Wireless Time-Triggered Real-Time Communication

Bernhard Huber[1], Wilfried Elmenreich[1]

[1]Institut für Technische Informatik,
TU Vienna, Austria
e9926084@student.tuwien.ac.at
wil@vmars.tuwien.ac.at

Abstract — *Due to the increasing demand for mobility in the area of distributed systems, the use of wireless communication gains in importance.*
We present a wireless real-time communication protocol for the interconnection of several possibly mobile fieldbus clusters. The protocol is based on the time-triggered paradigm. A single dedicated master node is used to keep the clocks of all slaves synchronized, thus establishing a global time among all communication participants. To enhance dependability, the clock synchronization algorithm is designed to tolerate certain transient fail-silent failures of the master as well as the communication channel and permanent fail-silent behavior of up to $\lfloor \frac{n_s}{2} \rfloor$ *communication links, where* n_s *describes the number of slaves in the system.*
The paper presents a case study implementation, that shows the usability of the introduced protocol with commercial off-the-shelf components.

1 Introduction

The gradually miniaturization of electronic components, the increasing power of embedded processors, and the advance of micro-electro-mechanical systems (MEMS) enable the development of small, powerful, and cost-effective sensor/actuator elements which can be deployed in a wide range of applications. This trend assists the increasing popularity of fieldbus networks, where a bunch of such sensor/actuator elements is interconnected to a fieldbus cluster at low cost and using less wiring.
Applications, where fieldbus networks are used, range from industrial control systems like plant control, robot control, or supervision systems, over building automation systems for elevator control, air conditioning, or lightning control, to applications in the automotive sector used for engine management, diagnostics, or X-by-wire.

However, precisely because of the increasing number of sensor/actuator nodes and the new fields of application, the opportunities of fieldbus networks with respect to flexibility and mobility are not endless. Most notably the interconnection of several mobile fieldbus systems, i.e mobile industrial robots, or in inhospitable environments where less or suboptimal communication infrastructure is provided, considerably increases the requirements

on fieldbus networks and may suggest the use of a wireless communication protocol.
In most of the above mentioned fields of application real-time constraints have to be met, to correctly deliver the intended service. In applications, e.g., where a communication delay between the sensing and actuating instant affects the quality of the provided service, the real-time capabilities of the used communication protocol have to be considered. In robot soccer, e.g., tracking and estimating the course of the ball is a process, where communication delays reduce the chance of purposively hitting the ball.

The main design objective of the here introduced communication protocol was predominantly to establish a wireless real-time communication among a number of mobile robots, each one containing a local real-time TTP/A [1] fieldbus network. Since the design of this protocol is not subject to any restrictions forced by TTP/A, it could be used in any application where a time-triggered wireless communication is necessary.

The remainder of the paper is organized as follows: Section 2 summarizes related work on wireless real-time communication in sensor networks. The requirements of real-time fieldbus networks are discussed in section 3. Section 4 describes the architecture and the features of the wireless protocol. The implementation results of the case study are described in section 5. Section 6 concludes this paper.

2 Related Work

The fusion of fieldbus networks with wireless technologies is still a quite recent field of research these days.

Alves et al. propose a hybrid wired/wireless PROFIBUS solution, where proper real-time behavior of the overall network should be guaranteed. In this solution timing unpredictability problems, caused by the coexistence of heterogeneous transmission media in the same network, are eliminated through insertion of an extra inactivity (idle time) by master stations [2].

Another example for dealing with wireless communication in automation systems is the research project *R-Fieldbus:* High performance wireless fieldbus in industrial related multi-media environment [3]. The objective of this project is to develop a radio-based physical layer, based on the existing and available technologies in the LAN and WAN world, for modern production systems.

3 Requirements for Real-Time Fieldbus Networks

This section outlines the main requirements of real-time fieldbus networks identified by Kopetz, Elmenreich, and Mack in [4]:

Timeliness: As implied by the name, the most important requirement of a real-time fieldbus for embedded control applications is timeliness. This means temporal predictability, low latency, and minimal latency jitter.

High Efficiency: Due to the fact, that sensory data and actuator commands often consists only of a few bytes, the only way to achieve high data efficiency, is to keep the protocol overhead as small as possible. High data efficiency is also important for providing low latency.

Dependability: High dependability of real-time fieldbus networks is crucial for many fields of applications. This includes mechanisms for fault-tolerant behavior of a system as well as a high error-detection coverage. Because of the stringent constraints, a fieldbus system has to operate in, providing high dependability is a challenging task for the system designer.

On-line Diagnostics: In a fieldbus network the interconnected sensors can vary from very simple sensor/actuator devices to complex *smart sensors* that encapsulate sensor/actuator elements, data pre-processing components and network controllers in a single unit. There is a need for runtime monitoring and configuration of such complex devices. Special attention has to be paid that the diagnostic support does not cause a so called *Probe Effect* [5].

