SEMI-AUTOMATIC COMPENSATION OF THE PROPAGATION DELAY IN FAULT-TOLERANT SYSTEMS

Thomas Losert, Wilfried Elmenreich, Martin Schlager
Institute of Computer Engineering
Vienna University of Technology
Vienna, Austria
{thomas, wilfried, smartin}@vmars.tuwien.ac.at

ABSTRACT
In control systems the jitter is a major problem since in a time-varying system the theoretical results for analysis and design of time-invariant systems cannot be used directly. Reducing the jitter increases the stability of the closed control-loop thus leading to enhanced reliability.

This paper presents a general model that can be applied to bus topologies as well as to star topologies. Based on this model an algorithm is presented that allows to improve the precision of a set of distributed clocks by measuring the propagation delay of the communication lines and compensating the jitter introduced by the propagation delay.

Some fault-tolerant architectures already provide means for coping with propagation delays but require manually entering the values in a configuration-tool. With this algorithm the system supports this error-prone task by providing validity checks for the entered values or measuring these values automatically thus rendering this maintenance step obsolete.

KEY WORDS
Hard Real-Time Systems, Communication Model, Compensation of Propagation Delay, Precision of Distributed Clocks, Fault Tolerance.

1 Introduction

In a distributed fault-tolerant application, a scenario that has to be considered is the physical destruction of a limited part of space, e. g., due to a fire on board of an aeroplane. In order to reduce the risk of a correlated malfunction and reach a failure probability on the order of 10^{-9} as required for ultra-dependable systems (see [1, p. 5]) safety-critical components should be spatially separated although this approach will increase the total length of cable. Even for non-safety-critical systems like the in-flight-entertainment system, where each seat requires a connection to the server, signals have to cover significant distances of cabling (see [2]). Substantial propagation delays must be considered for remote-controlled power plants or railroads, in satellite communication, or in submarine communication which often utilizes modulated sound instead of electromagnetic waves.

A short calculation reveals that 200 m of cable introduce a propagation delay of about 1 μs whereas the propagation delay of sound for the same distance in water is about five orders of magnitude above. Further, the propagation delay introduced by network equipment (e. g., switches) can be estimated with several microseconds. If not corrected, the propagation delay has to be considered as jitter in the real-time application, thus degrading precision in an ensemble of distributed clocks as described in [3] or stability in control applications since in a time-varying system the theoretical results for analysis and design of time-invariant systems cannot be used directly (see [4]).

Another issue is that often the geometric dimensions of the cabling are not known in advance or are subject to change during the development process. Thus, these correction terms introduce a source of error. Each parameter in a tool that has to be adjusted manually is a potential source of problems. A robust and easy configurable architecture should determine as much parameters as possible automatically and provide validity checks whenever possible. This reduces the mental complexity for the system designer and lowers development and maintenance costs since problems can be detected earlier.

According to [5] time-triggered architectures are ideally suited for the periodic operation in distributed fault-tolerant control systems. The implementation of safety critical systems like X-by-wire probably will fail without the framework of time-triggered architectures. To the best of our knowledge none of the time-triggered architectures that are available today allow the correction of the propagation delay as proposed in this paper.

Since the algorithm for measuring the propagation delay and reconstructing a model for the communication subsystem can be subdivided into several steps whereas the execution time of each step can be bounded, this algorithm can be executed periodically and the traffic can be interleaved with real-time communication. Thus the execution of this algorithm does not influence the real-time communication.

Based on the assumption that on a correct channel the variation of the propagation delay in a running system (e. g., due to variation in temperature) can be neglected, this algorithm could be used in safety critical applications during the startup or during maintenance, when the system is in a safe state, in order to perform a last check of the calibration values that have been entered manually. In uncritical applications maintenance costs can be cut down since the system is able to determine the necessary parameters automatically.

The remainder of this paper is structured as follows: Section 2 describes the communication model that is flexible enough to cover bus topologies as well as star topologies and is used as the basis for the algorithm presented in section 3 which is used for precise measuring of the propagation delay. In section 4 a simplified communication model is described that can be reconstructed based on the measurement values. Section 5 presents an efficient algorithm for compensating the propagation delay that requires just one calibration value per node for a star topology and is flexible enough to cover all communication systems that are possible in the chosen communication model. Section 6 introduces aspects of fault-tolerance and section 7 discusses some details of this algorithm whereas section 8 concludes this paper.

