Fault-Tolerant Clock Synchronization in Ring-Networks

Klaus Echtle
ICB – Institute for Computer Science and Business Information Systems
University of Essen-Duisburg, Essen, Germany
Phone: (+49) 201-183-2352
echtle@dc.uni-due.de

Zoha Moztarzadeh
ICB – Institute for Computer Science and Business Information Systems
University of Essen-Duisburg, Essen, Germany
Phone: (+49) 201-183-4558
moztarzadeh@dc.uni-due.de

ABSTRACT
Fault-tolerant clock synchronization is an important task in many safety-relevant distributed real-time systems built by a bridge-connected network, where the bridges are connected via point-to-point links (like Time-Sensitive Network). This paper proposes a new protocol for fault-tolerant clock synchronization for not fully connected networks, in particular ring topologies with only two disjoint paths between any pair of bridges. This reduction in topological redundancy allows for much cheaper networks. The new protocol – named "ring forward and answer", RFA for short – tolerates arbitrary behavior except complementary compensation (called ABC-failure, which comes close to Byzantine behavior) of any single bridge.

CCS Concepts
• **Networks → Network protocols → Network protocol design**

Keywords
Distributed real-time systems; fault-tolerant clock synchronization; ring topology; Byzantine failure

1. INTRODUCTION
Many safety-critical real-time applications require a globally consistent clock time among the components of a distributed systems. The relative synchronization of clocks must be achieved by a fault-tolerant clock synchronization algorithm.

Different approaches are based on a master-slave structure, where a master periodically broadcasts time information. Examples are the Network Time Protocol (NTP) [2], and the standard IEEE-1588, Precision Time Protocol (PTP) [6], [7]. The master node synchronizes its time with a grand master node connected to a time reference like GPS or UTC. This type of protocol may not be adequate to guarantee safety of the whole system due to the limited coverage of master failures and the potentially distant time reference. Moreover, this method causes fault-tolerance problems even if a secondary master clock is provided as a spare. If the primary master fails by sending slightly wrong time values rather than failing silently, a globally consistent switch-over to the secondary master clock cannot be guaranteed in the presence of arbitrary failure types [8], [9].

SAC 2017, April 03 - 07, 2017, Marrakech, Morocco

@2017 ACM. ISBN 978-1-4503-4486-9/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3019612.3019864

Internal clock synchronization algorithms guarantee bounds on the maximum difference among the clocks, but the system's global time can deviate from an external clock reference. Some of these approaches [1], [3], [4], [5] use a fault-tolerant distributed convergence algorithm without any non-tolerated single point of failure. In these algorithms nodes periodically exchange time-messages containing their clock values. A single period is referred to as synchronization interval. An example of a distributed convergence algorithm is complete message exchange followed by the fault-tolerant midpoint algorithm (FTMA) [1]. FTMA relies on the hypothesis that at most f nodes are faulty and may exhibit Byzantine behavior. Then at least $n = 3f + 1$ nodes with $2f + 1$ disjoint paths between any pair of nodes are required. In FTMA each node determines the offsets to all respective other nodes. Then it discards the f lowest and the f highest offset values and calculates the arithmetic mean of the minimum and the maximum of the remaining offset values to determine the correction term of its local clock.

The provision of three disjoint paths between any pair of nodes for $f = 1$ means a costly topological demand requiring a considerable number of links. In contrast to this, ring topologies need just n connections between n nodes. This is close to the minimum $n - 1$. However, there are only two disjoint paths between any pair of nodes, which makes tolerance of Byzantine behavior impossible. Some researchers propose an improvement of the robustness by integration of IEEE-1588 with network redundancy protocols like RSTP [11] or high-availability seamless redundancy (HSR) [10]. RSTP is not suitable for real-time applications. On occurrence of a link failure or a node failure, the network must be reconfigured to rebuild the logical path. HSR is a redundancy protocol that provides duplicated frames for separate physical paths. It tolerates a single message loss (not arbitrary behavior) by sending data in each direction of the ring. Since modern network bridges own clocks we provide a solution to fault-tolerant clock synchronization in a bridge-connected network| where a ring (or a set of rings) can be embedded statically. As is natural for rings, only two disjoint paths are guaranteed between any pair of bridges. Nevertheless the proposed protocol is resilient to one faulty bridge which may exhibit an ABC-failure. This type of failure only slightly deviates from a Byzantine failure, because we only exclude the rare case where two different types of very special malfunctions occur during two subsequent protocol actions (see section 2.2 for details). Arbitrary clock errors and loss, corruption, delay, and misdirection of messages are fully tolerated.