Sensor/Actuator Integration: To cope with changes in the controlled object and to benefit from the ongoing performance enhancements of MEMS, it should be possible that new or enhanced sensors could be integrated in the network, without the need of an extensive reconfiguration or even the suspension of the service (Plug and Play).

Low Complexity and Low Cost: Fieldbusses are often used in mass-produced applications. Thus, the costs of micro controllers, sensor/actuator devices, wiring, and installation has to be kept minimal to achieve a widespread use of the developed system.

3.1 Difficulties originated from Wireless Communication

Additionally to the above mentioned requirements, some further problems arise out of the use of a wireless communication medium.

Correctness of Transmitted Data:
As mentioned before, dependability is a crucial requirement for fieldbus applications. The correctness of the processed data is a must for the correctness of an application.
A wireless communication channel is inherently more interference-prone than a wired communication channel, and there are less protection possibilities against interferences than for wired communication channels (e.g., shielding, twisting, etc.). Thus, adequate mechanisms for error detection and error correction should be provided by wireless fieldbus systems.

The support for error detection is mostly done with additional parity bits at byte level and/or special check bytes at frame level. Successfully detected transmission errors are a pre-requisite for every error correction mechanism. A well-established error correction mechanism is the *automatic repeat request* (ARQ) scheme, where erroneous transmitted data is retransmitted.
The addition of bits and/or bytes to the original data causes a constant extension of the needed transmission time that can be considered a priori. On the other hand, error correction mechanisms depending on the retransmission of the erroneous data cause deviations of the original transmission time which can not be determined a priori. For real-time systems, where deadlines have to be met, this is usually not feasible. Uhlemann, Aulin, Rasmussen, and Wiberg [6] have proposed a ARQ scheme that deals with that problematic.

Communication (link) failures:
Additionally to transmission errors, the complete loss of communication links is more likely in systems using wireless communication. Depending on the used transmission medium, obstacles between sender and receiver, the transmission distance or natural or artificial environmental interferences have a more or less severe effect on the communication channel. Therefore, a wireless communication protocol has to deal with transient or even permanent loss of parts of the network.

4 Wireless Protocol Approach

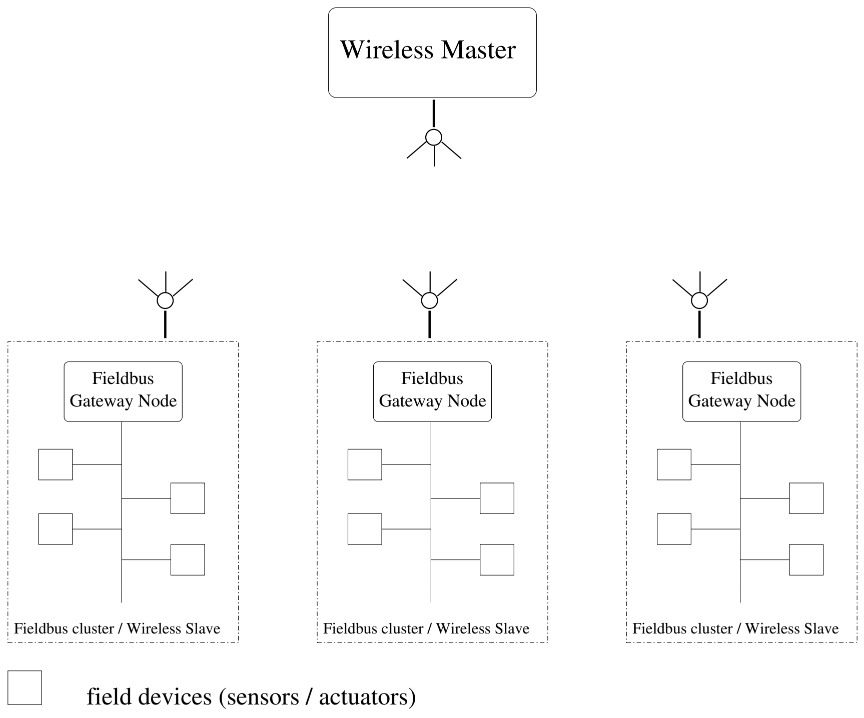
Wireless Master
Fieldbus Gateway Node
Fieldbus Gateway Node
Fieldbus Gateway Node
Fieldbus cluster / Wireless Slave
Fieldbus cluster / Wireless Slave
Fieldbus cluster / Wireless Slave
field devices (sensors / actuators)

Figure 1: Wireless interconnected fieldbus clusters

The here introduced communication protocol is based on the time-triggered paradigm. The main design objective was predominantly to establish a wireless real-time communication among several, possibly mobile, TTP/A clusters (see figure 1). A short introduction to TTP/A can be found in section 5.1.
In addition to the wireless real-time service, this protocol provides support for wireless monitoring of all nodes within the communication cluster without disturbing the real-time service or causing a “Probe Effect” [5].

