2008 IEEE International Workshop on Factory Communication Systems

WFCS 2008 Proceedings

May 21 - 23, 2008
Steigenberger Hotel de Saxe
Dresden, Germany

Edited by:
Gianluca Cena and Françoise Simonot-Lion

IEEE
Industrial Electronics Society

TU Dresden
Institute of Applied
Computer Science

IEEE Catalog Number: CFP08WFC-PRT

Table of Contents

Wireless Networks II

Session Chair: Andreas Willig, TU Berlin, Germany

Industrial Communications

Session Chair: Max Felser, Berne Univ. of Applied Sciences, Switzerland

Keynote

Session Chair: Gianluca Cena, IEIIT-CNR, Italy

Safety & Security

Session Chair: Julian Proenza, University of the Balearic Islands, Spain

WiP Session 2: Building Automation, Industrial Communications, Applications

Session Chair: Nicolas Navet, INRIA, France

Dependable Networks

Session Chair: Ye-Qiong Song, INPL-LORIA, France

State-of-the-Art

Session Chair: Wolfgang Kastner, TU Vienna, Austria

Middleware I

Session Chair: Klaus Kabitzsch, TU Dresden, Germany

Middleware II

Session Chair: Leon Urbas, TU Dresden, Germany

Middleware III

Session Chair: Martin Wollschlaeger, TU Dresden, Germany

Network Time Synchronization in a Safe Automation Network

Thomas Novak
Vienna University of Technology
Institute of Computer Technology
Gusshausstrasse 27-29/384
1040 Vienna, Austria
novakt@ict.tuwien.ac.at

Berndt Sevcik
Vienna University of Technology
Institute of Computer Technology
Gusshausstrasse 27-29/384
1040 Vienna, Austria
sevcik@ict.tuwien.ac.at

Abstract

Today there is a great request of using automation networks in safety critical environments. Thus, such systems used in industrial and building automation have been enhanced with safety features derived from strict safety requirements specified in IEC 61508.

One of the safety requirements to be met is the detection of hazardous events on the network such as the delay of a message. A safety measure to identify these hazardous events is the use of a timestamp within a safe protocol. However, a common time base among the nodes must be provided – by means of a network time synchronization mechanism.

A centralized and decentralized approach was developed within the SafetyLon project. Realization of the concept and implementation in the producer/consumer model of SafetyLon is outlined.

1. Introduction

In the last years automation networks have been more and more used in new fields of application that are very demanding regarding functional safety (short safety). As a consequence, they must be considered as safety critical systems and have to meet very strict requirements specified by the international standard IEC 61508 [10].

The standard deals with functional safety achieved by a safety related device. The idea is to reduce the inherent risk of an equipment under control (EUC) below a maximum tolerable level by using safety related devices. Within this paper the EUC comprises the network as well as the nodes of an automation network. The safety related device is integrated into every node and is realized by microcontrollers and embedded software. In the following such a node is called safe node.

Risk reduction is achieved by avoiding *systematic* and handling *stochastic* failures.

1. Fault avoidance: By applying different measures during the life cycle of a system, *systematic* failures should be avoided. Such measures are failure mode and effect analysis (FMEA) or code walkthroughs. However, that topic is beyond the scope of this paper.
2. Fault control: *Stochastic* failures cannot be avoided. Therefore faults must be controlled and handled by proper means. Generally speaking, fault control means that a fault does not result in a failure due to redundancy or that the fault has been detected and repaired before it resulted in a failure.

Faults result in failures that in the end lead to hazards. It must be distinguished between hazards coming from failures on the network and hazards resulting from failures in the safety related device on the safe node. The last-mentioned failures can either be *systematic* software failures or *stochastic* hardware failures. Refer to [10] for avoiding *systematic* software failures and [11], [12] for ways of detecting *stochastic* hardware failures.

In [5] typical hazards occurring on the network are mentioned. The consequences of these hazards are hazardous events:

- Data corruption
- Loss of messages
- Insertion of messages
- Delay, repetition, wrong sequence of messages
- Non-safe message looks like a safe message

The hazardous events must be detected with a certain probability. Therefore, so-called safety functions are specified performing tasks to detect the events. The probability of detecting hazardous events is categorized by safety integrity levels (SIL).

IEC 61508 specifies four safety integrity levels (SIL). Safety integrity level 1 (SIL 1) is the lowest and safety integrity 4 (SIL 4) is the highest level. Each level corresponds with a specific error probability per hour. The higher the level, the higher the performance of a safety function must be, i.e. the higher the likelihood of detecting hazardous events has to be.