2. PROTOCOL DEVELOPMENT

2.1 Protocol Structure
The proposed protocol, called "ring forward and answer" (RFA), is based on the forwarding of time-messages along a ring and sending answer-messages back. RFA is composed of multiple executions of the sub-protocols "Forward protocol" (FP) and "secondary

forward protocol" (SFP), where SFP is executed only in the faulty case.

In FP an initiator bridge sends its time by a *time-message* clockwise along the ring. The other bridges along the ring (called intermediate bridges) send confirming *answer-messages* counter-clockwise back to the initiator. Based on the answer-messages the initiator recognizes if the clockwise time information has reached the bridges correctly. If not, the initiator starts SFP by sending its time information counterclockwise in a *replacement-message*. This way an initiator can reach all bridges along a fault-free path, even in the presence of a faulty bridge – either clockwise by FP or counterclockwise by SFP, see Figure 1.

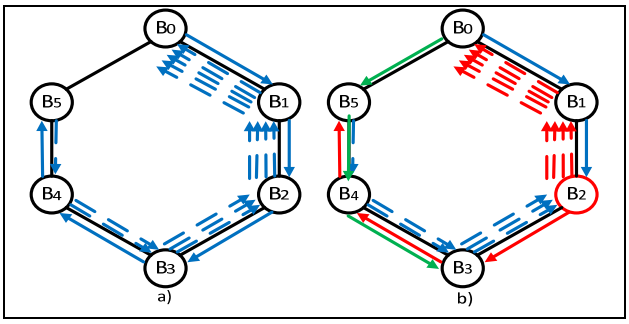

Figure 1: a) shows a protocol execution in the fault-free case, where B_0 is the initiator. Blue solid lines depict the time-message, blue dashed lines depict answer messages, sent back by each intermediate bridge. b) shows the case where faulty bridge B_2 distorts the information of the time-message (red lines) and answer-messages (red-dashed lines). Then the initiator sends a replacement-message (green lines) to those bridges, which have not received the correct information.

During protocol execution all bridges form a local vector composed of offset values (the differences between the local clock and the received clock values). At the end of a synchronization interval all bridges run FTMA on their offset vector to determine the correction term for adjusting their local clocks. According to the known properties of FTMA, $m \geq 3f + 1$ bridges have to act as initiator. Since $f = 1$ for a ring topology, we need at least 4 initiators, where any 4 bridges may take that role. However, for more initiators, ideally $m = n$, synchronization is more accurate. Each initiator launches an individual FP, where all of FPs run concurrently. Consequently, all bridges take part in a number of concurrent FPs (and SFPs if necessary).

RFA uses signatures for message authentication. This allows the bridges to conclude the path of the information flow by time- and answer-messages. All bridges digitally signs any outgoing message, denoted by

$T^{initiator\_id}_{sender\_id}$ for time-messages

$A_{signatures,\ sender\_id}$ for answer-messages, and

$R^{initiator\_id}_{sender\_id}$ for replacement-messages,

where B_{sender_id} is the respective sender and $:B_{sender\_id}$ is its signature. The signatures help to uniquely separate the messages belonging to different FPs. Otherwise the FTMA could be tricked by a faulty bridge that could appear as multiple bridges and thus unjustifiably dominate the fault-tolerant-midpoint decision. The signatures used for clock synchronization should be different from other signatures to prevent confusion. The signatures must withstand technical faults, not human attacks. Hence, they can be rather short to limit the computational overhead (16, 32 or 64 bits).