4.1 Protocol Architecture

The protocol is controlled by a single dedicated master node. This master node is responsible for establishing the common time base between all nodes within the cluster and to control the communication among all slave nodes. For bus arbitration the time division multiple access (TDMA) scheme is used.
It is assumed that each slave has assigned an 16-bit address as unique identifier. Thus,

theoretically $(16^2) - 1 = 65535$ slaves (the address 0x0000 is reserved for the master) could be addressed. In practice it makes sense to use the first eight-bit as slave address and the second eight-bit as a more in depth reference to a sub-node or sub-component of the addressed slave.

The master node provides a synchronized global time base to all slaves. The notion of a global time among all communication participants is absolute essential for every distributed system where some kind of events have to be put into relation to each other. This surely applies to sensor/actuator networks. An eight-byte long time format is used to represent the global time. This time format, introduced by Kopetz [1], is based on GPS time and is split up into two parts. A 40-bit value defines a horizon of 2^{40} seconds of this representation; that is, more than 10 000 years. The other part consists of a 24-bit value that supports a possible granularity of 2^{-24} seconds; that is about 60 nanoseconds.
To overcome the main drawback of a single master solution – the problem of single point of failure – the design of this protocol tolerates transient failures of the master node. This is very imported in respect of the wireless communication medium where short link failures are not exceptionally. The duration of the tolerated master failures mostly depends on the required precision of synchronized clocks (see section 4.3).

To detect a possible corruption of transmitted data, the protocol includes some error detection mechanisms. On byte level the transmitted data contains a parity bit for error detection. Several sequentially transmitted bytes form a frame. Every frame is concluded by a check byte, i.e. an eight-bit checksum.

Furthermore, the master node acts as *gateway* to a higher level monitoring or controlling instance, e.g. a PC for monitoring the state of the wireless real-time system. To accomplish the gateway behavior the master provides the diagnostic and maintenance as well as the configuration and planning interface [1] to the outside.

4.2 Communication Rounds

The protocol uses a round-based communication style. Each round consists of one or more frames. A frame is a sequence of bytes transmitted from a single node. Between any two frames there is an inter-frame gap, a duration without communication. The length of the inter-frame gap is a parameter that mostly depends on the reached synchronization quality. Subsequent rounds are separated by an inter-round gap (see figure 2).

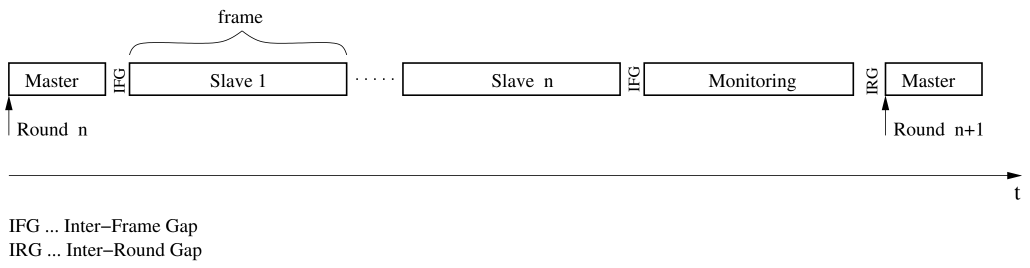

Figure 2: Typical communication round layout

The structure and duration of every round is static and defined a priori. The master selects the active round by transmitting a *round identifier* at the start of each new round. Due to this, the set of predefined communication rounds has to be common knowledge for all slaves in the system. The round identifier has a similar role as the fireworks byte of the TTP/A protocol introduced in [1].

We distinguish between three different frame types:

Master Frame: This frame is transmitted by the master node. The first byte of this frame indicates the start of a new round is the global synchronization point for all slaves. The master frame contains a *round identifier*, the next *monitoring request* and two *status/command* bytes.

Slave Frame: The slave frames are used to implement the real-time service. Depending on the individual needs, every slave can own one or more time slots for sending data frames. A configuration, where some slaves act as a kind of "passive member" without sending anything, is also possible.

Monitoring Frame: Different from the master or slave frame the monitoring frame is not statically assigned to a particular node. Depending on the monitoring request and the request type, this frame is either transmitted by the master or one of the slaves.