2 Communication Model

In this paper a communication model based on the 10 Base-5 ethernet standard ("thick ethernet", see [6, chp. 8]) is assumed. Since a distributed fault-tolerant clock consists of at least three clocks, a degenerated network consisting of a point-to-point link between two nodes is not considered in this paper.

As depicted in figure 1, a set of at least three nodes communicates with broadcast messages, i. e., each message that is sent by a correct node is received by each other correct node after an individual propagation delay. For this propagation delay an *a priori* known upper bound δ_{max} is given.

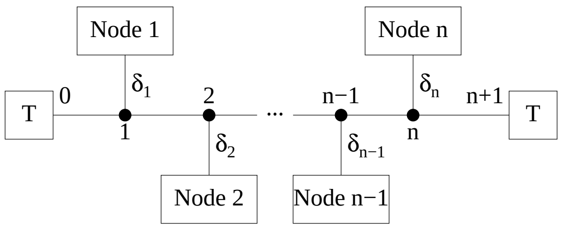

Figure 1. Model for the Propagation Delay

The *main line* is a cable with a terminator T at both ends for preventing reflections of the signal. A set of nodes N (whereas n is the number of nodes) is connected with *branch lines* to the main line. The propagation delay is independent from the direction of propagation of the signal on the medium.

A signal generated by a node propagates along its branch line, splitting up at the intersection point of this branch line with the main line and propagates along the main line towards both terminators. At each intersection point with another branch line it splits up again and is received by the node terminating this branch line.

We assume that a communication according to a collision-free media access strategy (e. g., TDMA) has been established already. Thus, the problem of collisions when two or more nodes send a frame approximately at the same instant will not be considered. Based on the *a priori* knowledge that the end-to-end delay between two arbitrary nodes is bound by δ_{max} the nodes are already synchronized and a global view of time has been established (see [7] for an example of how to reach synchronicity), but without compensation of the propagation delay the precision of the set of clocks is poor.

Without loss of generality we call one arbitrary end of the main line the *begin* and the other one the *end*. All positions of interest (i. e., begin and end as well as the intersections with the branch lines to the nodes) are labelled with ascending numbers. We refer to the propagation delay from the begin to position k of the main line with $\phi_{0,k}$. This could be imagined as sending a short impulse from the terminator at the beginning and measuring the propagation delay to this position.

The propagation delay on the main line between the intersection with the branch line of node i and node j can be calculated as

$$\phi_{i,j} = \phi_{j,i} = |\phi_{0,i} - \phi_{0,j}|.$$

We refer to the propagation delay introduced by the branch line from node i to the intersection point with the main line with δ_i. Thus, the propagation delay between Node i and Node j can be calculated as

$$\delta_{i,j} = \delta_{j,i} = \delta_i + \phi_{i,j} + \delta_j$$

and – based on the assumption stated above – is bounded with δ_{max} (i. e., $\delta_{max} \geq \delta_{i,j}\ \forall i, j \in N$).

This general model also covers two well-known topologies that are widely used: The bus topology and the star topology (in the latter the propagation delay introduced by the main line is zero).

The point in time in the TDMA schedule intended for the begin of the transmission of a message m by node i is denoted with t_i^m while $t_{snd,i}^m$ is used for the actual begin of transmission. With $t_{rcv,i,j}^m$ we denote the instant where node j starts receiving the message that has been sent by node i.

3 Measuring Delays

In a communication system as described in the previous section, that consists of three nodes, the portions of the main line can be assigned to the branch lines thus having an equivalent model in star topology, i. e., without main line (see figure 2). We call the intersection of the three communication lines of the nodes r, a, and o the center $C_{r,a,o}$ of the star and the delay from node a to this center $\delta_a^{r,a,o}$.

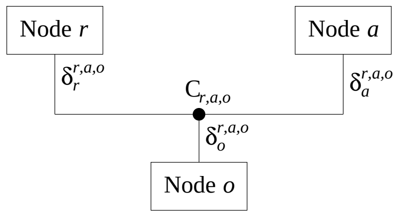

Figure 2. Measuring Delays with three Nodes

The measurement of the propagation delay is based on a calibration request *creq* of *a priori* known length that is sent by a requesting node r to an answering node a in the system. After an *a priori* known delay $\delta_{wait,a}$ (for receiving and parsing the packet) node a sends the calibration answer *cans*. The observing node remains passive (i. e., receives only) and calculates the delay $\delta_a^{r,a,o}$.