Adding the signature, the time t_j and the delay d_j to a message x by bridge B_j is expressed by the extension function $f_j(x) := (x, t_j, d_j)$. The signed messages for the case where B_0 is the initiator are described as follows and shown in Figure 1:

- B_0 sends time-message $T^0_0 := (t_0):B_0$ containing its local time t_0 to its neighbor B_1.
- The intermediate bridges receive the time-message to calculate the local offset value. When forwarding the time-message, an intermediate bridge measures its local delay, and inserts it into the time-message, to allow for delay compensation. A small deviation θ from the true delay is caused by the non-perfect accuracy of delay measurement. An intermediate bridge also sends an answer-message counterclockwise to be forwarded until it reaches the initiator. The answer-message contains the local time of the bridge and the measured delay.

  In Figure 1 B_1 receives T^0_0 and modifies T^0_0 by the extension function f_1. The result $T^0_1 := f_1(T^0_0) := (((t_0):B_0),t_1, d_1):B_1$ is sent clockwise to the adjacent neighbor B_2. Simultaneously the answer-message $A_1 := f_1(T^0_0) := (((t_0):B_0),t_1, d_1):B_1$ is sent back to B_0. Later on, B_1 will also forward the answer-messages created by B_2, B_3, etc.
- B_2 acts similar to B_1 by sending $T^0_2 := f_2(T^0_1)$ to B_3 and $A_2 := f_2(T^0_1)$ to B_1. Later on, B_2 will forward the answer-messages on their paths $B_3 \rightarrow B_2 \rightarrow B_1 \rightarrow B_0$, $B_4 \rightarrow B_3 \rightarrow B_2 \rightarrow B_1 \rightarrow B_0$, $B_5 \rightarrow B_4 \rightarrow B_3 \rightarrow B_2 \rightarrow B_1 \rightarrow B_0$, etc. B_3, B_4 etc. act accordingly.

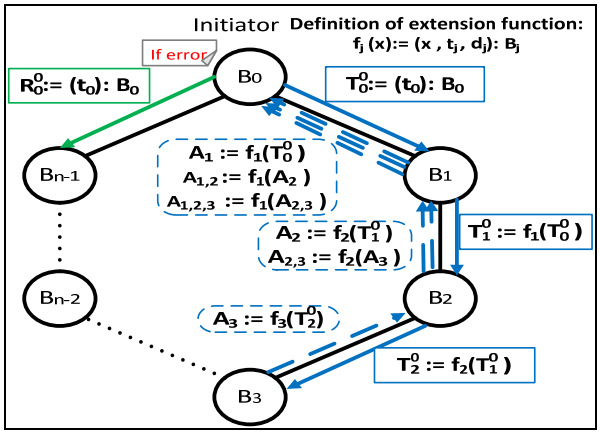

Figure 2: An example of protocol execution

- The initiator checks the incoming answer-messages. If any answer-message is missing, or the sum of the delays in the message exceeds the bound δ, the initiator initiates SFP by sending its current time t_0 in a replacement-message $R^0_0 := (t_0):B_0$ counterclockwise, see Figure 2. The value of δ is twice the maximum propagation delay in the ring (for both time and answer-messages), multiplied with the maximum inaccuracy τ of the local delay measurements $\delta = 2 \cdot h \cdot \tau$. (1), where h is the number of hops: $h = n - 1$. The initiator determines how far a replacement has to be forwarded along the ring in

counterclockwise direction. It must be forwarded up to the first bridge that made an inconsistent entry to its answer message (a consistency check of the local times and delays found in the message). When forwarded along the ring a replacement-message does not cause generation of answer messages.