4.3 Quality of Synchronization

The quality of the global time, the common time base shared by all nodes of the system is essential for every time-triggered system. Standard measures for the quality of an ensemble of clocks are *accuracy* and *precision* [7].
The accuracy of clock k at micro-tick i is defined as:

$$accuracy_i^k = \max\{offset_i^k\} \tag{1}$$

The offset denotes the time difference between the respective micro-ticks of clock k in respect to the reference clock. Given a *period of interest*, that maximum offset over all micro-ticks in this period is called $accuracy^k$ of clock k.

$$\Pi_i = \max_{\forall 1 \leq j,k \leq n} \{offset_i^{jk}\} \tag{2}$$

Π_i is called the precision of the ensemble of clocks at micro-tick i. The maximum of Π_i over an *period of interest* is called precision Π of the ensemble. An externally synchronized ensemble of clocks with an accuracy A is also internally synchronized with a precision of at most 2A [7].
The offset of a clock k to a reference clock depends on the drift rate ρ^k of the clock. As mentioned above, the external synchronization event for all clocks in the ensemble is the first byte of the master frame. Due to the fact that the period between two master frames depends on the round length, the synchronization interval is equal to the round length of the active communication round. Thus, the maximum round length t_{round_max} and ρ^k determine the worst case accuracy of clock k.

$$A^k_{worst\_case} = \rho^k \cdot t_{round\_max} \tag{3}$$

Thus, the worst case precision of the ensemble is defined as:

$$\Pi_{worst\_case} = 2 \cdot \rho^k \cdot t_{round\_max} \tag{4}$$

To guarantee that no slave in the cluster misses the start of any frame, the length of the inter-frame gap t_{IFG} must be at least the worst case precision.

$$\frac{\Pi_{worst\_case}}{t_{IFG}} = k \qquad k \geq 1 \tag{5}$$

Out of (4) and (5) the lower bound of the inter-frame gap t_{IFG} is given as:

$$t_{IFG_{min}} = \frac{2 \cdot \rho \cdot t_{frame}}{k - 2 \cdot \rho \cdot (n+2)} \qquad k \geq 1 \tag{6}$$

t_{frame} is the overall time needed for the transmission of the master frame, the monitoring frame and n slave frames.

4.4 Self-Deactivation of Inexactly Synchronized Nodes

In a time-triggered system with an TDMA bus arbitration scheme, bus access is controlled by the progression of time. Thus, timing violations of the communication scheme have to be avoided in any case to prevent collisions on the shared communication medium.
To adhere to this condition we have established a local *time confidence value* on each node. The time confidence value denotes an estimation of the precision in the worst case, i.e., the node has drifted with the maximum expected drift rate. This estimation determines the maximum time between two subsequent synchronization events without violating the communication scheme.

The individual estimation mostly depends on the drift rate of the clock. Thus, the maximum time between two synchronization points is defined as:

$$t_{sync} = \frac{t_{IFG}}{2 \cdot \rho} \tag{7}$$

The initial time confidence value is an integer expression of the estimation and results from:

$$C_{init} = \lfloor \frac{t_{sync}}{t_{round\_max}} \rfloor \tag{8}$$

A slave node decrements its confidence value in every round where it fails to synchronize. To ensure that a node does not violate the communication scheme, it is only allowed to transmit data over the shared communication medium if the confidence value is greater than zero. Thus, the initial time confidence value determines the number of tolerated rounds without synchronization, before the node has to stop the communication.

Each frame in a round can be seen as an potential synchronization event and thus contains the initial and actual confidence of the sender. The master has the highest confidence. Thus, the reception of the first byte of the master frame can be seen as global synchronization event for all slaves in the system. Nevertheless, every correct synchronized node can overtake the role of a "second-rate time master" for a subset of other nodes, to keep them synchronized. This protocol feature is very important for wireless communication among possibly mobile nodes, where a fully connective network or even a permanent connection to the master can not be assured.

After the reception of a frame, each node k calculates an updated local confidence value $C^{k\prime}$. The updated local confidence $C^{k\prime}$ of node k with and initial confidence of C^k_{init} according to a synchronizing frame from node s with parameters C^s and C^s_{init} is defined as:

$$C^{k\prime} = \lfloor \frac{C^s}{C^s_{init}} \cdot C^k_{init} \rfloor \tag{9}$$

If $C^{k\prime} > C^k$, node k corrects its time according to the frame reception and alters its own local confidence value as defined in (9). Equation 9 ensures that the local confidence of a node remains less or equal than the nodes initial confidence and that the received confidence is adjusted according to the quality differences of both clocks. Furthermore, equation 9 ensures that the nodes confidence value is reseted to its initial value on the next successful synchronization with the master.

