An Algorithm for Fault-Tolerant Clock State&Rate Synchronization

Klaus Schossmaier and Bettina Weiss
Technische Universität Wien
Department of Automation
Treitlstraße 1, A-1040 Vienna, Austria
{kmschoss, bw}@auto.tuwien.ac.at

Abstract

We propose a fault-tolerant algorithm for synchronizing both state and rate of clocks in a distributed system. This algorithm is based on rounds, uses our fault-tolerant Optimal Precision (OP) convergence function as the means of synchronization, and maintains a collection of intervals to keep track of real-time, internal global time, and clock rates. The analysis shows that the interlocking between state and rate synchronization can be easily solved, and that oscillator stabilities together with the transmission delay uncertainties of packets predominate the internal synchronization. In addition, average case results gathered from simulation experiments with our SimUTC toolkit prove to be about one order of magnitude better than the worst case ones from the analysis of our state&rate algorithm.

1. Introduction

Many services in a distributed system depend on a reliable and accurate time distribution, which is accomplished by algorithms for synchronizing the clocks in the ensemble. It was also discovered in [5] that several services can be implemented using synchronized clock rates (high-consonance) rather than synchronized clock states (high-precision). Of course, the latter is more powerful, but it relies significantly on the former, which can be exploited for improvement. However, almost all clock synchronization approaches focus solely on the state of clocks leaving behind the possibility to influence the rate as well.

In our project SynUTC[1], we pay attention to this state&rate synchronization problem, since it turned out that —beyond enhancements concerning the hardware of a LAN-based distributed system— an advanced algorithmic approach is indispensable to achieve a fault-tolerant synchronization in the μs-range. Therefore we embarked on the development of algorithms for synchronizing clock rates and combined them with existing ones for clock states.

[1]This research received support from the Austrian Science Foundation (FWF) grant P10244-ÖMA, the OeNB "Jubiläumsfonds-Projekt" 6454, the BMWV research contract Zl.601.577/2-IV/B/9/96, and the Austrian START programme Y41-MAT. For further reference see http://www.auto.tuwien.ac.at/Projects/SynUTC

2. Motivation

We begin our investigation by studying a simulation experiment on the classical round-based Fault-Tolerant Averaging (FTA) clock synchronization algorithm [7] conducted with our SimUTC toolkit, see [21] and [22]. The simulated distributed system consists of $n = 5$ nodes that communicate via a broadcast-type LAN whose average packet transmission delay $\delta = 60\ \mu$s with maximum uncertainties $\epsilon^-_{\max} = 10\ \mu$s and $\epsilon^+_{\max} = 20\ \mu$s. Each node p hosts a clock $C_p(t)$, where the respective average drifts ρ_p are 0 ppm, 0.5 ppm, -0.5 ppm, 2 ppm, and -1 ppm. Section 3 will provide definitions for the above parameters. A representative segment of the simulation run is drawn in the four plots of Figure 1, where simulation time t flows from left to right.

The upper left plot depicts the *(traditional) accuracy* $\alpha_p(t) = C_p(t) - t$ of clocks p, and the lower left plot the *precision* $\pi(t) = \max_{p,q}\{C_p(t) - C_q(t)\}$ of the ensemble. Both plots exhibit the expected sawtooth-like shape, which clearly illustrates the periodic resynchronizations every $P_S = 12$ s. Achieving (small) bounds on $\alpha_p(t)$ and $\pi(t)$ despite a limited number of faults refers to the well-known *clock state synchronization* problem. We emphasize "state" since clock readings $C_p(t)$ are the relevant quantities.

In addition to the state, however, the *rate* $v_p(t)$ of a clock $C_p(t)$ can be easily obtained by taking the first derivative of it, hence $v_p(t) = dC_p(t)/dt$. Using this notion, the upper right plot of Figure 1 depicts the *(rate) drift* $\theta_p(t) = v_p(t) - 1$ of clocks p, and the lower right plot the *consonance* $\gamma(t) = \max_{p,q}\{v_p(t) - v_q(t)\}$ of the ensemble. Observe that the rate drift of a clock swings around its average drift, and the consonance around the maximum difference

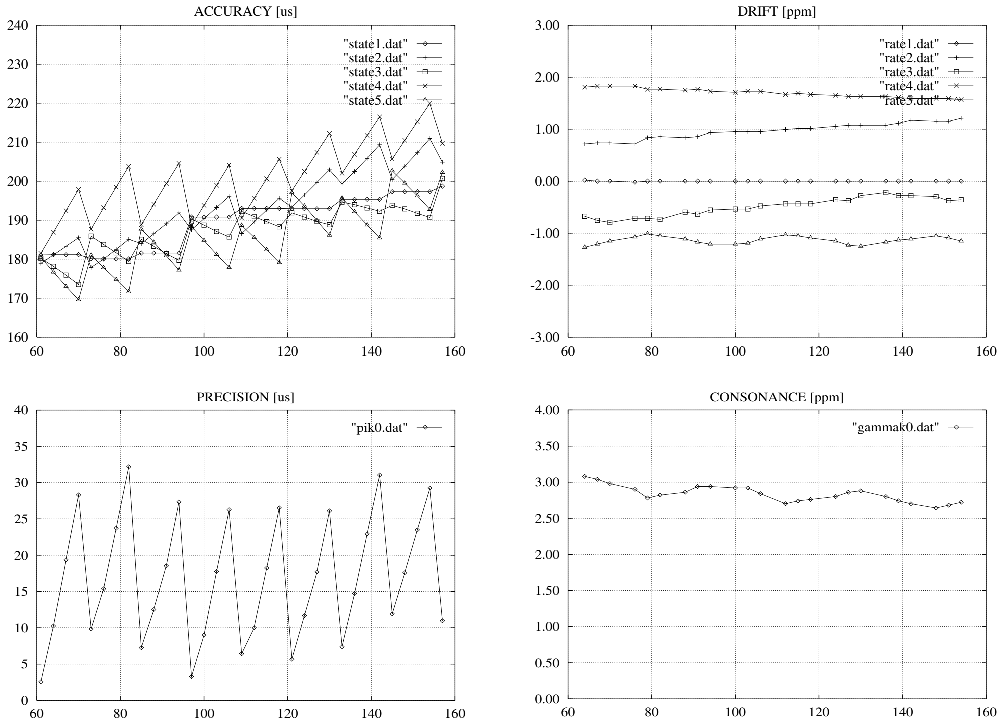

Figure 1. Traditional Clock State Synchronization.

among them, thus $\theta_p(t) \approx \rho_p$ and $\gamma(t) \approx \max_{p,q}\{\rho_p - \rho_q\}$. Intervening clock adjustments are suppressed, otherwise spikes would impair the plot. It should be clear that an ensemble of free-running clocks entails the same kind of plots on the right-hand side of Figure 1.

How can we improve state synchronization? The clock accuracy $\alpha_p(t)$ can only get under control when access to some reference time is available, e.g., via GPS receivers, leading to the topic of *external clock synchronization*. In our example the accuracies start around 180 μs and grow to 200 μs. Still, it would be desirable to have an on-line measure how far off the clocks are with respect to real-time; we return to that issue in Section 3.2.

The precision $\pi(t)$ of an ensemble is either dominated by the transmission delay uncertainties $\epsilon^{\pm}_{\max}$ or by the clock drifts ρ_p, which can be balanced by the resynchronization period P_S. In our example the latter is mainly responsible for the maximum of 33 μs. Furthermore, we can observe that precision $\pi(t)$ remains bounded regardless of the common increase/decrease of accuracies $\alpha_p(t)$. According to [6], nothing can be done to beat the precision arising from the transmission delay uncertainties, except by shrinking them with sophisticated timestamping techniques. Our solution [11] for that subproblem inserts a timestamp on-the-fly into the memory holding a packet, hence sojourn times in waiting queues do not contribute to delay uncertainties.

From the above discussion there remains only one possibility to further reduce the precision, namely by synchronizing the clock rates. In analogy to the accuracies, clocks may have large rate drifts $\theta_p(t)$, but the common deviation does not matter, it is the consonance $\gamma(t)$ that drives the clocks apart! Our simulation experiment features a consonance around 3 ppm, which determines the gradient in the precision plot. Achieving (small) bounds on $\theta_p(t)$ and $\gamma(t)$ despite a limited number of faults introduces the sister problem to clock state synchronization, coined as *clock rate synchronization*. Now we emphasize "rate" since clock advancements $C_p(t_2) - C_p(t_1)$ during $t_2 - t_1$ are of interest.