2.2 Assumption of non-compensating failure

Whenever an answer-message is received by the initiator, it is validated by checking whether the message is syntactically correct and the absolute value of the time in the message plus the sum of the delays in the message is smaller than the bound δ. The subsequent decision whether or not to initiate SFP could be false without an additional assumption to exclude a rare, yet possible case: When a faulty bridge sends a message clockwise in the ring and distorts the timing information by +u, then it must not distort the timing of a counterclockwise message by –u (or approximately –u), see Figure 3. In other words: A faulty bridge does not compensate its own error by a complementary error in the opposite direction. This means, our protocol tolerates arbitrary behavior except complementary compensation (ABC-failure).

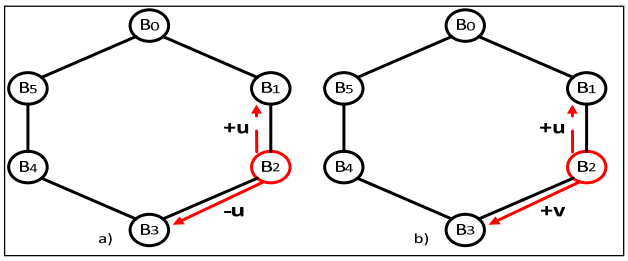

Figure 3: Assumption of the protocol. a) is excluded. b) is allowed where u+v !≈ 0. B_2 is faulty and all other bridges are fault-free.

2.3 Fault Tolerance

In the following some example fault cases are discussed:

- Fail-silence and fail-omission will be recognized by the initiator, and tolerated by launching SFP. Fail-silence or fail-omission of the initiator itself will be tolerated locally in each bridge by FTMA, see Figure 4 a) and b).
- Message corruption is detected by the signature check. A corrupted message is ignored, and tolerated like omission.
- If a faulty bridge inserts wrong delay information, the initiator will be able to detect the failure by comparing the time information in the message with its local clock time. Similar comparisons performed by the intermediate bridges will also reveal the failure. Note that, according to the failure assumption, "self-compensating" modifications of a clockwise and a counterclockwise message are excluded.

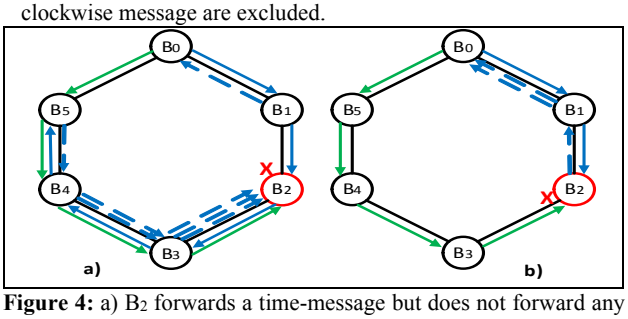

Figure 4: a) B_2 forwards a time-message but does not forward any answer-message. The initiator B_0 gets an answer-message from B_1 but it misses the answer-messages of the further bridges. in b) B_2 stops forwarding the time-messages.

2.4 Timing

Initiator's timeouts $T_{nextSync}$, T_{FP} and $T_{protocol}$ for RFA are set simultaneously when it starts FP by sending a time-message. T_{FP} is the maximum duration of FP (maximum transmission duration of all time-messages and answer-messages). After all answer-messages have been received T_{FP} is reset. If T_{FP} expires when still some answer-messages are missing, the initiator starts SFP. $T_{protocol}$ is set to a point in time later than the maximum duration of both FP and SFP. When $T_{protocol}$ expires, the initiator executes the FTMA on its offset vector and sets its clock accordingly. Timeout $T_{nextSync}$, defines the point in time when the next execution of RFA has to be started (in the subsequent synchronization interval). Intermediate bridges do not apply timeouts.

3. ANALYSIS OF THE PROTOCOL

The analysis of the protocol's temporal behavior will be based on the variables listed in Table 1.