4.5 Fault Tolerance and Error Recovery

Kopetz ([7], page 73) defined the term fault hypothesis as:
"a statement about the assumptions that relate to the type and the frequency of faults that the computer system is supposed to handle".
For the here introduced clock synchronization algorithm, following assumptions can be made:

1. Permanent fail-silent behavior of the slave nodes is tolerated.
2. Transient fail-silent behavior of the master is tolerated, if the down time of the master node is less than the maximum synchronization interval (see section 4.3).
3. Transient failures of all links in the communication channel are allowed, if the time the links are broken is less than the maximum synchronization interval.
4. Permanent failure of $\lfloor \frac{n_s}{2} \rfloor$ (n_s is the number of slave nodes in the system) links in the communication channel is tolerated (see below). If we assume the probability of a link failure with 1%, the probability of the occurence of $\lfloor \frac{n_s}{2} \rfloor$ *independent* link failures is $0.01^{\lfloor \frac{n_s}{2} \rfloor}$.
5. With an absolute reference time at the master, the reintegration of the master after a transient failure is always possible.
6. Without an absolute reference time all slaves have noticed a permanent master failure and on this account safely stop their service not later than $\max_i (C_{init}) \cdot t_{round\_max}$ ($0 \leq i \leq n_s$), which is known in advance. A safe restart of the system can then be performed.

Toleration of $\lfloor \frac{n_s}{2} \rfloor$ permanent link failures
Let us consider the communication system as a fully connected undirected graph $G = (V, E)$ consisting of a set of nodes $V = \{v_0, v_1, \ldots, v_{n_s}\}$, where v_0 is the master node and n_s the number of slaves in the system, and a set of edges $E = \{e_{i,j} | 0 \leq i \leq n_s; i \neq j\}$. A function $c(e_{ij}) = c(e_{ji})$, where $c(e_{ij}) = 1$ if the link between node v_i and node v_j is correct or $f(e_{ij}) = 0$ otherwise, divides V into the pairwise disjunct subsets $C_i = \{v_j | e_{ij} \in E; c(e_{ij}) = 1; 0 \leq j \leq n_s; j \neq i\}$ and $F_i = \{v_j | e_{ij} \in E, c(e_{ij}) = 0; 0 \leq j \leq n_s; j \neq i\}$ with

$C_i \cup F_i \cup v_i = V$.
The constraint of a maximum of $\lfloor \frac{n_s}{2} \rfloor$ link failures per node is sufficient to guarantee that any slave node can be synchronized directly or indirectly by the master. To proof this assumption, the argumentation for a proof by contradiction is as follows:

Since there are at most $\lfloor \frac{n_s}{2} \rfloor$ link failures, $|F_i| \leq \lfloor \frac{n_s}{2} \rfloor$ $(0 \leq i \leq n_s)$ and therefrom $|C_i| = |V| - |F_i| - 1 = (n_s + 1) - \lfloor \frac{n_s}{2} \rfloor - 1 = \lceil \frac{n_s}{2} \rceil$ (note that this assumption is valid for the slave nodes as well as the master).
If there is a situation, where a slave v_i $(1 \leq i \leq n_s)$ cannot synchronize with the master v_0, v_i must not reach v_0 neither directly nor indirectly via a still synchronized slave v_j. In other words $v_0 \notin C_i$ and $C_i \cap C_0 = \emptyset$.
The overall set of communication nodes can be defined as $V = C_i \cup C_0 \cup v_i \cup v0$ $(1 \leq i \leq n_s)$. Due to $C_i \cap C_0 = \emptyset$ and $v_i, v_0 \notin \{C_i \cup C_0\}$, the cardinality of V is given as $|V| = |C_i| + |C_0| + 1 + 1 = \lceil \frac{n_s}{2} \rceil + \lceil \frac{n_s}{2} \rceil + 2 \geq n_s + 2$ which is a contradiction to the initial definition of V with $|V| = n_s + 1$.

4.6 Runtime Configuration/Monitoring

As the variability of real-time systems increases, assisted by the use of wireless communication media, the necessity of *dynamic reconfiguration* [8] of running systems increases. Debugging and analyzing the incorrect behavior of a distributed system is a very challenging task where *monitoring* is often used to locate the cause of the incorrect behavior. Monitoring allows to gather runtime information that cannot be obtained by a static analysis of the source code [9].
To cope with this requirements we use a static amount of bandwidth, the so called monitoring frame described in section 4.2, to deliver dynamic configuration as well as monitoring service.

The configuration/monitoring service is controlled by the master. The master frame, transmitted at the beginning of each round, contains among other information the 5-byte address and the mode of operation (two bits included in the slave address). The address format (see figure 3) aims at the use of the *interface file system* (IFS) of the TTP/A protocol [1] as the underlying source and sink of all communication activities.