In [14] we developed a framework for clock rate synchronization, but the proposed algorithm lacks of a particular convergence function and the interlocking between clock state and rate synchronization is not taken into account. Both deficiencies are solved for the most part in [15], which is used as the basis for our expositions here. As a preview, see the plots in Figure 4 on what we can accomplish by repeating the same simulation experiment with an algorithm that combines state and rate synchronization, hence called *state&rate algorithm*.

The paper is organized as follows: Section 3 provides the building blocks for state&rate synchronization along with the system assumptions, which are all expressed by intervals. Section 4 contains the construction of our algorithm including an evaluation by using discrete event simulation.

3. Background

Our clock synchronization approach relies on an interval-based paradigm inspired by [8] and [4]. We use bold letters like $\boldsymbol{I}$ to denote an ordinary interval $\boldsymbol{I} = [x, y]$, $x \leq y$, with lower and upper *edge* x and y, respectively. An *(ordered) set* of intervals is denoted by a calligraphic bold letter like $\boldsymbol{\mathcal{I}}$. Moreover, a distinguished *reference point* I partitions the interval into a *negative part* ι^- and a *positive part* ι^+. To that end, we introduce the notation of an *asymmetric* interval

$$\boldsymbol{I} = [I \pm \boldsymbol{\iota}] = [I - \iota^-, I + \iota^+] \tag{1}$$

with length interval $\boldsymbol{\iota} = [-\iota^-, \iota^+]$. An interval with swapped parts is written as $\bar{\boldsymbol{\iota}} = [-\iota^+, \iota^-]$. Operations on these intervals are straightforward and will be defined as we encounter them.

3.1. Oscillator-Clock Model

Each node p in our distributed system hosts a *clock* $C_p(t)$ that is paced by an *oscillator* with nominal frequency f_p. Due to influences like temperature, aging, or pressure, the instantaneous frequency $f_p(t)$ of an oscillator deviates slightly from the specified f_p. Such an oscillator is usually directly coupled to the clock by a simple hardware counting device, thus clock rate $v_p(t)$ becomes

$$v_p(t) = S_p f_p(t), \tag{2}$$

where *coupling factor* S_p is the constant $1/f_p$. The distinction between oscillator and clock is the hook to introduce a rate adjustable clock, since even though $f_p(t)$ changes, factor S_p should be tuned in such a way that clock rate $v_p(t)$ remains approximately constant. Obviously, the realization of such a clock needs to provide some means to modify S_p like our UTCSU-ASIC [17].

Definition 1 (oscillator drift) *Each clock $C_p(t)$ is driven by an oscillator with instantaneous frequency $f_p(t)$ subject to the* drift condition

$$f_p(t)/f_p \in [1 \pm \boldsymbol{\rho}_p] \qquad \forall\, t \geq t_0, \tag{3}$$

where f_p is the nominal frequency and $\boldsymbol{\rho}_p = [-\rho_p^-, \rho_p^+]$ the drift interval of the oscillator.

Unfortunately, the above drift condition does not consider the fact that oscillators change their frequency very reluctantly, which is crucial for rate resynchronization. Oscillator characterizations like [19] gave reason to the supplemental assumption that the relative frequency change $f_p(t_2)/f_p(t_1)$ can be bound with a linear envelope for reasonable durations $t_2 - t_1 \geq 0$.

Definition 2 (oscillator stability) *Each clock $C_p(t)$ is driven by an oscillator with instantaneous frequency $f_p(t)$ subject to the* stability condition

$$f_p(t_2)/f_p(t_1) \in [1 \pm \boldsymbol{\sigma}_p(t_2 - t_1)] \qquad \forall\, t_1 \leq t_2, \tag{4}$$

where $\boldsymbol{\sigma}_p = [-\sigma_p, \sigma_p]$ is the stability interval of the oscillator.

To give an example, a typical temperature compensated quartz oscillator (TCXO) has a drift around 1 ppm and a stability around 0.001 ppm/s. In our simple clock model we do not consider effects due to clock granularity and discrete rate adjustments, see [12] for an extension. In addition we define uniform bounds $\rho_{\max} = \max_p\{\rho_p^-, \rho_p^+\}$ and $\sigma_{\max} = \max_p\{\sigma_p\}$.

3.2. Accuracy and Rate Intervals

The idea of the interval-based approach introduced in [8] is to represent time information by an *accuracy interval* $\boldsymbol{A}_p(t)$. In practice, such intervals are maintained by an ordinary clock $C_p(t)$ in conjunction with a time-dependent interval $\boldsymbol{\alpha}_p(t) = [-\alpha_p^-(t), \alpha_p^+(t)]$ of *accuracies* taken relatively to the clock's value, hence $\boldsymbol{A}_p(t) = [C_p(t) \pm \boldsymbol{\alpha}_p(t)]$.

Definition 3 (accuracy interval) *Let $C_p(t)$ be the clock at node p and $T_0 = C_p(t_0)$ its state at real-time t_0. The accuracy interval $\boldsymbol{A}_p = [T_0 \pm \boldsymbol{\alpha}_p]$ with $\boldsymbol{\alpha}_p = [-\alpha_p^-, \alpha_p^+]$ is* accurate *at t_0 if*

$$t_0 \in \boldsymbol{A}_p. \tag{5}$$

The rate $v_p(t)$ of a clock cannot be observed directly, but similar to an accuracy interval we can introduce a *rate interval* $\boldsymbol{R}_p(t)$ with reference point $R_p(t)$ and *inverse rate drifts* $\vartheta_p^-(t)$ and $\vartheta_p^+(t)$ as its lengths to represent rate information. Such intervals, in particular, *local* rate intervals with reference point $R_p(t) = 1$, are maintained by an algorithm.

Definition 4 (rate interval) *Let $C_p(t)$ be the clock at node p and $v_p(t) = dC_p(t)/dt$ its rate at real-time t. The rate interval $\boldsymbol{R}_p = [R_p \pm \boldsymbol{\vartheta}_p]$ with $\boldsymbol{\vartheta}_p = [-\vartheta_p^-, \vartheta_p^+]$ is* correct *during real-time interval $[t_1, t_2]$ if*

$$1/v_p(t) \in \boldsymbol{R}_p \qquad \forall\, t_1 \leq t \leq t_2. \tag{6}$$

Each node p maintains an accuracy interval $\boldsymbol{A}_p(t)$, which should remain accurate to ensure the convenient accuracy estimation $|\alpha_p(t)| \leq \max\{\alpha_p^-(t), \alpha_p^+(t)\}$. Due to the uncertainty of clock drifts, accuracy intervals need to be enlarged (also called deteriorated) accordingly. This *drift compensation* is traditionally done with the fixed drift interval $\boldsymbol{\rho}_p$ taken from the oscillator, but the dynamic local rate interval $\boldsymbol{R}_p(t)$ provides us with current information about the clock drift, since drift $|\theta_p(t)| \leq \max\{\vartheta_p^-(t), \vartheta_p^+(t)\}$ by neglecting second order terms.

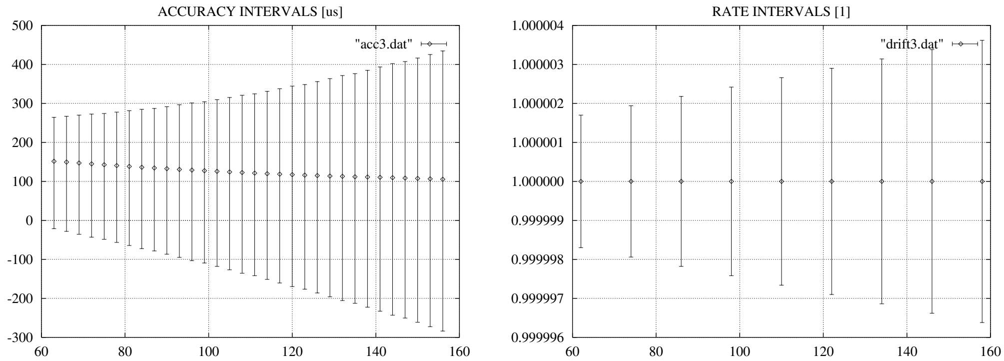

Figure 2. Deterioration of Accuracy and Rate Intervals.

