame contains like the other ones, the frame type and the source id as in Figure 5 shown. Afterwards a State Byte is appended, which is used to mark the last assigned slot. Therefore, the first bit is used and the remaining ones are for future use reserved. The end of the header includes four bytes for the confirmation of Data Frame receptions. The receipt of messages in the last n − 1 slots is documented in the ActiveNodesBits.

A node x can recognize if it is in the Data Phase, when it is contained in its local node list. So that the node is known to the other ones it has to be present in their local lists. The devices clear its node list after sending an Init Frame, so the device has to be added to a neighbor list. If the neighbor sends its own Init or Sync Message later, then the address of the node x is contained in the list of the received message. Therefore, a node does not add itself to its local list but through received node lists in the Init or Sync Phase. This ensures, that a node is known to the other nodes.

3.5 Micro Mode

The purpose of the Micro Mode is the registration of a new or failed node via the µSlots at the end of the assigned slots. In the following the registration of new nodes via the µSlots is explained, whereby the described process is also valid for failed nodes. The term “Micro Mode” was used instead of “Micro Phase”, because the latter is misleading. In contrast to the other phases only a single slot is used instead of a whole frame, and only the new node is in this mode. Other known nodes do not switch to the Micro Mode. Therefore, the term “Micro Mode” is more suitablefor the registration process of a node via µSlots. A challenge here is the detection of the µSlot because the TDMA-schedule is unknown.

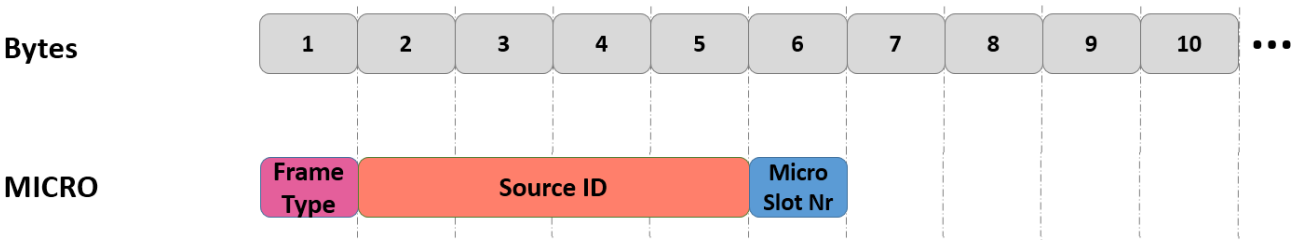


Figure 6. Protocol header of the Micro Mode.

The Micro Messages have to be small because of the limited µSlot size. For that reason, the header of the Micro Frames are the shortest in the protocol. The first five bytes are the frame type and the source id, in analogy to the other phases. In the end, the number of the occupied µSlot is appended as shown in Figure 6.

If a node receives an Init Message then the registration is made as described in the Init Phase. A registration via the µSlots is only made in case of a reception of Sync- or Data-Frames. This means, that the network is already formed and the current node has to be added instead of a new formation.

If the network is in the Sync Phase, then the new node receives Sync Frames with the current schedule, the node list. The node has to check the list and determine the slot. If the current slot is the last occupied one, then the next slot contains the µSlots.

Whereas, in the Data Phase no node lists are sent within the frames and therefore the schedule is unknown to the new nodes. For this reason, in the Data Header the first bit of the State Byte is

designed. This Micro Bit marks the last slot before the µSlots. After the node has determined the empty slot, a µSlot is chosen randomly. The node computes the duration to its chosen µSlot, whereas the reception of the last message is the reference point again. This means, the message which set the Micro Bit or in case of the Sync Phase, the last slot. The sent Micro Frame contains the randomly chosen µSlot. This is necessary, so that the other nodes can determine the remaining time to the end of the frame. Again relative to the reception of the Micro Message. The remaining time has to be determined since several nodes can register itself via the µSlots within the current frame.

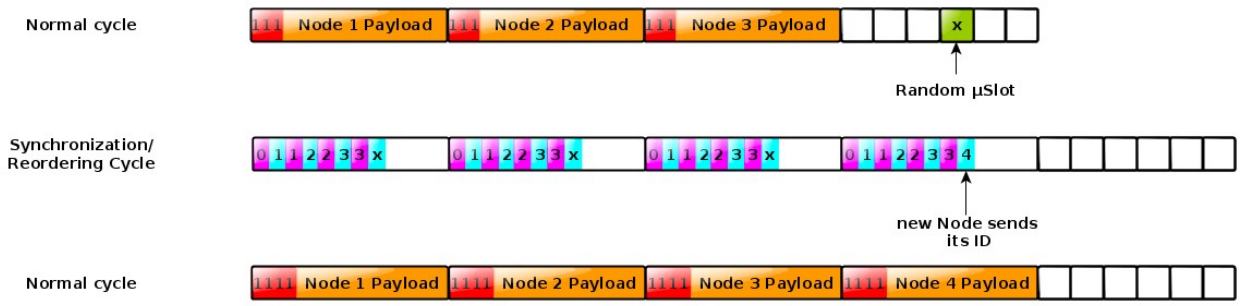


Figure 7. A new node is connecting.