Regarding the aforementioned hazardous events safety functions would be [2]:

- CRC (cyclic redundancy check) to ensure data

978-1-4244-2350-7/08/$25.00 ©2008 IEEE.

integrity,

- Watchdog to detect loss of a message,
- safe addressing scheme and timestamp to discover insertion of messages,
- timestamp to identify delay, repetition, wrong sequence of a message,
- redundancy with cross comparison to detect repetition, loss, insertion, wrong sequence of a message.

Using timestamps as means to detect hazardous events on the network during message exchange among safe nodes requires a network time synchronization mechanism. Timestamps are only an effective safety measure if safe nodes have the same time base.

In the following a network time synchronization approach is being presented: a centralized and distributed one. Especially, the paper focuses on the realization in and implementation into the SafetyLon The approach was developed during the SafetyLon project. The European collective research project SafetyLon has the goal to make the EN 14908 (Local Operating Network, LON) technology [14] safe according to the requirements of SIL 3.

Consequently, the remainder of the paper is structured as follows: section 2 conveys information on network time synchronization approaches in *safety-related* automation systems. Section 3 presents the case study. It outlines some aspects of SafetyLon, required to understand the following sections: hardware and software architecture of a node, and the communication concept. Section 4 discusses the time synchronization concepts whereas section 5 mentions the realization within the SafetyLon. Finally, section 6 is related to the implementation of the two network time synchronization approaches.

2. State of the Art

SafetyLon is supposed to be the first safe building automation network. However, not the first to be enhanced with safety features. Especially, some industrial automation networks have been realized meeting the requirements specified by IEC 61508.

EtherCAT which exists since 2003 is an Ethernet based subsystem which was extended with safety functions and is specified as "Safety over EtherCAT" [4]. It meets like SafetyLon SIL 3 requirements of IEC 61508. Safe and non-safe communication is transmitted over the same communication medium. It is based on a producer/consumer concept. Safety functions are encapsulated in a Safety Layer which handles the EtherCAT messages. Instead of a time synchronization mechanism, a dedicated master/slave connection is realized which enables the full observation of the transmission path.

Another SIL 3 compliant system is "Ethernet Powerlink Safety" [6]. Safety functions are located in a dedicated Safety Layer within the embedded software. Safety functions to detect hazardous events on the network are similar to those in SafetyLon: duplication of the data to ensure data integrity, usage of timestamps and safe addressing model to identify inserted messages are implemented. Time synchronization is based on a relative time mechanism [3] which measures the time difference between sender and receiver and considers it for later communication.

PROFIsafe [5] is another SIL 3 compliant system where safety functions are integrated into a PROFIsafe layer located on top of the OSI model. Time synchronization mechanism is not implemented into the protocol because acknowledgements are used instead.

Time synchronization itself is a broad field of research. There are mechanism available like NTP [9], IEEE 1588 [7] or hardware based implementation like SynUTC [8]. SynUTC and IEEE 1588 specify concepts to synchronize time with an accuracy of smaller than microseconds. NTP synchronizes time in the range of a few milliseconds. The very simple requirements of SafetyLon regarding time accuracy do not justify the implementation of those concepts. The measurement of the delay from IEEE 1588 e.g. is replaced by a simple transaction time. Also the storing of four timestamps in a packet specified in NTP to achieve a high time accuracy are not required. Less overhead during transmission is favoured (packet size typical 90 byte for NTP in contrast to 22 byte with SafetyLon). Hardware based time stamping like implemented in SynUTC was not considered because of the already pre-defined hardware.

In SafetyLon time accuracy of about 10 ms is required. Also requirements with regard to hardware resources (64 kByte RAM, 256 kByte Flash memory, clock rate 43.2 MHz) have to be considered. Consequently, resources saving mechanisms are mandatory.

3. Case Study

As already mentioned shortly, the network time synchronization approach presented in the paper was developed in the SafetyLon project. It is a European collective research project, project number 012611, supported by the European Union within the Sixth Framework Program. The consortium consists of 17 partners: universities, companies and user groups of seven European countries.

SafetyLon is the safe extension to the LON technology. As a consequence a standard EN 14908 [14] node is enhanced with additional hardware

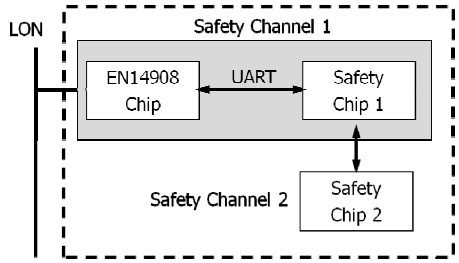

Figure 1 SafetyLon node

(Figure 1) and embedded safety operating software [2]. Doing so has the advantage that safe and non-safe services are provided on a safe node. Moreover, it is possible to use safe and standard nodes within the same network. And the powerful network management tools used to setup the network are upgraded to meet requirements of a SIL 3 compliant system [15].