Lemma 1 (drift compensation of accuracy intervals) *Let $C_p(t)$ be the clock at node p and $T_1 = C_p(t_1)$ resp. $T_2 = C_p(t_2)$ its state at real-time t_1 resp. t_2 with $t_1 \le t_2$. If accuracy interval $\boldsymbol{A}_p = [T_1 \pm \boldsymbol{\alpha}_p]$ is accurate at t_1 and local rate interval $\boldsymbol{R}_p = [1 \pm \boldsymbol{\vartheta}_p]$ is correct during $[t_1, t_2]$, then accuracy interval*

$$\boldsymbol{A}_p' = \boldsymbol{A}_p + (T_2 - T_1)\boldsymbol{R}_p \tag{7}$$

is accurate at t_2, when no resynchronization occurred during $[t_1, t_2]$.

Proof Obviously, if $t_1 \in \boldsymbol{A}_p$ then $t_2 \in \boldsymbol{A}_p + (t_2 - t_1)$, but the real-time duration $t_2 - t_1$ cannot be observed directly. Due to Definition 4 the correctness of the local rate interval $\boldsymbol{R}_p$ asserts that $1/(1+\vartheta_p^+) \le v_p(t) \le 1/(1-\vartheta_p^-)$ for any $t \in [t_1, t_2]$. Integrating this inequality from t_1 to t_2 and recalling the definition of clock rate $v_p(t)$ provides

$$\frac{t_2 - t_1}{1 + \vartheta_p^+} \le T_2 - T_1 \le \frac{t_2 - t_1}{1 - \vartheta_p^-}, \tag{8}$$

which delivers the desired bounds upon $t_2 - t_1$ as interval $(T_2 - T_1)\left[1 \pm [-\vartheta_p^-, \vartheta_p^+]\right]$. A final substitution yields $t_2 \in \boldsymbol{A}_p + (T_2 - T_1)[1 \pm \boldsymbol{\vartheta}_p]$ which confirms the lemma. □

Essentially, we have shown that an accuracy interval $\boldsymbol{A}_p$ can be accurately maintained with a correct local rate interval $\boldsymbol{R}_p$. The rate intervals themselves can be kept correct if they are deteriorated with the stability interval $\boldsymbol{\sigma}_p$, hence called *stability compensation*.

Lemma 2 (stability compensation of rate intervals) *Let $C_p(t)$ be the clock at node p and $T_1 = C_p(t_1)$ resp. $T_2 = C_p(t_2)$ its state at real-time t_1 resp. t_2 with $t_1 \le t_2$. If local rate interval $\boldsymbol{R}_p$ is correct at t_1, and $\boldsymbol{\sigma}_p$ is the stability interval of the oscillator, then*

$$\boldsymbol{R}_p' \doteq \boldsymbol{R}_p + [0 \pm (T_2 - T_1)\boldsymbol{\sigma}_p] \tag{9}$$

is correct at t_2, when no resynchronization occurred during $[t_1, t_2]$.

Proof The correctness of local rate interval $\boldsymbol{R}_p = [1 \pm \boldsymbol{\vartheta}_p]$ with $\boldsymbol{\vartheta}_p = [-\vartheta_p^-, \vartheta_p^+]$ at t_1 implies that $1 - \vartheta_p^- \le 1/v_p(t_1) \le 1 + \vartheta_p^+$, and the stability condition of Definition 2 ensures that $1 - \sigma_p(t_2 - t_1) \le v_p(t_2)/v_p(t_1) \le 1 + \sigma_p(t_2 - t_1)$ in the absence of resynchronizations until t_2. Combining these inequalities provides

$$\frac{1 - \vartheta_p^-}{1 + \sigma_p(t_2 - t_1)} \le \frac{1}{v_p(t_2)} \le \frac{1 + \vartheta_p^+}{1 - \sigma_p(t_2 - t_1)}. \tag{10}$$

Since $\sigma_p(t_2 - t_1) \ll 1$ we can approximate expression $1/(1 \pm \sigma_p(t_2 - t_1))$ by $1 \mp \sigma_p(t_2 - t_1)$, and further by $1 \mp \sigma_p(T_2 - T_1)$ due to (8) in conjunction with the fact that $\vartheta_p^\pm \ll 1$. Applying these approximations on (10) leads to $1/v_p(t_2) \in [1 \pm (\boldsymbol{\vartheta}_p + (T_2 - T_1)\boldsymbol{\sigma}_p)]$ which proves the desired inclusion. □

In order to demonstrate the deterioration of both accuracy and rate intervals, we simulate a free running clock with our SimUTC toolkit. The clock itself has an average drift of -0.5 ppm and its stability does not exceed 0.003 ppm/s. The left plot of Figure 2 draws $\boldsymbol{A}_p(t) - t$ against simulation time t, in other words, the interval of accuracies $\boldsymbol{\alpha}_p(t) = [-\alpha_p^-(t), \alpha_p^+(t)]$ from $\boldsymbol{A}_p(t) = [C_p(t) \pm \boldsymbol{\alpha}_p(t)]$ is tied to the traditional accuracy $C_p(t) - t$ as reference point. A diamond in the plot symbolizes the latter and verifies the characteristics of the clock. Note that $\boldsymbol{A}_p(t)$ is accurate at t if $0 \in \boldsymbol{A}_p(t) - t$ in this plot.

The deterioration of $\boldsymbol{\alpha}_p(t)$ is governed by the local rate interval $\boldsymbol{R}_p(t) = [1 \pm \boldsymbol{\vartheta}_p(t)]$ drawn in the right plot of Figure 2. It starts around 1.7 ppm and grows to 3.6 ppm, which explains the slight quadratic increase of $\alpha_p^\pm(t)$. The compensation for the clock's average drift is more than sufficient, just as for the clock's stability by the 0.02 ppm/s deterioration of $\boldsymbol{\vartheta}_p(t)$.

In our approach, accuracy intervals are deteriorated continuously by a novel hardware clock [17], whereas rate intervals are precomputed in such a way that their correctness holds until the next update.

3.3. Precision and Consonance Intervals

So far we have only considered clock states&rates with respect to real-time, which is pertinent to the external synchronization problem. To address the internal one, expressed by precision $\pi(t)$ and consonance $\gamma(t)$ as motivated in Section 2, we aim to reuse the same principles. For that purpose we construct a so-called *internal global time* $\tau(t)$ that follows the "center" of the ensemble of clocks. A characterization of $\tau(t)$ is defered to Section 4.1, however, without knowing it exactly we can already introduce *precision intervals* $\boldsymbol{\pi}_p(t)$ and *consonance intervals* $\boldsymbol{\gamma}_p(t)$ analogous to accuracy and rate intervals.

Definition 5 (precision interval) *Let $C_p(t)$ be the clock at node p and $T_0 = C_p(t_0)$ its state at real-time t_0. The associated precision interval $\hat{\boldsymbol{A}}_p = [T_0 \pm \boldsymbol{\pi}_p]$ with $\boldsymbol{\pi}_p = [-\pi_p^-, \pi_p^+]$ is $\boldsymbol{\pi}_p$-*accurate *at t_0 w.r.t. the internal global time $\tau(t)$ if*

$$\tau(t_0) \in \hat{\boldsymbol{A}}_p. \tag{11}$$

Precision intervals turn out to be an elegant analysis tool for internal clock state synchronization [12] and provide the useful precision estimation $\pi(t) \le \max_p\{\pi_p^-(t)\} + \max_p\{\pi_p^+(t)\}$, which is independent of the value $\tau(t)$. To compensate for the adverse clock drifts, how should we deteriorate the precision intervals? By reconsidering accuracy intervals, we notice that the difference between clock rate $v_p(t)$ and ideal rate 1 of real-time t is responsible for the enlargement of $\boldsymbol{\alpha}_p(t)$, which is given by rate interval $\boldsymbol{R}_p(t)$. Because precision interval $\boldsymbol{\pi}_p(t)$ goes along with internal global time $\tau(t)$, we need to capture its rate $\varphi(t) = d\tau(t)/dt$ relative to the rate $v_p(t)$ of the clock.