ClusterID	NodeID	FileNr	Op–Code	RecordOffset	#Records

ClusterID	system wide unique identifier	(8 bit)
NodeID	cluster wide unique identifier	(8 bit)
FileNr	adressed file number	(6 bit)
Op–Code	operation mode (monitoring/configuration)	(2 bit)
RecordOffset	adressed record number (1 record = 4 bytes)	(8 bit)
#Records	number of adressed records	(8 bit)

Figure 3: Address format used in master frame

Depending on the operation type, the monitoring frame is used either by the slave (monitoring mode) or by the master (configuration mode). Due to the fact that the monitoring

frame is scheduled a priori, the non-disturbance of the real-time communication service can be guaranteed. If a configuration/monitoring request exceeds the static size of the monitoring frame, the request is proceeded in the monitoring frame of the succeeding rounds.
If all nodes, the master and all slaves, keep their protocol and node setup in a structured writable memory space, like the *interface file system* (IFS) of TTP/A, the configuration mode can be used to adopt node properties (e.g., sensor/actuator settings, self-describing information) as well as protocol properties (e.g., communication scheme) [10].

5 Case study: Wireless Networking between Multiple TTP/A Clusters

The case study is used to show the usability of the here presented communication protocol for establishing a real-time communication service among multiple self-contained fieldbus clusters (figure 4). We applied the fieldbus protocol TTP/A for implementing this clusters because of the support for runtime configuration, modification or exploration of all nodes of the system without influencing the predefined timing constraints of the real-time service [11].

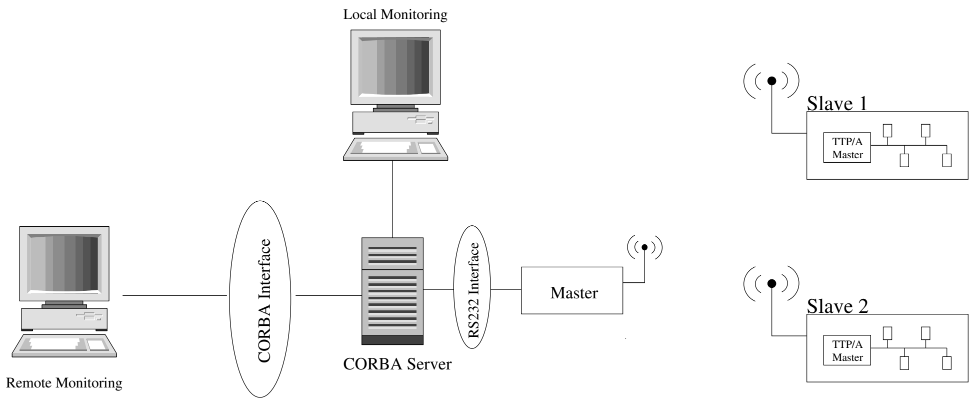

Figure 4: Wireless real-time communication with monitoring support among multiple TTP/A clusters

5.1 TTP/A: A low cost Real-Time Fieldbus

The TTP/A protocol is a time-triggered protocol for the communication among smart transducer nodes within a cluster [12]. A smart transducer is an integration of one or more sensors/actuators with a micro controller and a real-time network interface forming a mechatronic component [13].

TTP/A is a master/slave protocol controlled by a single master and up to 255 slaves nodes. The master provides the common time base for all nodes. Communication in TTP/A is organized in rounds. A round consists of several frames (sequentially transmitted bytes of a single node) which are separated by an inter-frame gap. A round starts with the so called fireworks frame from the master. The fireworks frame identifies the name and thus the type of the round and acts as a synchronization event for the slave nodes.

Two types of rounds are known in TTP/A, master/slave rounds and multi-partner rounds. While master/slave rounds implement the configuration and planning as well as the diagnostic and management interface [14], the multi-partner rounds provide the real-time service interface. Master/slave rounds have a fixed layout and can be used for configuration and detection of new slave nodes [15]. Multi-partner rounds are defined by the so called *round definition lists* (RODL) which are specified a priori and are common knowledge to all nodes in the cluster.

TTP/A is providing a structure where all relevant data, like round definitions, application specific parameters or sensor/actuator calibration values, is stored. This distributed interface file system (IFS) acts as source and sink of all data exchanged among the nodes of a cluster.

5.2 Wireless Communication

For wireless communication a *BIM-433-64 radio transceiver module* from Radiometrix Ltd. was chosen. This module provides half-duplex data transmission at ranges up to 200 meters external and 50 meters in buildings. The transceiver module operates on 433.92 MHz, a European license exempt frequency band. Data rates up to 64 kBit/s are achievable.