3.1. Hardware Architecture

Every safe node includes a standard EN 14908 chip to access the LON. Additionally, there are two safety chips: Safety Chip 1 and Safety Chip 2 [2]. Safety Chip 1 is physically connected to the standard EN 14908 chip and uses it as network interface.

Both safety chips perform the safety functions in close cooperation, are synchronized with each other and are connected via a serial interface.

Sensors and actuators are connected to the safety related input and output unit. The inputs and outputs are controlled by both safety chips.

3.2. Software architecture

The software architecture of SafetyLon embedded software is based on a layered architecture [2] as illustrated in Figure 2. The Application Layer on top of the model is offering functions for developing safe user-defined applications to the application developer.

The Safety Layer in the middle incorporates all the safety functions required to achieve SIL 3. Besides the online hardware self tests [11], the safe protocol stack [2] and the network time synchronization is part of the Safety Layer. Safety functions are called periodically by the Safety Operating Scheduler.

The lower layer is divided into two parts. The Safety Chip Interface is responsible for the communication between the safety chips and is separated into an API and an interrupt based serial driver. The Network Access Layer offers functions to access the LON, independent of the underlying third party software [19], [21]. The Network Access Layer is only available on Safety Chip 1 because only this safety chip is connected to the EN 14908 chip as mentioned in subsection 3.1. Standard LON

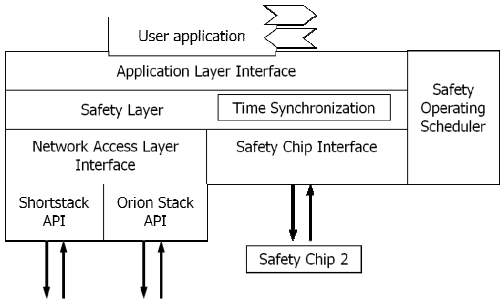

Figure 2 Software architecture of safety operating software (Safety Chip 1)

communication channel and the third-party software is treated as grey channel and not considered in safety considerations.

3.3. Communication Concept

Safe communication among nodes is based on the producer/consumer model. In general, producers are generating (producing) messages and consumers are "consuming", i.e. are processing messages. Typically, producers are sensors and consumers are actuators.

Producers and consumers on a single node get safe addresses. Additionally, consumers keep a list with safe addresses of the connected (bound) producers. This process is called safe binding.

When a producer wants to send a message, it includes its safe address into the message. The consumer only processes a message if the safe address is listed in the table, i.e. it is a valid producer. This addressing scheme is called source based addressing model.

The producers with a safe address are periodically sending "hello" or "keep-alive" messages, called heartbeats in the following. They are sent to proof the aliveness of producers. In case of missing heartbeats, i.e. timing expectation has not been fulfilled, watchdog is triggered and the consumer has to enter a defined safe state.

Safe messages (Figure 3) are exchanged using a safe message format of a safe protocol. It is specified in a way so that hazardous events mentioned in section 1 can be detected [2]. Therefore it consists of two parts to increase the level of integrity, including beside a length field (ID):

- 3 byte safe address field (Address),
- 2 byte timestamp field (Ms word, Ls word),
- 1-8 byte payload field (Data),
- 1 byte CRC field.

The safe protocol is embedded into the payload field of the LonTalk [18]. As a consequence, routing of messages is provided by LonTalk only.

ID	Address	Time stamp MSWord	Data n Byte	CRC 1	ID	Address	Time stamp LSWord	Data n Byte	CRC 2

Figure 3 Safe protocol message structure

4. Synchronization Concept

Faulty transmission is caused by hazardous events resulting for example from broken cabling, stochastic failures or wrong wiring. As already mentioned in section 1, typical hazardous events are delay of a message, message loss or duplication of a message which have to be detected [3]. Some of such hazardous events can be detected by using acknowledgements, sequence numbers or safe addresses.

The detection of a delay of a message is absolutely required in a safe automation network with devices that store messages temporarily, e.g. store-and-forward routers. That service and the respective watchdog functionality, however, can only be provided if timing expectation can be checked. A common timing expectation among nodes is only possible when nodes have the same time base. That is why a network time synchronization service is included into SafetyLon. It is a fundamental part of the system during startup of the network and during operation.