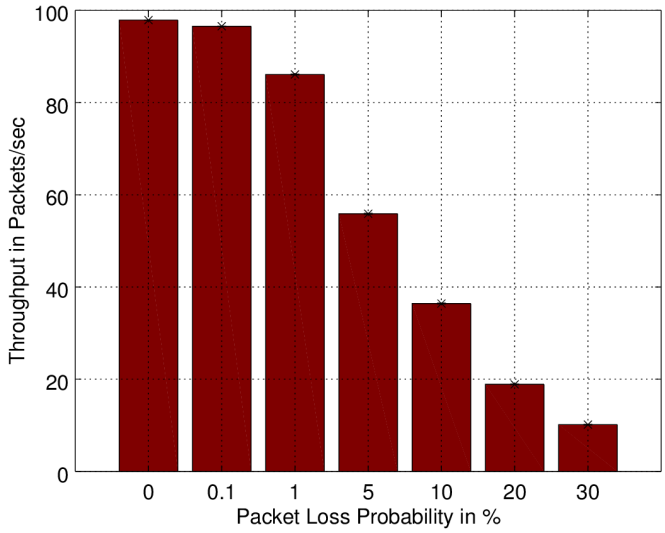
4 EVALUATION

The protocol has been tested in simulation and on real hardware. For the hardware testbed the aforementioned ultrawideband modules DW1000 by Decawave Inc. have been used, combined with STM32F4 series micro-controllers by STMicroelectronics [5] and the satellite operating system RODOS [6]. For testing special situations, which are hard to replicate in a hardware testbed the Omnet++ based simulation framework Castallia [7] has been utilized.

4.1 Simulation

The Simulation enables testing of the functionality and behavior of the protocol in special conditions. It has to be mentioned that Castalia does not support Ultrawideband connections (yet). Therefore general robustness and properties of the protocol can be evaluated, however real performance numbers can not be derived, as the real timing characteristics of the ultrawideband modules are not implemented in Castalia.

Scenarios like a varying packet loss in the wireless connection are hard to evaluate in a pure hardware testbed. Evaluated in the Catalia simulation of the protocol the packet loss characteristics are presented in figure 8. As anticipated with growing packet loss the throughput decreases, but even at 30% packet loss the network is still in a working (but limited bandwidth) condition.



Firgure 8. Measured throughput in the simulation with packet loss.

Figure 9 shows how long it takes the network to react on failing nodes and establish a working synchronized state again. In this sample a network consisting of 3 and of 8 nodes has been tested. In the first scenario (left side) two nodes failed and in the second scenario all nodes in the network failed. As one can see the results for the first scenario matches the second in a 3 node network, as two failed nodes are equivalent to a complete network failure in the 3 node setup. In the 8 node setup the time needed to return to a stable network is considerably lower, as there is no need to to start with a initialization phase again, because the rest of the network stays in a synchronized state.