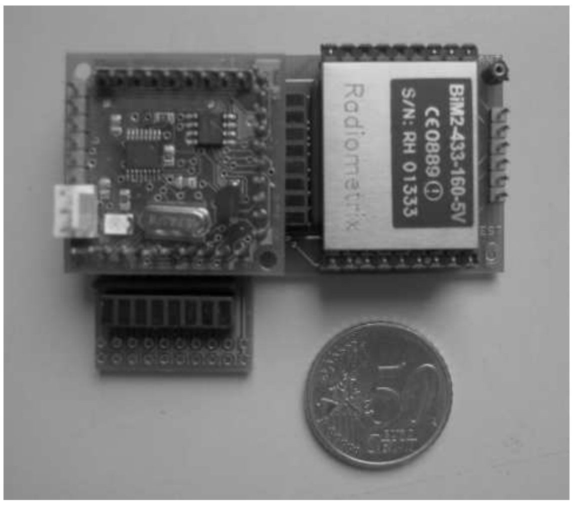

Figure 5: Master node connected to BIM-433-64 transceiver module

The transceiver module can deal with serial digital data as input and output respectively and thus can easily communicate with a standard *UART* (Universal Asynchronous Receiver and Transmitter) available on most current micro controllers. We implemented *manchester coding* and *modified frequency modulation* (MFM) as encoding scheme for the wireless transmitted data. To decouple and disburden the master and slave nodes from the data encoding, a two-tiered approach is used (see figure 5): Master and slave nodes transmit/receive the raw data to/from an intermediate *encoding unit*. This unit performs the data en/decoding and controls the transceiver module.
This intermediate encoding unit performs a rather simple task and can be implemented as well via a *field programmable gate array* (FPGA).

5.3 Protocol Implementation

The example implementation was carried out on Atmel 8-bit RISC micro controllers. Table 1 shows the three different micro controller types used for the implementation. The data encoding from simple UART format to a *manchester encoded* or a MFM encoded data format, is done by the so called *encoding unit*. This unit is used with both node types, the master as well as the slaves. Besides to the data encoding, this unit is also used for the controlling of the radio transceiver modules.

		Master	Slave	Encoding Unit
Micro Controller		ATmega128	ATmega 103*	AT90S2313
Available Memory	Flash	128K Bytes	128K Bytes	2K Bytes
	EEPROM	4K Bytes	4K Bytes	128 Bytes
	SRAM	4K Bytes	4000 Bytes	128 Bytes
Clock Frequency	max.	16 MHz	16 MHz	10 MHz
	used	14,7456 MHz	14,7456 MHz	9,8304 MHz
Peripheral Features	8-bit Timer/Counter	2 (0 used)	2 (1 used)	1 (1 used)
	16-bit Timer/Counter	2 (1 used)	1 (1 used)	1 (1 used)
	Serial U(S)ART	2 (2 used)	1 (1 used)	1 (1 used)

Table 1: Micro controllers used for case study.

*ATmega128 micro controller in ATmega103 compatibility mode

Important for the implementation of the case study was the availability of two serial UARTs at the gateway node (Atmel ATmega 128 micro controller). The use of a clock frequency of 14,7456 MHz allows a communication speed between the gateway node and the monitoring application via the hardware UART of 115,2 kBd. This speed is not achievable with a software implementation of the UART protocol and would consume to much processing power.
Furthermore, the use of two hardware UARTs affects the code size of the gateway node. Table 2 shows the code size of the current implementation of the wireless protocol. The overall needed resources for the slave node implementation include the implementation of the wireless protocol as well as the TTP/A protocol for communication in the local cluster.

	Master		Slave			
	available	used	available	used	Wireless	TTP/A
Flash	128 kBytes	$\approx$ 6.8 kBytes	128 kBytes	$\approx$ 9.3 kBytes	$\approx$ 4 kBytes	$\approx$ 5.3 kBytes
EEPROM	4 kBytes	0 kBytes	4 kBytes	322 Bytes	200 Bytes	122 Bytes

Table 2: Code size and memory usage of case study implementation

At the encoding unit, the UART data has to be received (via hardware UART), encoded according to the used encoding scheme (manchester or MFM) and transmitted via the radio transceiver module (software UART). To reduce the harmonic parts of the transmission signal, the signal transitions are unsharpened by a raised cosine filter. Both, the need for a software UART implementation and the raised cosine filter, cause that the encoding

units are bottleneck of the case study implementation with respect to the communication speed. With the current implementation, a maximum wireless speed of 19200 Bd can be reached (note that manchester encoded data needs two transitions per bit, therefore the real data rate is 9600 Bit/s).
To overcome this limitation, a FPGA implementation of the encoding unit was designed, where the full capabilities of the radio transceivers, up to 64 kBit/s, can be used.
As mentioned in section 3, a low protocol overhead is a crucial demand for fieldbus networks. Table 3 shows the frame length, the payload, the protocol specific data, and the resulting protocol overhead in percent.