Definition 6 (consonance interval) *Let $C_p(t)$ be the clock at node p and $v_p(t) = dC_p(t)/dt$ its rate at real-time t. The associated consonance interval $\hat{\boldsymbol{R}}_p = [R_p \pm \boldsymbol{\gamma}_p]$ with $\boldsymbol{\gamma}_p = [-\gamma_p^-, \gamma_p^+]$ is $\boldsymbol{\gamma}_p$-*correct *during real-time interval $[t_1, t_2]$ w.r.t. rate $\varphi(t) = d\tau(t)/dt$ of internal global time $\tau(t)$ if*

$$\varphi(t)/v_p(t) \in \hat{\boldsymbol{R}}_p \qquad \forall\, t_1 \le t \le t_2. \tag{12}$$

In view of the lesson we have learned in Section 2, it is intuitively clear that a precision interval $\boldsymbol{\pi}_p(t)$ calls for a deterioration with its consonance interval $\boldsymbol{\gamma}_p(t)$.

Lemma 3 (consonance compensation of precision intervals) *Let $C_p(t)$ be the clock at node p and $T_1 = C_p(t_1)$ resp. $T_2 = C_p(t_2)$ its state at real-time t_1 resp. t_2 with $t_1 \le t_2$. If associated precision interval $\hat{\boldsymbol{A}}_p = [T_0 \pm \boldsymbol{\pi}_p]$ is $\boldsymbol{\pi}_p$-accurate at t_1 w.r.t. the internal global time $\tau(t)$, and associated local consonance interval $\hat{\boldsymbol{R}}_p = [1 \pm \boldsymbol{\gamma}_p]$ is $\boldsymbol{\gamma}_p$-correct during $[t_1, t_2]$ w.r.t. a constant rate of internal global time $\tau(t)$ during $[t_1, t_2]$, then $\hat{\boldsymbol{A}}'_p = [T_2 \pm \boldsymbol{\pi}'_p]$ with*

$$\boldsymbol{\pi}'_p = \boldsymbol{\pi}_p + (T_2 - T_1)\boldsymbol{\gamma}_p \tag{13}$$

is $\boldsymbol{\pi}'_p$-accurate at t_2 w.r.t. the constant rate of $\tau(t)$, when no resynchronization occurred during $[t_1, t_2]$.

Proof The given $\boldsymbol{\pi}_p$-accurateness of $\hat{\boldsymbol{A}}_p$ at t_1 means that $\tau(t_1) \in [T_1 \pm \boldsymbol{\pi}_p]$. Since there are no resynchronizations until t_2, we know that

$$\tau(t_2) \in [T_1 \pm \boldsymbol{\pi}_p] + \varphi(t_2 - t_1) \tag{14}$$

if φ denotes the constant rate of $\tau(t)$ during $[t_1, t_2]$. According to Definition 6, the $\boldsymbol{\gamma}_p$-correctness of $\hat{\boldsymbol{R}}_p$ during $[t_1, t_2]$ ensures that $v_p(t)(1 - \gamma_p^-) \le \varphi \le v_p(t)(1 + \gamma_p^+)$ for any $t \in [t_1, t_2]$. Integrating this inequality from t_1 to t_2 yields $\varphi(t_2 - t_1) \in (T_2 - T_1)\left[1 \pm [-\gamma_p^-, \gamma_p^+]\right]$ by reusing parts from the proof of Lemma 1. Plugging the latter into (14) provides $\tau(t_2) \in \left[T_2 \pm (\boldsymbol{\pi}_p + (T_2 - T_1)\boldsymbol{\gamma}_p)\right]$, where the reference point equals the one in (7), and the deteriorated precision interval comes out as claimed in the lemma. □

It remains to examine the deterioration of consonance intervals, since they play an important role in the analysis of internal clock rate synchronization [15] and provide the consonance estimation $\gamma(t) \le \max_p\{\gamma_p^-(t)\} + \max_p\{\gamma_p^+(t)\}$, which is independent of the value $\varphi(t)$. When the latter is assumed to be constant, we are dealing with the same situation as for rate intervals.

Lemma 4 (stability compensation of consonance intervals) *Let $C_p(t)$ be the clock at node p and $T_1 = C_p(t_1)$ resp. $T_2 = C_p(t_2)$ its state at real-time t_1 resp. t_2 with $t_1 \le t_2$. If associated consonance interval $\hat{\boldsymbol{R}}_p = [1 \pm \boldsymbol{\gamma}_p]$ with $\boldsymbol{\gamma}_p = [-\gamma_p^-, \gamma_p^+]$ is $\boldsymbol{\gamma}_p$-correct at t_1 w.r.t. a constant rate of internal global time $\tau(t)$ during $[t_1, t_2]$, and $\boldsymbol{\sigma}_p$ is the stability interval of the oscillator, then $\hat{\boldsymbol{R}}'_p = [1 \pm \boldsymbol{\gamma}'_p]$ with*

$$\boldsymbol{\gamma}'_p \doteq \boldsymbol{\gamma}_p + [0 \pm (T_2 - T_1)\boldsymbol{\sigma}_p] \tag{15}$$

is $\boldsymbol{\gamma}'_p$-correct at t_2 w.r.t. the constant rate of $\tau(t)$, when no resynchronization occurred during $[t_1, t_2]$.

Proof The same line of reasoning as in the proof of Lemma 2 can be applied, where the assumed constant rate $\varphi = d\tau(t)/dt$ of internal global time replaces the ideal rate 1 of real-time. □

3.4. Interval Transmission

In the previous sections all intervals are maintained locally, but how can we pass them on to other nodes without loosing their meaning? Before looking at each of the intervals, we make the assumption that nodes communicate via a packet-based communication subsystem which can either be a fully connected *point-to-point* network or a *broadcast-type* network with synchronous behavior.

Definition 7 (packet transmission) *A packet transmitted from node q to node p experiences a delay $\delta'_{p,q}$ subject to the* delay condition

$$\delta'_{p,q} \in \boldsymbol{\delta}_{p,q} = [\delta_{p,q} \pm \boldsymbol{\epsilon}_{p,q}], \tag{16}$$

where $\delta_{p,q} \geq 0$ represents the average transmission delay and $\boldsymbol{\epsilon}_{p,q} = [-\epsilon^-_{p,q}, \epsilon^+_{p,q}]$ the interval of delay uncertainties.

For convenience we define $\epsilon^-_{\max} = \max_{p,q}\{\epsilon^-_{p,q}\}$ and $\epsilon^+_{\max} = \max_{p,q}\{\epsilon^+_{p,q}\}$ as uniform bounds, as well as $\epsilon_{\max} = \epsilon^-_{\max} + \epsilon^+_{\max}$. Our synchronization algorithm depends on a good setting of these transmission delay parameters, since an underestimation can lead to faulty intervals and an overestimation to a performance loss. Values for them have to be measured with (round-trip) protocols prior to having the system turned on or during an initialization phase, and need to be checked and if necessary updated on-line. For a LAN with a shared broadcast channel $\epsilon^{\pm}_{\max}$ is typically in the ms-range by using software timestamping, but the μs-range is reachable with our hardware timestamping approach [11].

In case of accuracy and precision intervals, their exchange among nodes is easy, since both of them capture only global quantities, i.e., real-time t and internal global time $\tau(t)$. When such an interval arrives at a node, we only need to compensate for the experienced transmission delay to maintain its property.

Lemma 5 (delay compensation of accuracy and precision intervals) *Let node q transmit a packet with accuracy interval $\boldsymbol{A}_q$ at real-time t_q to node p that receives it at real-time t_p. If $\boldsymbol{A}_q$ is accurate at t_q and associated precision interval $\hat{\boldsymbol{A}}_q$ is $\boldsymbol{\pi}_q$-accurate at t_q w.r.t. the internal global time $\tau(t)$, then* remote accuracy interval

$$\boldsymbol{A}_{p,q} = \boldsymbol{A}_q + \boldsymbol{\delta}_{p,q} \tag{17}$$

is accurate at t_p, and $\hat{\boldsymbol{A}}_{p,q}$, which has the same reference point as $\boldsymbol{A}_{p,q}$, is $\boldsymbol{\pi}_{p,q}$-accurate at t_p w.r.t. $\tau(t)$ with remote precision interval

$$\boldsymbol{\pi}_{p,q} \doteq \boldsymbol{\pi}_q + \boldsymbol{\epsilon}_{p,q}. \tag{18}$$

Proof Since $\boldsymbol{A}_q$ is accurate at t_q, we know that $t_q + \delta'_{p,q} \in \boldsymbol{A}_q + \delta'_{p,q}$ for any delay $\delta'_{p,q}$. The left-hand side is simply the reception time t_p, and by applying (16) on the right-hand side we end up with (17). Similarly, for the associated precision interval $\hat{\boldsymbol{A}}_q$ we know that $\tau(t_q) + \delta'_{p,q} \in \hat{\boldsymbol{A}}_q + \delta'_{p,q}$ for any delay $\delta'_{p,q}$. The left-hand side is approximated with $\tau(t_p)$ since $|d\tau(t)/dt - 1| \ll 1$ and by applying (16) again on the right-hand side we have (18). □

In case of rate and consonance intervals a compensation for the transmission turns out to be more difficult, since the node dependent clock rate $v_p(t)$ is also involved, see Definition 4 and 6. For that purpose we need to introduce a so-called *quotient rate interval* that mediates between the clock rates.