The measurement can be broken down to the following steps:

1. Node r sends a small packet with the calibration request *creq* to the network at the instant $t_{snd,r}^{creq}$.

2. The packet *creq* arrives at node a at the instant $t_{rcv,r,a}^{creq} = t_{snd,r}^{creq} + \delta_r^{r,a,o} + \delta_a^{r,a,o}$ while it arrives on node o at the instant $t_{rcv,r,o}^{creq} = t_{snd,r}^{creq} + \delta_r^{r,a,o} + \delta_o^{r,a,o}$.

3. After an *a priori* known delay $\delta_{wait,a}$ node a sends the calibration answer *cans* at the instant $t_{snd,a}^{cans} = t_{rcv,r,a}^{creq} + \delta_{wait,a}$.

4. The packet *cans* arrives at node r at the instant $t_{rcv,a,r}^{cans} = t_{snd,a}^{cans} + \delta_a^{r,a,o} + \delta_r^{r,a,o}$ while it arrives on node o at the instant $t_{rcv,a,o}^{cans} = t_{snd,a}^{cans} + \delta_a^{r,a,o} + \delta_o^{r,a,o}$.

5. Node o can calculate $\delta_a^{r,a,o}$ since $t_{rcv,a,o}^{cans} - t_{rcv,r,o}^{creq} = \delta_{wait,a} + 2 \cdot \delta_a^{r,a,o}$.

Although $\delta_a^{r,a,o}$ is already known at node o and thus node r could calculate $\delta_r^{r,a,o}$ as

$$t_{rcv,a,r}^{cans} - t_{snd,r}^{creq} = \delta_{wait,a} + 2 \cdot (\delta_r^{r,a,o} + \delta_a^{r,a,o})$$

this is inadvisable due to measurement errors that could possibly sum up. Further, most of the considerations discussed in section 7 would become invalid and additional – possibly unknown – parameters would become relevant in this case.

After the first round the nodes change their roles (node a becomes node r', node r becomes node o', and node o becomes node a') and the calibration answer *cans* is taken as calibration request *creq*$'$ for the second round, thus allowing node o' (the former node r) to calculate $\delta_{a'}^{r',a',o'} = \delta_o^{r,a,o}$. Finally, the nodes change their roles again and the calibration answer *cans*$'$ of round two is taken as calibration request *creq*$''$ for the third round, which allows node o'' (the former node a) to calculate $\delta_{a''}^{r'',a'',o''} = \delta_r^{r,a,o}$. Distribution of the measured delay can be performed as payload of the respective calibration answer since the observing node becomes the answering node in the next round.

With the algorithm outlined above $\delta_r^{r,a,o}$, $\delta_a^{r,a,o}$, and $\delta_o^{r,a,o}$ can be measured by sending four packets to the communication medium. A fifth packet is required for communicating the last result to the other nodes.

For the quality of this measurement the parameter $\delta_{wait,a}$ is crucial since it directly influences the calculation for the propagation delay. Besides from comparing the calculated propagation delay with the *a priori* known value δ_{max} or checking the measured values by a human operator, transient problems in the answering node that cause a deviation from $\delta_{wait,a}$ can be detected with a given probability by repeated measurements. This aspect is further considered in section 6.

This algorithm is not limited to the model of nodes connected with branch lines to a common main line (see previous section) or in star topology but can be generalized to other network topologies as long as the prerequisites presented in section 2 are not violated.

4 Reconstruction of a Simplified Model

Due to the limitation that only the propagation delay from one node to a particular reference point (the center $C_{a,b,c}$ of the star established by three arbitrary nodes a, b, and c of the cluster) can be measured, it is not possible to reconstruct the full model as described in section 2 unambiguously. Thus, we use the measured delays to reconstruct a simplified model. The set of simplified models is a subset of the set of full models.

In fact the network in figure 3 is equivalent to the network in figure 1 if $\delta_1' = \delta_1 + \phi_{1,2}$, $\delta_2' = \delta_2$, $\delta_{n-1}' = \delta_{n-1}$, and $\delta_n' = \delta_n + \phi_{n-1,n}$. Further the terminators do not participate in the communication and thus have

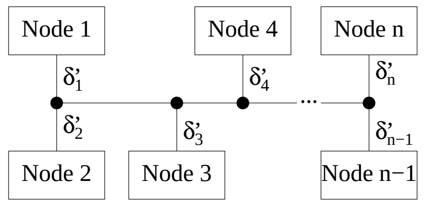

Figure 3. Simplified Model for the Network in Figure 1

been removed from the model.

In order to reduce ambiguity we mandate that the simplified model of a cluster consisting of three nodes is in star topology whereas in a cluster with more than three nodes the first two nodes and the last two nodes share the same intersection with the main line, i. e., $\phi_{1,2} = \phi_{n-1,n} = 0$.