	Length	Payload	Protocol data	Overhead
Master frame	10 Bytes	0 Byte	10 Bytes	100 %
Slave frame	37 Bytes	32 Bytes	5 Bytes	13,51%
Monitoring frame	34 Bytes	32 Bytes	2 Bytes	5,88%

Table 3: Overhead of the wireless protocol in the case study implementation

The overall length of a communication round is made up of:

$$round\_length = gateway\_frame + n{\cdot}slave\_frame + monitoring\_frame \tag{10}$$

Where n depicts the number of interconnected slaves (TTP/A clusters). In our case study setup – two TTP/A cluster are connected – the overall round length results in $roundlength = 10 + 2 \cdot 37 + 34 = 118$ Bytes. Therefrom, the protocol overhead results in 22 Bytes or 18,64% of the communication data.

6 Conclusion

The introduced communication protocol is well-suited for the interconnection of several, possibly mobile fieldbus networks, due to the wireless communication medium and the real-time characteristics. The protocol is based on the time-triggered paradigm, thus the communication schedule is static and defined a priori. Although the protocol supports the run-time configuration of the communication schedule, the rapid and incessant integration/drop of nodes in/from the network is not supported.

Using wireless communication increases the requirements, particularly on dependability, of the communication protocol. For time-triggered systems the global time is crucial for the correct function of the system. Therefore, the clock synchronization algorithm of this protocol is designed to tolerate a certain kind of failures, including transient fail-silent failures of the master and permanent fail-silent behavior of up to $\lfloor \frac{n_s}{2} \rfloor$ communication links.

The accomplished case study shows that the protocol can be used on off-the-shelf hardware components, which reduces the required implementation costs.

References

[1] Stephan Eberle, Christian Ebner, Wilfried Elmenreich, Georg Färber, Peter Göhner, Wolfgang Haidinger, Michael Holzmann, Robert Huber, Ralf Schlatterbeck, Hermann Kopetz, and Alec

Stothert. Specification of the TTP/A protocol. Research Report 61/2001, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2001.

[2] Mario Alves, Eduardo Tovar, Francisco Vasques, Gerhard Hammer, and Klaus Rther. Real-time communications over hybrid wired/wireless profibus-based networks. *Euromicro Conference on Real-Time Systems (ECRTS'02)*, 14th, June 2002.

[3] Joerg Haehniche and Lutz Rauchhaupt. Radio communication in automation systems: the R-fieldbus approach. In *3rd IEEE International Workshop on Factory Communication Systems (WFCS 2000), 6.-8. September 2000, Porto, Portugal*, Sep. 2000.

[4] Hermann Kopetz, Wilfried Elmenreich, and Christoph Mack. A comparison of LIN and TTP/A. In *3rd IEEE International Workshop on Factory Communication Systems (WFCS 2000), 6.-8. September 2000, Porto, Portugal*, pages 99–107, Sep. 2000.

[5] C.E. McDowell and D.P. Helmbold. Debugging concurrent programs. *ACM Computing Surveys*, 21, No.4, Dezember 1989.

[6] Elisabeth Uhlemann, Tor Aulin, Lars K. Rasmussen, and Per-Arne Wiberg. Concatenated hybrid ARQ - a flexible scheme for wireless real-time communication. In *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, pages 35–45, Sept. 2002.

[7] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[8] J. Kramer and J. Magee. Dynamic configuration for distributed systems. *IEEE Transactions on Software Engineering*, 11 (4), 1985.

[9] K. Slind J. Joyce, G. Lomow and B. Unger. Monitoring distributed systems. *ACM Transactions on Computer Systems*, 5 (2), May 1987.

[10] Philipp Peti, Roman Obermaisser, Wilfried Elmenreich, and Thomas Losert. An architecture supporting monitoring and configuration in real-time smart transducer networks. In *The First IEEE International Conference on Sensors (IEEE SENSORS 2002)*, Jun. 2002.

[11] P. Peti. Monitoring and configuration of a TTP/A cluster in an autonomous mobile robot. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

[12] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[13] R. Schlatterbeck and W. Elmenreich. TTP/A: A low cost highly efficient time-triggered fieldbus architecture. *SAE World Congress 2001, Detroit, Michigan, USA*, March 2001.

[14] W. Elmenreich, W. Haidinger, and H. Kopetz. Interface design for smart transducers. *IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary*, 2001.

[15] Wilfried Elmenreich, Wolfgang Haidinger, Philipp Peti, and Lukas Schneider. New node integration for master-slave fieldbus networks. In *Proceedings of the 20th IASTED International Conference on Applied Informatics (AI 2002)*, pages 173–178, Feb. 2002.