The service enables checking of every message during regular operation regarding e.g.:

- Is the receiving time of the message in the future?
- Is the last received message older than the last saved one?
- Is the maximum delay too high?

Time synchronization in SafetyLon is similar to the relative time synchronization method [3] implemented in Ethernet Powerlink Safety (EPLsafety) [1]. With regard to a network time resolution[1] of 10 milliseconds on a TP/FT-10 channel [20], sending and receiving of a single message lasting about 50-60 milliseconds, the time accuracy of the relative time synchronization method is sufficient.

The network time synchronization approach is based on a request/response service. In the payload of a request and response message a CTN (Consecutive Timing Number) is transmitted (Figure 4). The CTN value is used to correlate the request with the response for detection of chronological disorder of replies and to prevent faulty synchronization.

Each SafetyLon message contains a sending timestamp in the message header. To calculate the offset of the timing source, the timestamp of the header is extracted (Offset = ST1 - MT1). If the interval between the request and the response is too long (Response Time Δt = ST1 - ST2) the message is discarded and a new request is initiated. This condition is always checked in order to detect too big deviations due to network delays.

[1] Network timers are realized by counters. The counter value , i.e. the network time, is increased from *x* to *x+1* every 10 milliseconds. Therefore, the network time resolution is the time that goes by between value *x* and *x+1*.

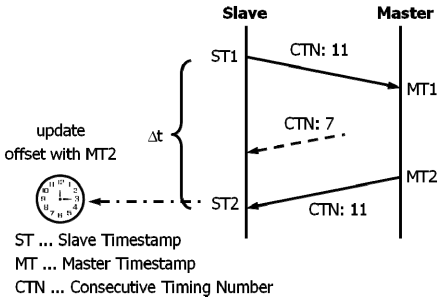

Figure 4 Time synchronization mechanism

The request/response service is used in two different time synchronization approaches. The first is based on a central algorithm (single master/multiple slaves) and the second one is based on a distributed or consumer/producer algorithm with different time bases (multiple masters and multiple slaves).

In the central algorithm a dedicated node being the timing master is available. The master receives a request from the slave and responds with the current time. The slave uses the received time as new network time as shown in Figure 3.

In the distributed algorithm every producer synchronizes the network time with its associated consumer. In contrast to Figure 4 the timing master, i.e. the consumer, starts to send its network time to all producers. They are using the value as their new network time and are sending a response as acknowledgement. Thus, all producers connected to a single consumer share the same network time. However, different consumers, especially on different nodes, need not to have the same network time. It is likely that various network times are present.

To sum up, in the centralized algorithm network time of all nodes sending request to timing master is updated with the same value. Moreover, the network time is valid for all producers and consumer on a single node. However, only consumers on the same node and producers connected to the same consumer respectively share the same network time when the distributed algorithm is applied. Hence, producer on the same node as well as consumer on different nodes have different network times.

The request/response service is necessary because the slave must be able to measure the delay for example due to congestion on the network. If the time between sending the request and receiving the response is above a specified time limit, accuracy of time is not guaranteed. The time limit is set according to the network time resolution.

5. Realization of the Concept

In SafetyLon the centralized algorithm uses an additional time synchronization consumer (here and

after called only consumer) on slave side and a dedicated time synchronization producer (here and after called only producer) on master side. That is not the case applying the distributed or consumer/producer algorithm. However, realization of the concept is equal and presented as follows.

On the contrary to the fact presented in subsection 3.3 that a producer sends and a consumer receives messages, time synchronization is initiated by the consumer. The reason is the following: Only on consumer side timing expectation by means of a watchdog timer is checked. If watchdog is triggered, consumer can enter a defined safe state. Thus, consumer sends request and, if it does not receive at least a response message with a defined time frame, it enters safe state.

Network time synchronization is a safety function. Consequently messages are sent via the safe protocol. For that reason a safe binding between producer and consumer is required. As outlined in subsection 3.3 the producer and the consumer receive a safe address. Moreover, on consumer side the safe address of the producer is stored in a list, called the consumer table. Only if a message with a safe address stored in the consumer table has been received, it is processed.

The realization can be seen in Figure 5 for the network variable service of EN 14908 discussed in detail in section 6. Only for time synchronization requests the consumer is sending the safe address of the producer in the payload of the request message. The sender address is the safe address of the consumer. As already mentioned in subsection 3.3, SafetyLon is using a source based addressing model where the source address is part of the safe protocol header.