The simplified model is generated as follows (without loss of generality the nodes are numbered from 1 to n in an arbitrary order):

1. The nodes 1, 2, and n of the set of nodes are used to reconstruct a model for these three nodes in star topology with $C_{1,2,n}$ as center as described in the previous section.

2. If another node is left (i. e., node n is not equal to node 3) node $n-1$ is integrated to the model by measuring the star with the nodes $1, 2, n-1$ and the star $1, n-1, n$. If necessary we rearrange the numbering of the nodes in the model to ensure that node 1 and 2 are at the beginning (or end) and node $n-1$ and n are at the end (or beginning).

3. Each remaining node i is integrated by measuring the star $1, i, n$ and the star $2, i, n-1$. If necessary we rearrange the numbering of the nodes in the model to ensure that node 1 and 2 are at the beginning (or end) and node $n-1$ and n are at the end (or beginning).

This allows calculation of a valid simplified model as well as adapting the model if further nodes have to be integrated later.

5 Compensation of Propagation Delay

Based on the simplified communication model as described above, this approach allows compensation of the propagation delay on the communication lines. Each node i in the cluster requires to know the following parameters:

$$\delta_i \quad \text{and} \quad \phi_{i,j}\ \forall j \in N \backslash \{i\}.$$

Each sending node i compensates the delay of message m introduced due to the branch line of node i by sending it δ_i before the intended point in time t_i^m. Thus, message m arrives at the intersection of its branch line with the main line exactly at t_i^m. Each receiving node j corrects the timestamp for the perception of the message $t^m_{rcv,i,j}$ by subtracting $\phi_{j,i} + \delta_j$ from this timestamp. This compensates the propagation delay introduced by the main line and the branch line of the receiving node (cf. figure 4).

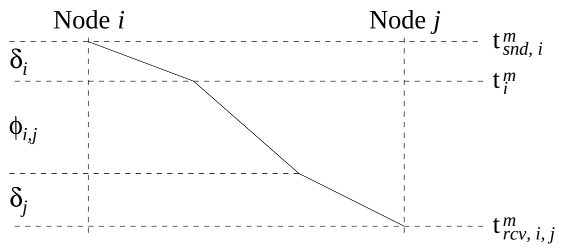

Figure 4. Compensation of Propagation Delay for message m

For star topologies, where the term $\phi_{i,j}$ equals to zero for all pairs of nodes i, j it is sufficient for each node i to know the propagation delay δ_i only. This is a highly scalable approach since the number of correction values per node does not depend on the number of nodes in the cluster. Furthermore, the measurement round as described in section 3 can be extended from three nodes to n nodes and the communication effort can be reduced to $n + 1$ messages per communication channel plus an extra message for communicating the last result.

For masking certain types of faults a central control device can be placed at the center of a network in star topology. The design of this device is simplified also, since all frames arrive at the intended instant and thus no correction terms are required in this device.

6 Fault-Tolerance Aspects

According to [8] a failure occurs when the delivered service deviates from the expected or specified service. An error is the occurrence in the system that leads to the failure and a fault is the cause of the error.

In order to apply fault-tolerance to the measurement algorithm discussed in section 3, we have to distinct between faults in the requesting node, answering node, or observing node.

Faults in the requesting node r are easy to detect since the particular instant of sending the request does not influence the result and the other nodes can detect if no correct request is sent at all. In a star topology a fault in the observing node o can be masked by having at least three observing nodes connected to the center of the same star and use voting.

While it is trivial to deal with faults in the requesting node or observing node, a fault in the answering node a (i. e., the answering node responds after a delay that differs from $\delta_{wait,a}$) cannot be detected if the overall delay is below δ_{max}. Since this fault influences the communication line to the faulty node only there is no impact on correct nodes.

Based on the assumptions on occurrence of transient faults repeated measurements and comparing the results allow to increase the possibility for detection to any desired level.

Other types of faults can be tolerated by using an architecture that provides the required degree of service even in the presence of the faulty nodes according to the desired fault-hypothesis (see [9]).

7 Discussion

In distributed control systems two parameters have major influence on the performance of the closed control loop: the latency and the jitter. Depending on the application, the latency can be compensated by applying state estimation algorithms in order to gain a similar performance as for a locally controlled device (see [10] for an example).

The jitter is a major problem since in a time-varying system the theoretical results for analysis and design of time-invariant systems cannot be used directly. Reducing the jitter is correlated with improving the performance and stability in a distributed control application. In [11] and [12] the negative effects that jitter can introduce in a control-loop are demonstrated as well as the importance of compensation.