Definition 8 (quotient rate interval) *Let $C_p(t)$ and $C_q(t)$ be the clocks at different nodes p and q with their respective rates $v_p(t)$ and $v_q(t)$. A quotient rate interval $\boldsymbol{Q}_{p,q} = [Q_{p,q} \pm \boldsymbol{\nu}_{p,q}]$ with $\boldsymbol{\nu}_{p,q} = [-\nu^-_{p,q}, \nu^+_{p,q}]$ is* valid *during real-time interval $[t_1, t_2]$ if*

$$v_q(t_q)/v_p(t_p) \in \boldsymbol{Q}_{p,q} \qquad \forall\, t_1 \leq t_p, t_q \leq t_2. \tag{19}$$

In Section 3.5 we propose a simple protocol to obtain a quotient rate interval that captures the relative rates between a pair of clocks. These intervals enable us to carry over received rate/consonance intervals by using interval multiplication, which is defined as $[I_1 \pm \boldsymbol{\iota}_1]\cdot[I_2 \pm \boldsymbol{\iota}_2] = [(I_1 - \iota^-_1)(I_2 - \iota^-_2), (I_1 + \iota^+_1)(I_2 + \iota^+_2)]$ and by setting the reference point to $I_1 I_2$

Lemma 6 (delay compensation of rate and consonance intervals) *Let node q transmit a packet with local rate interval $\boldsymbol{R}_q$ at real-time t_q to node p that receives it at real-time t_p, and let $\boldsymbol{Q}_{p,q}$ be a quotient rate interval valid during $[t_q, t_p]$. If $\boldsymbol{R}_q$ is correct at t_q and associated local consonance interval $\hat{\boldsymbol{R}}_q$ is $\boldsymbol{\gamma}_q$-correct at t_q w.r.t. a constant rate of internal global time $\tau(t)$ during $[t_q, t_p]$, then* remote rate interval

$$\boldsymbol{R}_{p,q} = \boldsymbol{Q}_{p,q}\cdot\boldsymbol{R}_q \tag{20}$$

is correct at t_p, and $\hat{\boldsymbol{R}}_{p,q} = \boldsymbol{Q}_{p,q}\cdot\hat{\boldsymbol{R}}_q$ is $\boldsymbol{\gamma}_{p,q}$-correct at t_p w.r.t. $\tau(t)$ with remote consonance interval

$$\boldsymbol{\gamma}_{p,q} = \boldsymbol{Q}_{p,q}\cdot\boldsymbol{\gamma}_q. \tag{21}$$

Proof The correctness of local rate interval $\boldsymbol{R}_q = [1 \pm \boldsymbol{\vartheta}_q]$ with $\boldsymbol{\vartheta}_q = [-\vartheta^-_q, \vartheta^+_q]$ at t_q asserts that $1 - \vartheta^-_q \leq 1/v_q(t_q) \leq 1 + \vartheta^+_q$, and the quotient rate interval $\boldsymbol{Q}_{p,q} = [Q_{p,q} \pm \boldsymbol{\nu}_{p,q}]$ with $\boldsymbol{\nu}_{p,q} = [-\nu_{p,q}, \nu_{p,q}]$ ensures that $Q_{p,q} - \nu_{p,q} \leq v_q(t_q)/v_p(t_p) \leq Q_{p,q} + \nu_{p,q}$. Combining these inequalities provides

$$(Q_{p,q} - \nu_{p,q})(1 - \vartheta^-_q) \leq \frac{1}{v_p(t_p)} \leq (Q_{p,q} + \nu_{p,q})(1 + \vartheta^+_q),$$

and making use of the above defined interval multiplication we have $1/v_p(t_p) \in [Q_{p,q} \pm \boldsymbol{\nu}_{p,q}] \cdot [1 \pm \boldsymbol{\vartheta}_q]$ which confirms (20) of our lemma. Similarly, for the associated local consonance interval $\hat{\boldsymbol{R}}_q = [1 \pm \boldsymbol{\gamma}_q]$ with $\boldsymbol{\gamma}_q = [-\gamma^-_q, \gamma^+_q]$ it follows that

$$(Q_{p,q} - \nu_{p,q})(1 - \gamma^-_q) \leq \frac{\varphi(t_p)}{v_p(t_p)} \leq (Q_{p,q} + \nu_{p,q})(1 + \gamma^+_q),$$

which leads to (21) when setting $\varphi(t_p) = \varphi(t_q)$ since rate $\varphi(t)$ of internal global time is constant during $[t_q, t_p]$. □

Figure 3 summarizes the relations between the intervals for clock state&rate synchronization. The left side shows local intervals at node q and the right side remote intervals at node p. The circled arrows indicate deterioration operations and the horizontal arrows delay compensation operations.

$$
\begin{array}{llllll}
\text{state:} & \boldsymbol{A}_q & \boldsymbol{\pi}_q & \xrightarrow{\quad\boldsymbol{\delta}_{p,q}\quad} & \boldsymbol{A}_{p,q} & \boldsymbol{\pi}_{p,q} \\
 & \circlearrowright & \circlearrowright & & & \\
\text{rate:} & \boldsymbol{R}_q & \gamma_q & \xrightarrow{\quad\boldsymbol{Q}_{p,q}\quad} & \boldsymbol{R}_{p,q} & \gamma_{p,q} \\
 & \circlearrowright & \circlearrowright & & & \\
\text{stability:} & \boldsymbol{\sigma}_q & \boldsymbol{\sigma}_q & & &
\end{array}
$$

Figure 3. Clock State&Rate Intervals.

3.5. Relative Rate Measurements

It only remains to determine the quotient rate interval $\boldsymbol{Q}_{p,q}$ from Definition 8 that quantifies how fast/slow a remote clock $C_q(t)$ is with respect to a local clock $C_p(t)$, hence the gap in the delay compensation of rate and consonance intervals in Lemma 6 can be filled.

Informally, the protocol for this *relative rate measurement* is based on repeated packet pairs, such that remote node q broadcasts periodically a packet that contains its current clock state $T_q = C_q(t_q)$. Local node p records its current clock state $T_{p,q} = C_p(t_{p,q})$ upon packet arrival. Therefore, node p can obtain clock states T_q, $T_{p,q}$ from a packet arrival as well as $T_q^{\prec}$, $T_{p,q}^{\prec}$ from a previous one. With these four clock states and taking into account the transmission delay characteristics from Definition 7 and the clock characteristics from Definition 2, node p can compute the desired quotient rate interval $\boldsymbol{Q}_{p,q}$.

Lemma 7 (relative rate measurement) *Let $C_p(t)$ be a clock at local node p driven by an oscillator with stability interval $\boldsymbol{\sigma}_p$, and $C_q(t)$ a clock at remote node q driven by an oscillator with stability interval $\boldsymbol{\sigma}_q$. When executing the above sketched protocol in order to obtain the clock states $T_q^{\prec} = C_q(t_q^{\prec})$, $T_{p,q}^{\prec} = C_p(t_{p,q}^{\prec})$, $T_q = C_q(t_q)$, and $T_{p,q} = C_p(t_{p,q})$, then quotient rate interval*

$$\boldsymbol{Q}_{p,q} \doteq \left[\frac{T_q - T_q^{\prec}}{T_{p,q} - T_{p,q}^{\prec}} \pm \boldsymbol{\nu}_{p,q}\right] \tag{22}$$

with $\boldsymbol{\nu}_{p,q} = [0 \pm ((T_q - T_q^{\prec})(\boldsymbol{\sigma}_p + \boldsymbol{\sigma}_q)/2 + (\boldsymbol{\epsilon}_{p,q} + \bar{\boldsymbol{\epsilon}}_{p,q})/(T_{p,q} - T_{p,q}^{\prec}))]$ is valid during $[t_q^{\prec}, t_{p,q}]$ when no resynchronization occurred.