The producer receives the request message and checks if the payload field contains the valid producer address. If not, the message is discarded. Otherwise the producer generates a response message. Now the producer address is the source address and the sender address of the received message, i.e. consumer address, is included in the payload field.

After receiving the response message on consumer side, the producer address is compared with the associated one listed in the consumer table. In addition, it is verified if the payload field contains the safe address of the consumer. With this mechanism the consumer can assure that the response it received is from the producer it sent the request to first. Only if the cross comparison of safe addresses is successful, the time synchronization message is processed, otherwise it is discarded.

6. Implementation in SafetyLon

As shown in Figure 2, network time synchronization is a software module within the Safety Layer. It is implemented in a way to meet requirements of

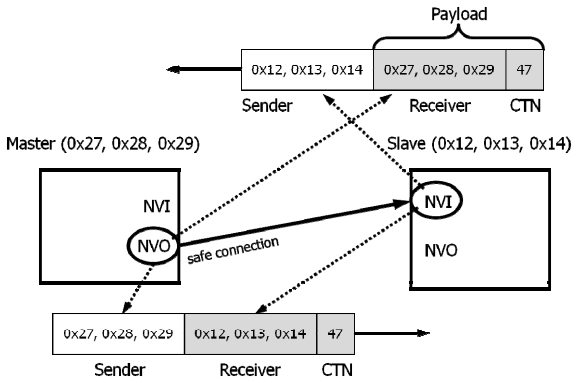

Figure 5 SafetyLon addressing for time synchronization with network variable service

IEC 61508 regarding software development. In other words, structured programming, modularization of the software, coding rules are applied as outlined in detail in [13].

6.1. Integration into the System

Network time synchronization software module is integrated into the software on both safety chips. Due to the underlying Network Access Layer on Safety Chip 1, the time synchronization software module is absolutely independent of the hardware architecture. The module is periodically called by the Safe Operating Scheduler on each safety chip independently of each other. The scheduling, the internal coupling of and same time synchronization related data on every safety chip make it possible that the time synchronization is executed on both safety chips at the same time.

As a consequence, every safety chip must keep two important tables for storing the safe binding between producers and consumers: the consumer table lists the safe consumer addresses and all connected producers with their state (last message received, receive rate); the producer table the safe producer addresses and timing information (last message send, send rate). The importance of the tables for the time synchronization is twofold:

- to realize the network time synchronization concept mentioned in section 5,
- to acquire timing information: when should a request be triggered next and when has a response to be received the latest.

The synchronization interval depends on the drift (G_{Quarz}) of the safe node oscillators and the maximum allowable timing window (Δt_{Drift}). Under the assumption that both oscillators drift in opposite direction the interval can be calculated by:

$$\Delta t_{Drift} = G_{Quarz} \cdot T_{Sync}$$

At least three synchronization trails are recommended during an interval.

Since both safety chips have the same time synchronization related data available, the time synchronization module independently triggers sending a part of the request and the response messages on each safety chip. As mentioned in subsection 3.3, a safe message always consists of two duplicated message parts. So in the safe protocol stack – another part of the Safety Layer – of Safety Chip 1 the first and of Safety Chip 2 the second part of a request or response message is built. Safe protocol stack of Safety Chip 1 concatenates both parts and triggers the sending process.

To keep time synchronization related data consistent on both safety chips, the received safe request/response messages must be available on each safety chip. Thus, safe protocol stack of Safety Chip 1 that is only connected to the EN 14908 chip forwards the complete message to its companion.

In general, a safe node can receive three types of messages:

1. A safe network management message using the message format presented in [15] and to be handled by the safe network management tool.
2. An application related message using the safe message format where data is processed by the application.
3. A network time synchronization using safe message format and being handled by the network time synchronization module.

Message type 2 and 3 are using the same safe message format. Hence, a network time synchronization message is marked with the T/D (Time/Data) bit in the second byte of the safe address field (Figure 3). The T/D bit is the most significant bit in the second byte of the safe address. The bit is set and evaluated by the safe protocol stack.

6.2. Centralized Algorithm

The centralized algorithm with a single master and multiple slaves uses the EN 14908 concept of network variables (NV). Generally, a network variable is a data item that an application on Node A expects to get from Node B on a network (an input network variable) or expects to make available to Node B on a network (an output network variable). Network variables are used for operational data such as temperatures, or pressures. Each network variable has a special network variable type that specifies the units, scaling and structure of the NV [19]. Network variables of same type, but opposite direction can be connected – they are bound by performing a binding. Data of each NV can easily be referenced by a NV index. In the EN 14908 chip a table is kept where each NV index matches a LonTalk address for routing purpose. This kind of mechanism is used because it is supported in a very comfortable way by standard EN 14908 network management tools.