Table 1: Parameters for RFA analysis

ρ	Maximum value of the drift.
α	Clocks' difference at the beginning of a sync interval.
β	Max. Clocks' difference at the end of a sync interval.
δ	difference by which two bridges observe some clock
θ	inaccuracy of the local delay measurements
τ	Max. of the inaccuracy of the local delay measurements
n	Number of bridges
h	Number of hops, when a single message is forwarded along the ring: $h = n - 1$ for a complete round in the ring
k	Number of rounds (= duration a message needs to be forwarded around the ring)

The duration $T_{nextSync}$ from one clock synchronization to the next is the sum of the following durations:

- β to ensure that the next protocol execution does not start before all fault-free bridges have corrected their clocks
- T_{wait}, an additional time to prevent too frequent synchronizations for performance reasons (at the cost of worse synchronization accuracy)
- $T_{protocol}$, the duration of a synchronization protocol execution. This duration is the sum of
  - T_{timeFP}, the time for forwarding time-messages in FP
  - $T_{answerFP}$, the additional time for returning the answer-messages to the respective initiators
  - $T_{timeSFP}$, the time for forwarding replacement-messages in SFP

  These three durations are equally long: $h \cdot T_{trans} \cdot (1 + \rho)$ where T_{trans} is the duration of a message transmission between two neighboring bridges. The factor $(1 + \rho)$ expresses the fact that the timeliness of message forwarding along the ring may be checked by timeout based on a slow clock. For RFA the parameter $k = 3$, because a protocol execution consists of 3 sequential rounds. We obtain:

$$T_{nextSync} = 2\beta + T_{wait} + khT_{trans}(1 + \rho) \quad (2)$$

Depending on the maximum clock drift ρ and the interval duration $T_{nextSync}$, the following holds (factor 2 because the drift can be either positive or negative):

$$\beta = \alpha + 2\rho T_{nextSync} \quad (3)$$

FTMA guarantees convergence of ½, plus the difference in time by which two fault-free bridges observe some clock. According to (1) this difference is limited by $\delta = 2h\tau$. Thus we obtain:

$$\alpha = \frac{1}{2}\beta + \delta \quad (4)$$

Substituting $T_{nextSync}$ in (3) by equation (2) results in:

$$\beta = \alpha + 4\rho\beta + 2\rho T_{wait} + 2k\rho(1+\rho)hT_{trans} \quad (5)$$

Substituting α in (5) by equation (4) yields:

$$\beta = \frac{1}{2}\beta + \delta + 4\rho\beta + 2\rho T_{wait} + 2k\rho(1+\rho)hT_{trans} \quad (6)$$

By resolving the equation to β we obtain the primary achievement of the protocol:

$$\beta = \frac{\delta + 2k\rho(1+\rho)hT_{trans} + 2\rho T_{wait}}{\frac{1}{2} - 4\rho} \quad (7)$$

An approximation can be derived by setting $\frac{1}{2} - 4\rho \approx \frac{1}{2}$ and $(1+\rho) \approx 1$ as well as substituting δ by $\delta = 2h\tau$:

$$\beta = 4h\tau + 4k\rho hT_{trans} + 4\rho T_{wait} \quad (8)$$

As can be easily seen, this expression is linear in all four (as has been expected): ρ, T_{trans}, T_{wait} and h.
For RFA we set $k = 3$ and obtain

$$\beta = 4h\tau + 12\rho hT_{trans} + 4\rho T_{wait} \quad (9)$$

Since any fault-tolerant clock synchronization (regardless of the concrete protocol) requires at least k = 1, we obtain:

$$\beta = 4h\tau + 4\rho hT_{trans} + 4\rho T_{wait} \quad (10)$$

for an ideal protocol. Note that "half rounds" are not sufficient when the faulty node is located close to the initiator. Even an "ideal" synchronization protocol cannot achieve a maximum clock deviation better than formula (10).