Proof (sketch) Considering the real-time duration $t_q - t_q^{\prec}$ at sending node q and $t_{p,q} - t_{p,q}^{\prec}$ at receiving node p, we get $(t_q - t_q^{\prec}) - (t_{p,q} - t_{p,q}^{\prec}) \in \boldsymbol{\epsilon}_{p,q} + \bar{\boldsymbol{\epsilon}}_{p,q}$ by Definition 7. For the clock rate $v_q(t)$ at node q during $[t_q^{\prec}, t_q]$ we derive $v_q(t) \in \left[(T_q - T_q^{\prec})/(t_q - t_q^{\prec}) \pm (T_q - T_q^{\prec})\boldsymbol{\sigma}_q/2\right]$ and a similar expression for $v_p(t)$ during $[t_{p,q}^{\prec}, t_{p,q}]$ by exploiting Definition 2 in conjunction with approximation $1/(1 \pm \sigma_q(t_q - t_q^{\prec})) \doteq 1 \mp \sigma_q(T_q - T_q^{\prec})$. Putting all parts together in order to find bounds on the ratio $v_q(t)/v_p(t)$ leads after several algebraic manipulations to the quotient rate interval (22). □

4. Algorithm

In this section we put together our algorithm for combined clock state&rate synchronization based on the concepts from Section 3. State synchronizations take place in rounds with *state period* P_S, and rate synchronizations with *rate period* P_R. We set $P_R = mP_S$ with non-negative integer m, since useful relative rate measurements take much longer than a state period.

Definition 9 (state&rate algorithm) *Periodically, each node $p \in \{1, \ldots, n\}$ performs the following operations:*

- **Send:** *At $k_S P_S$ calculate round number $\bar{k}_S = k_S \bmod m$. Initiate broadcast of packets $\langle \boldsymbol{A}_p, \boldsymbol{R}_p, U_p \rangle$ if $\bar{k}_S = 0$, or $\langle \boldsymbol{A}_p \rangle$ otherwise.*
- **Reception and Preprocessing:** *Until $k_S P_S + \Delta$ timestamp the arriving packets $\langle \boldsymbol{A}_q \rangle$ or $\langle \boldsymbol{A}_q, \boldsymbol{R}_q, U_q \rangle$ from nodes q with $T_{p,q}$. Then compute the remote accuracy interval*

  $$\boldsymbol{A}_{p,q} = \boldsymbol{A}_q + \boldsymbol{\delta}_{p,q} + (k_S P_S + \Delta + E_p - T_{p,q})\boldsymbol{R}_p$$

  if $p \neq q$, or $\boldsymbol{A}_{p,p} = \boldsymbol{A}_p + (k_S P_S + \Delta + E_p - T_{p,p})\boldsymbol{R}_p$ otherwise, and store it in ordered set $\boldsymbol{\mathcal{A}}_p$. If $\bar{k}_S = 0$ then compute the quotient rate interval

  $$\begin{aligned}\boldsymbol{Q}_{p,q} \;=\; \big[&(T_q - T_q^{\prec} + U_q)/(T_{p,q} - T_{p,q}^{\prec} + U_p) \\ &+ [0 \pm ((T_q - T_q^{\prec} + U_q)(\boldsymbol{\sigma}_p + \boldsymbol{\sigma}_q)/2 \\ &\quad + (\boldsymbol{\epsilon}_{p,q} + \bar{\boldsymbol{\epsilon}}_{p,q})/(T_{p,q} - T_{p,q}^{\prec} + U_p))]\big]\end{aligned}$$

  if $p \neq q$, assign $T_q^{\prec} = T_q$ and $T_{p,q}^{\prec} = T_{p,q}$ for the next rate round, compute the remote rate interval

  $$\boldsymbol{R}_{p,q} = \boldsymbol{Q}_{p,q} \cdot \boldsymbol{R}_q + [0 \pm (\Delta + E_q)(\boldsymbol{\sigma}_p + \boldsymbol{\sigma}_q)]$$

  if $p \neq q$, or $\boldsymbol{R}_{p,p} = \boldsymbol{R}_p$ otherwise, and store it in ordered set $\boldsymbol{\mathcal{R}}_p$.
- **Computation:** *At $k_S P_S + \Delta$ compute the new accuracy interval*

  $$\boldsymbol{A}'_p = [A'_p \pm \boldsymbol{\alpha}'_p] = \boldsymbol{\mathcal{OP}}_{n-f}^{\boldsymbol{\pi}^{\circ}, \boldsymbol{\pi}^{H}, \Upsilon_{\max}}(\boldsymbol{\mathcal{A}}_p).$$

  If $\bar{k}_S = 0$ then compute the new rate interval

  $$\boldsymbol{R}'_p = [R'_p \pm \boldsymbol{\vartheta}'_p] = \boldsymbol{\mathcal{OP}}_{n-f}^{\boldsymbol{\gamma}^{\circ}, \boldsymbol{\gamma}^{H}, \Theta_{\max}}(\boldsymbol{\mathcal{R}}_p).$$

- **Adjustment:** *At $k_S P_S + \Delta + E_p$ set the clock to $\boldsymbol{A}'_p$ and deteriorate it with the local rate interval*

  $$\boldsymbol{R}_p = \boldsymbol{R}'_p / R'_p + \left[0 \pm (\bar{k}_S + 1) P_S \boldsymbol{\sigma}_p\right].$$

  If $\bar{k}_S = 0$ then adjust the clock rate by setting the coupling factor $S'_p = S_p R'_p$ and $U_p = \Upsilon'_p$ else $U_p = U_p + \Upsilon'_p$, where state adjustment $\Upsilon'_p = A'_p - (k_S P_S + \Delta + E_p)$. Finally, increment state round k_S.

Let us explain the above algorithm by examining a particular *state round* k_S, which is the $(\bar{k}_S+1)$-th round within the enclosing *rate round*. At $k_S P_S$ each node p initiates a broadcast of packet(s), containing the current accuracy interval $\boldsymbol{A}_p$ and, when a rate synchronization is also ahead as indicated by $\bar{k}_S = 0$, the local rate interval $\boldsymbol{R}_p$ as well as the sum U_p of applied state adjustments. The latter is necessary to compensate for interspersed state adjustments during relative rate measurements, otherwise the formula from Lemma 7 cannot be applied.

When node p receives such packets from another node q, it timestamps them with $T_{p,q}$ and computes the remote accuracy intervals $\boldsymbol{A}_{p,q}$ according to Lemma 5 and 1. Note that for $\boldsymbol{A}_{p,p}$ no delay compensation is necessary. Parameter $\Delta \geq 0$ should be large enough to ensure a full exchange of packets among the nodes, and also parameter $E_p \geq 0$ to finish all computations before the state synchronization launches the next round k_S+1 at $k_S P_S + \Delta + E_p$. In addition, when a rate synchronization is indicated, node p computes the quotient rate intervals $\boldsymbol{Q}_{p,q}$ and subsequently the remote rate intervals $\boldsymbol{R}_{p,q}$ by using Lemma 7, refined with the sum of state adjustments, and Lemma 6, respectively. Note that $\boldsymbol{Q}_{p,p}$ is not explicitly required, since $\boldsymbol{R}_{p,p} = \boldsymbol{R}_p$. All these intervals are stored in an appropriate ordered set.

The accuracy intervals in $\boldsymbol{\mathcal{A}}_p$ are supposed to contain the approaching resynchronization time t'_p, and the rate intervals in $\boldsymbol{\mathcal{R}}_p$ clock rate $v_p(t'_p)$. It is the purpose of a suitable interval-based *convergence function* to compute a new accuracy interval $\boldsymbol{A}'_p$ out of $\boldsymbol{\mathcal{A}}_p$ resp. a new rate interval $\boldsymbol{R}'_p$ out of $\boldsymbol{\mathcal{R}}_p$ that encloses t'_p resp. $v_p(t'_p)$ more closely. Moreover, in order to achieve internal clock synchronization, the new associated precision/consonance intervals are also required to be small. Sections 4.1 and 4.2 are devoted to this intricate problem and explain our convergence function $\mathcal{OP}(\cdot)$.