Applying the centralized algorithm, two dedicated roles are defined. The master is the source of timing information for all nodes in the network. The slaves are querying the master for the time information. Selection of the master node is done before compilation of the software. Each node provides two safe NVs for synchronization where one is configured as input (consumer role, NVI) and the second one as output (producer role, NVO). The timing master can be implemented as an already existing node in the network – preferably powerful regarding computational power and memory resources.

A safe binding is established only for the response path from the master (NVO) to the slave (NVI) (bold arrow in Figure 4). That is because a safe binding always results in sending a heartbeat periodically. As a consequence, the timing master would be flooded with heartbeats from many slaves. Moreover, from a more general perspective, it is absolutely irrelevant for the timing master if a timing slave is still available. Timing master is not affected in any case. On the contrary, timing slave is affected if timing master is not “alive” any more. It cannot synchronize network time, and therefore it is not able to check timing expectation; an important safety feature is not working.

The safe network variables are always assigned by definition to the last but one and the last NV index. So in case of n defined network variables necessary for safe user-defined application, $n+1$ is the output NV and $n+2$ the input NV on every node. The safe address of a producer (NVO) on master node side and a consumer (NVI) on slave node side can be acquired by specifying the corresponding NV index.

6.2.1. Slave Node Functionality

Generally, a node does not start to send and process application related messages unless it has received a valid time synchronization response message. In addition, it stops sending and processing application related messages in case of not getting a time synchronization response messages for a while. As a consequence, it has to be distinguished between startup time synchronization and the one during operation.

In case of startup synchronization the network time is set immediately to the received value. By doing so, the node can start to send/receive messages with out delay. In the following network time is adjusted by speeding up or slowing down the node internal network time. Such a way is required because otherwise watchdog functionality is affected.

For example, a consumer on Node A received a heartbeat from a producer on Node B at network time 435 time unit. It is specified that consumer gets the next heartbeat at 455 time unit, i.e. watchdog must be triggered every 20 time unit. Meanwhile 17 time units were gone by and no heartbeat has been received so far. If network time had to be adjusted to +4 time units

and was set immediately, watchdog would not be triggered and consumer would have to enter safe state.

As already outlined before, network time synchronization is a software module within the Safety Layer and called periodically by the Safety Operating Scheduler. Every time it is called by the scheduler a new request can be initiated and a received response can be processed.

Sending a request can have different causes:

- The node was never synchronized before (startup synchronization).
- The configured time synchronization interval expired.
- The existing requests are only valid for a limited time, i.e. until a timeout occurred. After expiration of the period the old request is marked invalid and a new request with a new CTN value is generated.

As soon as new messages have been available for processing, they are first checked regarding the following criteria.

- Is there a request with the same CTN value?
- Is the arrival time within the allowed response interval?
- Was the cross comparison of the safe addresses successful?

In case of fulfilling all three criteria, the response is processed or otherwise discarded and a new request is sent. If the calculated offset is greater than a predefined minimum value, but smaller a maximum value, the network time will be slowly adjusted in positive direction to the correct value. With every execution of the synchronization algorithm the internal network time is sped up or slowed down. A maximum value is specified because great deviations of the internal network time from the network time received are a sign for defect node hardware. Therefore, as soon as great deviations have been detected multiple times one after another, the node switches to safe state.

A slave node also enters the safe state when heartbeats from the producer on master side are not sent within a defined time frame. This kind of mechanism gives the consumer on slave side the possibility to check if the timing master is still available.

6.2.2. Master Node Functionality

The master algorithm has to provide the following functionality. Firstly, it is responsible for sending response messages after receiving a request. A request message is only checked for the right safe addresses as outlined in section 5. Next it triggers a response with the same CTN value and the consumer (sender) address received. Safe address of the producer and the current network timestamp are inserted into the response message by the safe protocol stack.

Secondly, it has to trigger the regular heartbeat messages. To differentiate heartbeat messages from response messages a special reserved safe address as consumer (sender) address is used. On the slave side the heartbeat just initiates an update of the consumer table and is then discarded. The CTN value in the payload is not of importance and is set to the last received value.