4. SIMUALTION

To compare the deviation of the clocks reached different scenarios have been simulated with varying the clock drift from the interval [0.0002 to 0.2] and failure types (wrong-content, delay, wrong-content, fail-silence). The following diagram shows the maximum observed value of clock deviation β (blue-solid line) compared to the upper bounds calculated for RFA according to formula (9) (blue-dashed line) and for an ideal protocol according to formula (10) (red-dashed line) obtained by 100.000 runs per combination of parameter values for a ring with $n = 5$ bridges, all of which act as initiator ($m = n$). The calculated worst-case ($k = 3$) is not significantly worse than the calculated worst-case of an ideal protocol ($k = 1$), if the clock drift is rather bad (A clock drift of 2% is very extreme). Simulations have shown that the worst-case occurs rarely, even in the presence of a fault. This holds for all investigated malfunctions of the faulty bridge. The maximum of β detected by 100.000 synchronizations is smaller than the calculated bound by a factor in the range of 4 to 5.

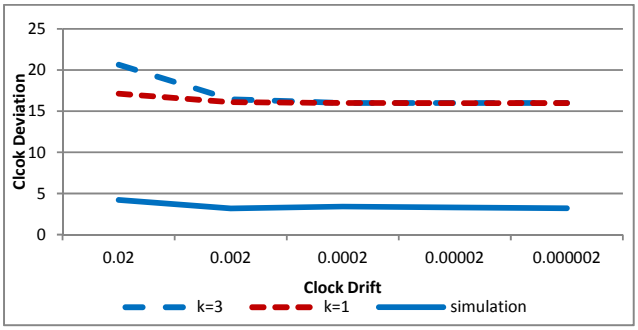

Figure 5: Maximum difference β between the clocks depending on the clock drift.

5. CONCLUSION

The new fault-tolerant clock synchronization protocol combines two advantages: full fault tolerance and low topological requirements. It can be applied to not fully connected networks, especially ring networks. The protocol achieves synchronization in the presence of an ABC-failure, which comes close to a Byzantine failure.

6. REFERENCES

[1] J. L. Welch and N. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," *Information and Computation,* Vol. 77, No. 1, April 1988, pp. 1-36.

[2] D.L. Mills, "Internet time synchronization: The network time protocol," IEEE Trans. Comm., Vol. 39, pp. 1482–1493, Oct. 1992.

[3] S. Manhaney nad F. Schneider, "Inexact Agreement: Accuracy, Precision, and Graceful Degradation," *Proceedings of the 4th ACM SIGACT-SIGOPS Symposium on Princliples in Distributed Computing,* August 1985, pp. 237-249.

[4] H. Kopetz and W. Ochsenreiter, "Clock Synchronization in Distributed Real Time Systems," *IEEE Transactions on Computers,* Vol. C 36, No. 8 August 1987, pp 933-940.

[5] M. Pflugegl and Blough, "A New and Improved Algorithm for Fault Tolerant Clock Synchronization," *Journal of parallel and Distributed Computing*, Vol. 27, 1995, pp 1-14.

[6] J.C. Eidson, M. Fischer, J. White, IEEE-1588 standard for a precision clock synchronization protocol for network measurement and control systems, in: Proceedings of the 34th Annual Precise Time and Time Interval Meeting, IEEE, 2002, pp. 243–254

[7] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "IEEE 1588-based synchronization system for a displacement sensor network," *IEEE Trans. Instrum. Meas.*, Vol. 57, no. 2, , Feb. 2008, pp. 254–260.

[8] G. Gaderer, P. Loschmidt, and T. Sauter, "Improving fault tolerance in high-precision clock synchronization," *IEEE Trans. Ind. Informat.*, Vol. 6, no. 2, pp. 206–215, May 2010.

[9] G. Gaderer, S. Rinaldi, and N. Kero, "Master failures in the precision time protocol," in *Proc. IEEE ISPCS*, Sep. 22–26, 2008, pp. 59–64.

[10] Kirrmann, H.; Kleineberg, O. Seamless and Low-Cost Redundancy for Substation Automation Systems (High Availability Seamless Redundancy HSR). In Proceedings of the IEEE Power and Energy Society General Meeting, San Diego, USA, 24–29 July 2011; pp. 1-7.

[11] The Institute of Electrical and Electronic Engineers, " ANSI/IEEE Std 801.2D, Media Access Control (MAC) Bridges", 2004