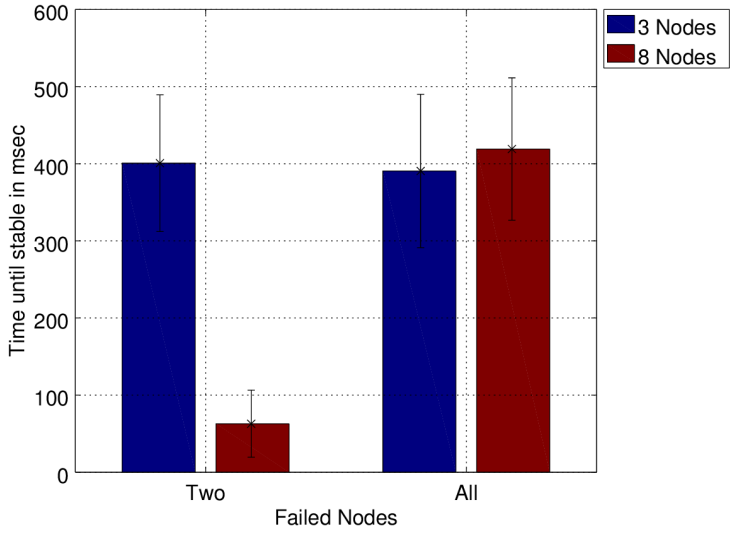
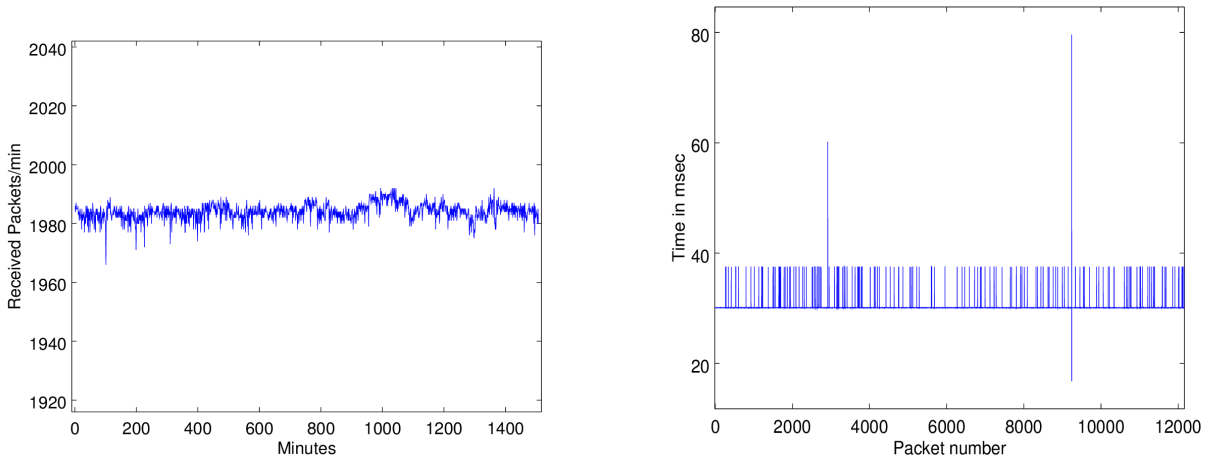


Figure 9. Measured throughput in the simulation with simultaneously failed nodes.

4.2 Hardware Testbed

In the hardware testbed we used 8 nodes each consisting of a DW1000 ultrawideband modules connected to a STM32 Discovery Boards, which feature a STM32F407 Microprocessor [5]. All measurements in the hardware testbed have been made for at least 24 hours, to catch sporadic errors and ensure longtime stability.



(a) Throughput in packets per minute. (b) Time between packets of the neighbor node.

Figure 10. Testbed with eight nodes and 100 byte payload.

Figure 10 show the result of one of the hardware tests. The 8 nodes created a network over the ultrawideband connection and continuously exchanged a payload of 100 bytes in each data frame. Between two packets send by the same node and received by another node in the network a time of m = 30.2 msec is measured with a standard deviation of 1.01 msec.

The measured data throughput is at 26.46 Kbit/sec. With chosen slot size of 10 msec per slot the theoretical limit would be 11.11 frames per second. However with the dynamically allocated slot time (see paragraph 3.3 "slot shift") of the protocol, the measured throughput comes up to three times more.

5 CONCLUSION

In this paper a innovative approach for a wireless sensor network protocol has been shown. The tests in simulation and on hardware showed a stable and robust operation. The protocol is not suitable for all wireless connections, but specialized for networks with a fixed, small number of nodes in confined space. For its purpose of enabling wireless connected avionics in satellites it features all required characteristics and will be further developed and tested in the ongoing YETE project.

6 ACKNOWLEDGMENT

This project is funded by the DLR agency under grant no. 50RA1332, what is gracefully acknowledged.

7 REFERENCES

[1] F. Kempf et al., *Reliable Networked Distributed On-Board Data Handling using a Modular Approach with Heterogeneous Components*, Wurzburg, DE.

[2] *IEEE 802.15.4 Standard-2011, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)"*, IEEE-SA Standards Board, 2011.

[3] P.~Moravek and V.~Stencel, Honeywell International, *UWB network demonstrator for space applications*, Czech Republic, 2014.

[4] *DW1000 User Manual*, Version 2.03, DecaWave Ltd, 2014.

[5] STMicroelectronics, *Datasheet: STM32F405xx*, STM32F407xx, Geneva, CHE, 2015.

[6] S. Montenegro, *RODOS Real Time Kernel Design for Dependability*, DLR, Bremen, DE, 2008.

[7] A. Boulis, *Castalia User's Manual Version 3.2*, NICTA, Sydney, AUS, 2011.