6.3. Distributed Algorithm

Additionally to the centralized algorithm (see Table 1 for a summary) a second method for time synchronization is presented. As already mentioned in section 4, the idea of this concept is that all producers in the network synchronize with the connected (bound) consumer. As a consequence, only producers bound to the same consumer share the same time base as well as all consumers on the same node. In other words, multiple masters represented by multiple consumers and multiple slaves (producers) exist. If a producer is connected to numerous consumers, the producer has to manage and use different time bases for each communication path – in the following called virtual times derived from the network time of a node. Especially in a meshed network many virtual times have to be handled by the nodes.

In contrast to the centralized algorithm where network variables are applied, the EN 14908 concept of explicit messaging with explicit addressing is used when implementing the distributed algorithm. The reason is the following: For sending application related messages (message type 2 in subsection 6.1) only network variable service is used. Thus, a consumer is always equal to a network variable input (NVI) and a producer to a network variable output (NVO). As outlined in [19] and summarized in subsection 6.2, NVI can only receive, NVO just send data items. Since the consumer triggers the network time synchronization, each consumer would require additional network time synchronization NVO to send the request. The same on the producer side: each producer would require extra network time synchronization NVI to receive a request. In particular, in complex network configurations (meshed networks) an enormous amount of additional time synchronization network variables would be necessary. Consequently, a great number of LonTalk addresses have to be stored, but only an address table of limited entries is available on the EN 14908 chip.

As a prerequisite of using explicit messaging service with explicit addressing, not only the safe binding between producer on Node A and consumer on Node B must be provided, but also the LonTalk addresses of Node A must be transmitted to Node B and vice versa. Both tasks are performed by the safe network management tool [15]. Not to forget, each node must store the explicit addresses that of course consumes

memory resources on Safety Chip 1 and Safety Chip 2. Compared to the EN 14908 chip, the safety chips provide more memory resources and accordingly more addresses can be stored.

Having the aforementioned in mind, a request is triggered by the consumer. The safe protocol stack not only builds the safe message format and forwards it to the Network Access Layer, but it also gives the LonTalk address to the Network Access Layer as parameter.

Table 1 Facts about centralized and distributed algorithm

	Central	Distributed
Time source	Dedicated node	Consumer
Number of time sources	Single master	Equal to number of consumers
Memory resources	Less on slave side	High on producer and consumer side
Risk of failure	High	Low
Time synchronization	Single network time	Multiple network times
Load balance	All requests handled by master	Every node handles some requests

Each safe node in the network is running the same network time synchronization software. On the contrary to the master algorithm, no configuration of different roles at compile time is required. The synchronization procedure is very similar to that in Figure 5. The difference is that the operation is initiated by the consumer acting as timing master. The producer receives the request. It takes the timing information as the new network time. Identical to the way mentioned in subsection 6.2.1, network time is set immediately at startup time synchronization. During operation it is sped up or slowed down. Producer acknowledges the request by sending a response to the consumer. The response time is checked by the consumer which reinitiates synchronization if the check failed.

Since there are not dedicated time synchronization network variables, an incoming time synchronization message must be checked whether it is a request from a consumer or a response from a producer. Therefore the safe address (source address) is used.

- If the message is a request, safe address is not found in the consumer table. However, the payload of the message contains a safe address which is found in the producer table.
- If the message is a response, the safe address (source address) and the safe address in the payload is found in the consumer table.

Implementing distributed algorithm results in many virtual network times. As a consequence, CTN value for each producer/consumer, time of sending a request and network time synchronization interval for every consumer must be stored – further memory resources are required. In addition, each producer has to store its current timing information shared with different consumers.

The disadvantage of the distributed algorithm is first a higher communication effort increasing with the degree of meshing. Also the management of the synchronization relations yields a higher need of memory (additional parameter) as outlined in Table 1 and computing power (numerous virtual network times). The big advantage is the missing of a single point of failure. In a centralized approach a defect of the timing master results in a safe state of all nodes connected. On the contrary, a faulty timing master in the decentralized approach only causes the connections to the producer to be not working. So availability of the whole network is just affected to some extent.

7. Conclusion

The goal of the SafetyLon project is to develop a safe building automation network based on the already existing LON. All requirements of IEC 61508 safety integrity level 3 (SIL 3) are met. Thus, a transport of safe and non-safe data over the same communication channel using a safe protocol is guaranteed. Network time synchronization is of fundamental importance in this protocol. Since the LON includes devices that add delays to the transmission (e.g. routers) hazardous events such as a delay, repetition or wrong sequence of a message are possible. A safety function to detect such hazardous events is applying timestamps to every safe message. However, timestamps are only an effective means if nodes have a common understanding of the time, i.e. are time synchronized.