Eventually, at $k_S P_S + \Delta + E_p$ the state adjustment is applied either instantaneously or by continuous amortization [12]. The additive amount Υ'_p is given by the difference between the reference point A'_p of the computed accuracy interval $\boldsymbol{A}'_p$ and the current clock state $k_S P_S + \Delta + E_p$. The less frequent rate adjustment is carried out by tuning coupling factor S_p, see Section 3.1. The amount here is given in a multiplicative way, since the reference point R'_p of the computed rate interval $\boldsymbol{R}'_p$ needs to be understood as the ratio between the desired new clock rate and the current (unknown) one. This leads to the new coupling factor $S'_p = S_p R'_p$ by virtue of (2), and by enforcing this rate adjustment the new local rate interval becomes $\boldsymbol{R}'_p / R'_p = [1 \pm \vartheta'_p / R'_p]$. To keep the latter correct for the next m state rounds, a deterioration by $P_R \boldsymbol{\sigma}_p$ is apparent, but doing it gradually by $(\bar{k}_S+1) P_S \boldsymbol{\sigma}_p$ at each state synchronization is more efficient. Last but not least, the algorithm updates the running sum U_p of applied state adjustments during a relative rate measurement period.

4.1. Convergence Function

Although different convergence functions could be used for clock state and rate synchronization, we decided to take the *optimal precision function* $\mathcal{OP}(\cdot)$ for both cases. This interval-based function has been developed in [10] and can be regarded as an extension of Marzullo's function $\mathcal{M}(\cdot)$ defined in [8].

Definition 10 (Marzullo function) *Given a set* $\boldsymbol{\mathcal{I}} = \{\boldsymbol{I}_1, \ldots, \boldsymbol{I}_n\}$ *of* $n \geq 1$ *non-empty intervals* $\boldsymbol{I}_j$ *with at least* $n-f \geq 1$ *of the intervals being non-faulty,* $\mathcal{M}_n^{n-f}(\boldsymbol{\mathcal{I}})$ *is defined as the largest interval* $[I^-, I^+]$ *whose edges* $I^\pm$ *lie in the intersection of at least* $n-f$ *different* $\boldsymbol{I}_j$*'s.*

In the context of state synchronization we explain the construction of function $\mathcal{OP}(\cdot)$, which irons out two deficiencies of function $\mathcal{M}(\cdot)$. The first improvement utilizes the fact that *accuracy adjustments* beyond the locally maintained accuracy interval are unnecessary, thus it is reasonable to set forth the intersection $\mathcal{M}_n^{n-f}(\boldsymbol{\mathcal{A}}_p) \cap \boldsymbol{A}_{p,p}$ at node p. The disadvantage of this modification is that a faulty interval $\boldsymbol{A}_{p,p}$ might hamper a recovery process.

The other improvement deals with *precision adjustments*, since the reference point of $\mathcal{M}(\cdot)$ is implicitly set to the midpoint. As discovered by [4] and revisited by [12], this simple approach fails to guarantee a small precision when the accuracy intervals in $\boldsymbol{\mathcal{A}}_p$ become too large. We solved that problem by inventing associated precision intervals $\hat{\boldsymbol{A}}_p$, which turn out to have a bounded length when following internal global time $\tau(t)$. However, the characterization in Definition 5 requires a certain precision interval to express the adhered accurateness. The analysis in [15], which is geared towards the algorithm from Definition 9, provides us with $\boldsymbol{\pi}^o$ for the local associated precision interval $\hat{\boldsymbol{A}}_{p,p}$ and $\boldsymbol{\pi}^H$ for remote ones $\hat{\boldsymbol{A}}_{p,q}$, thus

$$\hat{\boldsymbol{A}}_{p,q} = \begin{cases} [A_{p,q} \pm \boldsymbol{\pi}^H] & \text{if } p \neq q \\ [A_{p,p} \pm \boldsymbol{\pi}^o] & \text{otherwise.} \end{cases} \tag{23}$$

Note that $\boldsymbol{\pi}^o \subseteq \boldsymbol{\pi}^H$ because the local one is not spoiled by transmission delay uncertainties. Hence we are able to compute the set $\hat{\boldsymbol{\mathcal{A}}}_p$ of associated precision intervals at node p and after that the intersection $\mathcal{M}_n^{n-f}(\hat{\boldsymbol{\mathcal{A}}}_p) \cap \hat{\boldsymbol{A}}_{p,p}$ by using the same argument as above. Instead of setting the reference point to the midpoint, explorations in [10] suggest to weight the left edge with π^{o+} and the right edge with π^{o-}, drawn from the precision interval $\boldsymbol{\pi}^o = [-\pi^{o-}, \pi^{o+}]$. Moreover, the work of [3] revealed that state adjustments should be limited to a certain $\Upsilon_{\max}$, whose characterization can be found in [15] as well. In case that the resulting reference point lies outside the accuracy interval, we need to expand the latter accordingly.

Definition 11 (optimal precision function) *Given a set $\boldsymbol{\mathcal{A}}_p = \{\boldsymbol{A}_{p,1}, \ldots, \boldsymbol{A}_{p,n}\}$ of $n \geq 1$ non-empty remote accuracy intervals $\boldsymbol{A}_{p,q} = [A_{p,q} \pm \boldsymbol{\alpha}_{p,q}]$. For given precision intervals $\boldsymbol{\pi}^o$ and $\boldsymbol{\pi}^H$, fault-tolerance parameter f, and maximum state adjustment $\Upsilon_{\max}$, the result $\boldsymbol{A}'_p = [A'_p \pm \boldsymbol{\alpha}'_p]$ of the* optimal precision convergence function $\boldsymbol{\mathcal{OP}}^{\boldsymbol{\pi}^o,\boldsymbol{\pi}^H,\Upsilon_{\max}}_{n-f}(\cdot)$ *applied on $\boldsymbol{\mathcal{A}}_p$ is defined as follows:*

- *The reference point*

$$A'_p = \begin{cases} A_{p,p} + \Upsilon_{\max} & \text{if } A'_{p,p} - A_{p,p} > \Upsilon_{\max} \\ A_{p,p} - \Upsilon_{\max} & \text{if } A_{p,p} - A'_{p,p} > \Upsilon_{\max} \\ A'_{p,p} & \text{otherwise}, \end{cases}$$

where $A'_{p,p} = (\Pi^+_{p,p}\pi^{o-} + \Pi^-_{p,p}\pi^{o+})/(\pi^{o-} + \pi^{o+})$ is derived from the edges $\Pi^{\pm}_{p,p}$ of the associated precision interval

$$[\Pi^-_{p,p}, \Pi^+_{p,p}] = \boldsymbol{\mathcal{M}}^{n-f}_n(\hat{\boldsymbol{\mathcal{A}}}_p) \cap \hat{\boldsymbol{A}}_{p,p}.$$

- *The interval of accuracies*

$$\boldsymbol{\alpha}'_p = \left[-\max\{0, A'_p - A^-_{p,p}\}, \max\{0, A^+_{p,p} - A'_p\}\right],$$

where $A^{\pm}_{p,p}$ are the edges of the accuracy interval

$$[A^-_{p,p}, A^+_{p,p}] = \boldsymbol{\mathcal{M}}^{n-f}_n(\boldsymbol{\mathcal{A}}_p) \cap \boldsymbol{A}_{p,p}.$$

In order to obtain *drift/consonance adjustments* for the purpose of rate synchronization we can employ function $\boldsymbol{\mathcal{OP}}(\cdot)$ in a similar way. Here we apply it on the set $\boldsymbol{\mathcal{R}}_p$ of remote rate intervals to compute the new one $\boldsymbol{R}'_p = [R'_p \pm \boldsymbol{\vartheta}'_p]$, see Definition 9. Parameter $\boldsymbol{\gamma}^o$ resp. $\boldsymbol{\gamma}^H$ is used to compute the local resp. remote associated consonance intervals. A limited rate adjustment is imposed with the help of parameter $\Theta_{\max}$. The analysis in [15] provides a characterization for all three parameters.

Finally, let us briefly turn our attention to internal global time $\tau(t)$, which is the instrument to understand the analysis of internal state&rate synchronization. At any state resynchronization t_{k_S} we globally assign $\tau(t)$ a new value τ_{k_S}, which lies in the intersection of the new associated accuracy intervals $\hat{\boldsymbol{A}}'_p$. The analysis in [12] ensures the existence of such a value under the constraint that the latter intervals are of short length. In absence of rate synchronization, internal global time is being advanced just as real-time, thus $\tau(t) = \tau_{k_S} + (t - t_{k_S})$ up to the next state synchronization. However, our new (short) associated rate intervals $\hat{\boldsymbol{R}}'_p$, obtained at any rate resynchronization t_{k_R}, provide us with a new rate φ_{k_R} for internal global time. This leads to the refined definition of internal global time

$$\tau(t) = \tau_{k_S} + \varphi_{k_R}(t - t_{k_S}), \tag{24}$$

expressing a piecewise linear progression. Therefore the assumption about $\tau(t)$ in Lemma 3, 4, and 6 is sound, as long as interspersed state adjustments are taken into account.