The presented application of measuring the propagation delay of a cable for reaching higher precision in a set of distributed clocks requires an accuracy in the submicrosecond range. Although the algorithm for measuring the round-trip time as proposed in [13] might look similar to the algorithm presented in section 3 there are differences in details that allow to increase the accuracy of the measurement by elimination of possibly unknown parameters that could influence the result:

Neither $\delta_r^{r,a,o}$ (plus any additional delays in the sending path of node r) nor $\delta_o^{r,a,o}$ (plus any additional delays in the receiving path of node o) influence the measurement of $\delta_a^{r,a,o}$ as long as all delays remain constant. Since both packets, packet *creq* as well as packet *cans*, have been timestamped by the local clock of node o the current precision of the ensemble of clocks does not influence the quality of this measurement.

For even better results or fault detection several independent observing nodes could be attached to $C_{r,a,o}$ and the results further processed. Averaging allows improving the accuracy and voting allows fault-tolerance in the set of observing nodes.

8 Conclusion

This paper demonstrates how the propagation delay in a cluster of nodes can be measured. The proposed measurement method eliminates several undesired parameters by design of the measurement algorithm – thus allowing a better accuracy.

By using the measurement values a communication model of the system is reconstructed and the propagation delay of the communication channel compensated, thus, eliminating the delay that has to be considered as jitter if unknown. This allows e. g., a distributed clock with better precision or a control loop that is more stable thus improving the reliability of the whole system.

Acknowledgements

This work has been supported by the CRTC project (contract No. 808039) which is funded by the FIT-IT programme of the Austrian Federal Ministry of Transport, Innovation and Technology and by the DOC programme of the Austrian Academy of Sciences.

Further, the authors would like to thank the anonymous reviewers for valuable comments regarding this paper.

References

[1] Neeraj Suri, Chris J. Walter, and Michelle M. Hugue, editors. *Advances in Ultra-Dependable Systems.* IEEE Computer Society Press, Los Alamitos, CA, U. S. A., 1995. ISBN 0-8186-6285-9.

[2] Gerald D. Lui-Kwan. In-Flight Entertainment: The Sky's the Limit. *IEEE Computer*, 33(10):98–101, October 2000.

[3] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers, Boston, Dordrecht, London, January 1997. ISBN 0-7923-9894-7.

[4] Johan Nilsson. *Real-Time Control Systems with Delay.* Dissertation, Department of Automatic Control, Lund Institute of Technology, 1998. ISRN LUTFD2/TFRT–1049–SE.

[5] Albert Amos. Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems. In *Proceedings of the Embedded World 2004*, pages 235–252, Nuremberg, Germany, February 17–19, 2004. Available at `http://www.can.bosch.com/docu/embedded_world_04_albert.pdf`.

[6] Institute of Electrical and Electronics Engineers, New York, NY, U. S. A. *Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, 2000 edition, 2000. ISBN 0-7381-2674-8.

[7] Wilfried Steiner, John Rushby, Maria Sorea, and Holger Pfeifer. Model Checking a Fault-Tolerant Startup Algorithm: From Design Exploration To Exhaustive Fault Simulation. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2004)*, pages 171–180, Florence, Italy, June 28, – July 1, 2004.

[8] Victor P. Nelson. Fault-Tolerant Computing: Fundamental Concepts. *IEEE Computer*, 23(7):19–25, July 1990.

[9] Hermann Kopetz. On the Fault Hypothesis for a Safety-Critical Real-Time System. In *Keynote Speech at the Automotive Software Workshop San Diego (ASWSD 2004)*, San Diego, CA, U. S. A., January 10–12, 2004.

[10] Gustavo Hommerding Alt and Walter Fetter Lages. Networked Robot Control with Delay Compensation. Technical report, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, November 9–11, 2003. Available at `http://www.eletro.ufrgs.br/~fetter/rtlws03.pdf`.

[11] Pau Marti, Josep M^{a} Fuertes, Gerhard Fohler, and Krithi Ramamritham. Jitter Compensation for Real-Time Control Systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, pages 39–48, London, U. K., December 3–6, 2001. Available at `http://www.mrtc.mdh.se/publications/0321.pdf`.

[12] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. Stability of Networked Control Systems. *IEEE Control Systems Magazine*, 21(1):84–99, February 2001.

[13] David L. Mills. Internet Time Synchronization: the Network Time Protocol. RFC 1129, University of Delaware, Newark, DE, U. S. A., October 1989.