Each node in the network is executing a time synchronization algorithm which is based on a request/response service. Two implementations are specified: the first is based on a central approach (master/slave) and the second is based on a distributed algorithm (consumer/producer). The central algorithm uses a single master structure and the EN 14908 NV communication concept. The distributed algorithm can also be seen as a network with multiple masters, where all producers synchronize with their connected consumers. Regardless of the algorithm used, the synchronization algorithm is embedded into the Safety Layer of the software architecture and called periodically by the Safe Operating Scheduler.

The time synchronization concept was ported to the SafetyLon target hardware and is going to be subject of extensive system tests. A small test network was already set up and functionality as well as stability of network time synchronization were verified. It is becoming apparent that all SafetyLon requirements are met.

Additional aspects such as scalability of the algorithms are going to be investigated by means of a network simulator [17] which will be built based on the time discrete event triggered network simulation environment OMNET++ [16]. It focuses on synchronization errors, synchronization configuration parameters and hardware related issues under typical operating conditions.

Further investigations about the performance of the different time synchronization concepts are going to follow by extending the simulation tool and testing the integration of the time synchronization with the other software functions. More practical testing is going to examine long term stability of the complete system.

References

[1] P. Wratil, "Sicherheitsgerichtete Netzwerke – Ethernet Powerlink Safety", *PRAXIS Profiline – Vision of Automation*, Nov. 2004.

[2] T. Novak, T. Tamandl, "Architecture of a Safe Node for a Fieldbus System", in *Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN)*, Volume 1, pp. 101-106, 2007.

[3] P. Wratil, "Sichere Netzwerke – Technik und Anwendungen, Teil 1 Fehlerarten und Korrekturstrategien", *Elektronik*, Issue 21, 2005.

[4] G. Beckmann, "Die EtherCAT Sicherheitslösung", *AUTlook*, May 2007.

[5] D. Reinert, M. Schaefer (Publisher), *Sichere Bussysteme in der Automation*, Hüthig Verlag, ch. 3-4, 2001.

[6] P. Wratil, M. Kieviet, *Sicherheitstechnik für Komponenten und Systeme*, Hüthig Verlag, Heidelberg, ch. 5 and ch. 7, 2007.

[7] J. C. Eidson, "Measurement Control and Communication Using IEEE 1588", *Springer Verlag*, 2006.

[8] R. Höller. M. Horauer, G. Gridling, N. Kerö, U. Schmid, K. Schossmaier, "SynUTC – High Precision Time Synchronization over Ethernet Networks", in *Proceedings oft he 8th Workshop on Electronics for LHC Experiments (LECC'02)*, Colmar, France, pp. 428-432, 2002.

[9] T. Neagoe, V. Cristea, L. Banica, "NTP versus PTP in Computer Networks Clock Synchronization", in *Proceedings of IEEE International Symposium on Industrial Electronics*, Volume 1, pp. 317-362, 2006.

[10] "IEC 61508 – Functional safety of electric/electronic/programmable electronic safety-related systems", 1999.

[11] T. Tamandl, P. Preininger, "Online Self Tests for Microcontrollers in Safety Related Systems", in *Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN)*, Volume 1, pp. 137-142, 2007.

[12] T. Tamandl, P. Preininger, T. Novak, P. Palensky, "Testing Approach for Online Hardware Self Tests in Embedded Safety Related Systems", in *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1270-1277, 2007.

[13] Josef Börcsök, *Electronic Safety Systems*, Hüthig Verlag, Heidelberg, ch. 6.6, 2004.

[14] "EN 14908 – Open data communication in building automation, controls and building management – control network protocol", 2006.

[15] P. Fischer, M. Holz, M. Mentzel, "Network Management for a Safe Communication in an Unsafe Environment", in *Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN)*, Volume 1, pp. 131-136, 2007.

[16] A. Varga, "OMNET++ Discrete Event Simulation System User Manual", *Omnet Community*, Version 3.2, 2005.

[17] B. Sevcik, "Netzwerkzeitsynchronisation in sicheren Feldbussystemen", *M.S. thesis*, Vienna University of Technology, Institute of Computer Technology, ch. 5, 2007.

[18] "EN 14908-1 – Open data communication in building automation, controls and building management – control network protocol – Part 1: Protocol," 2006.

[19] Echelon Corporation, "ShortStack User's Guide," v2, 2002.

[20] "EN 14908-2 – Open data communication in building automation, controls and building management – control network protocol – Part 2: Twisted Pair Specification," 2006.

[21] LOYTEC, "Orion Stack Programmer's Manual", January 2005.