4.2. Fault Model

Supposing the distributed system is free of any failures, plain intersection seems to be the best choice as convergence function for the computation of the new intervals $\boldsymbol{A}'_p$ and $\boldsymbol{R}'_p$. Of course, in case of a realistic system the synchronization algorithm needs to be furnished with a convergence function that is fault-tolerant with respect to an *(abstract) fault model*. We call the latter "abstract" because faults are not viewed in terms of faulty system components, but rather as faulty intervals in the input sets $\boldsymbol{\mathcal{A}}_p$ or $\boldsymbol{\mathcal{R}}_p$. Leaned on the work of [9] intervals can be faulty with regard to their reference point, their lengths, or missing altogether, leading to the following classification.

Definition 12 (faulty interval) *Let $\boldsymbol{\mathcal{I}}_p = \{\boldsymbol{I}_{p,1}, \ldots, \boldsymbol{I}_{p,n}\}$ and $\boldsymbol{\mathcal{I}}_q = \{\boldsymbol{I}_{q,1}, \ldots, \boldsymbol{I}_{q,n}\}$ be the ordered sets of intervals obtained at any two non-faulty nodes p and q after reception and preprocessing. A pair of intervals $\{\boldsymbol{I}_{p,s}, \boldsymbol{I}_{q,s}\}$ supposed to contain some ξ suffers from a*

- crash fault *if $\boldsymbol{I}_{p,s} = \boldsymbol{I}_{q,s} = \emptyset$,*
- symmetric fault *if either both $\boldsymbol{I}_{p,s}$ and $\boldsymbol{I}_{q,s}$ are faulty in the sense that ξ is outside left or right of both intervals, or without loss of generality, $\boldsymbol{I}_{p,s} = \emptyset$ and $\boldsymbol{I}_{q,s} \neq \emptyset$ is non-faulty but its length is within certain bounds (receive omission),*
- asymmetric fault *if either both $\boldsymbol{I}_{p,s}$ and $\boldsymbol{I}_{q,s}$ are faulty in the sense that ξ is outside left in one of these intervals and outside right in the other one (true Byzantine fault), or without loss of generality, $\boldsymbol{I}_{p,s} \neq \emptyset$ is faulty and $\boldsymbol{I}_{q,s} \neq \emptyset$ is arbitrary,*
- restricted fault *if it suffers from a symmetric fault or a crash fault.*
- arbitrary fault *if it suffers from an asymmetric or a restricted fault.*

Separating different types of faults and building a hierarchy as in Definition 12 yields a fault model similar to the one of [1], however, we consider pairwise perceptions $\boldsymbol{\mathcal{I}}_p$ and $\boldsymbol{\mathcal{I}}_q$ at two different receiving nodes p and q in isolation. Our fault model is thus of wider applicability than conventional ones, since instead of working with system-global fault assumptions we require only the maximum number of faulty intervals in pairwise perceptions. Beyond that we rule out the possibility of *impersonating* other nodes or *jamming* the network.

Assuming that at most f_A resp. f_R of the pairs of intervals $\{\boldsymbol{I}_{p,s}, \boldsymbol{I}_{q,s}\}$, for each $1 \leq s \leq n$, suffer from arbitrary resp. restricted faults, the analysis in [15] shows that $\boldsymbol{\mathcal{OP}}(\cdot)$ can tolerate faults of these types when

$$n \geq 3f_A + 2f_R + 1. \tag{25}$$

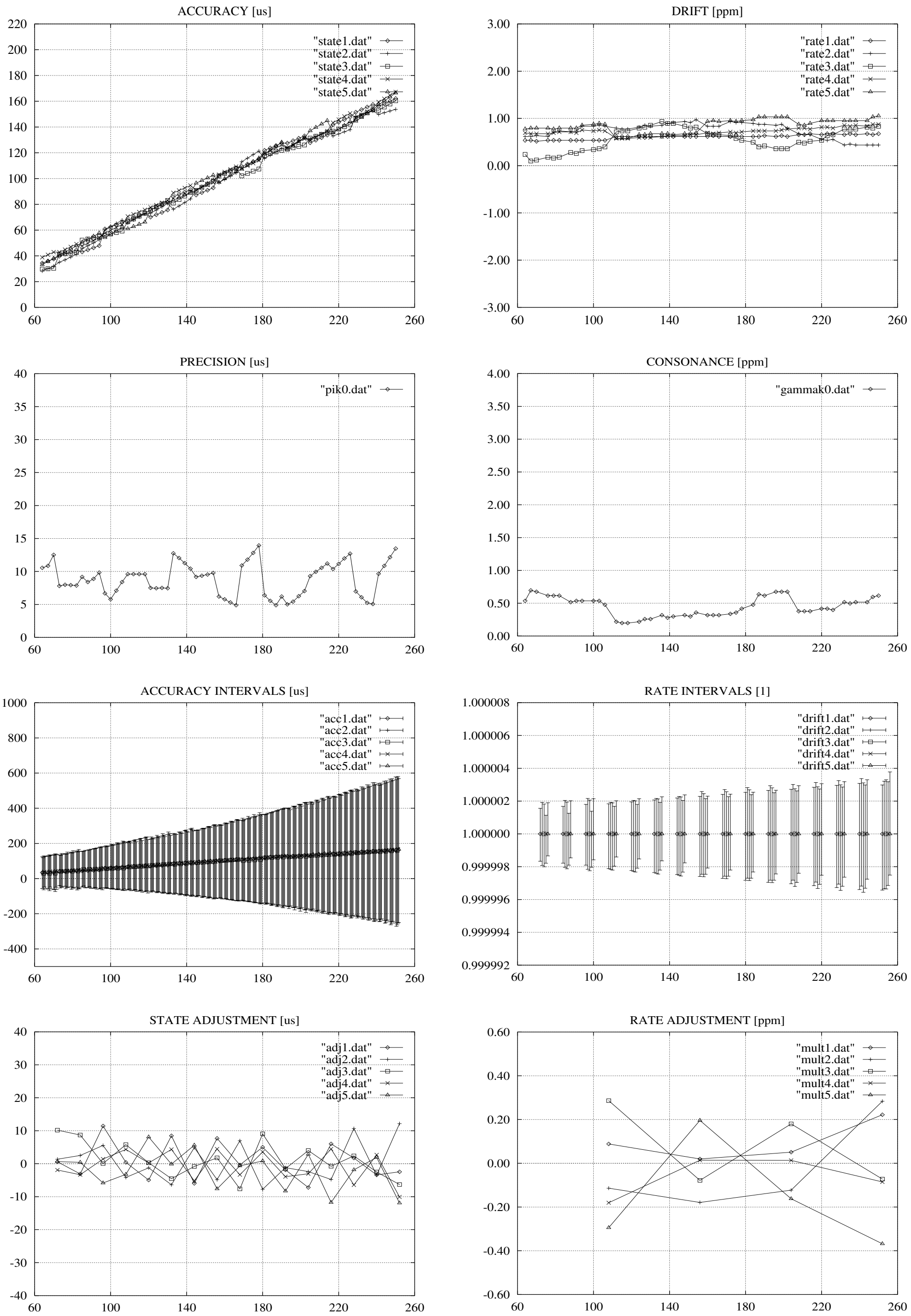

Figure 4. Clock State&Rate Synchronization.

In order to apply this result to our state&rate intervals some refinements of Definition 12 with respect to the symmetry need to be done, since faults may occur both in accuracy and associated precision intervals, as well as independently in rate and associated consonance intervals.

4.3. Simulation Results

We repeat the simulation with the distributed system described in Section 2, but instead of algorithm FTA we execute our state&rate algorithm from Definition 9. Details about implementing it with our SimUTC toolkit can be found in [16]. As before, we simulate $n = 5$ nodes having $f = 0$ faults, and synchronize the clock state at every $P_S = 12$ s. To make the algorithm work, we have chosen $m = 4$, $\Delta = 1.83$ ms, $E_p = 0$ s, and $\boldsymbol{\sigma}_p = [-0.01, 0.01]$ ppm/s. The latter is justified by the upper right plot of Figure 1, since it displays the clock drifts in absence of rate adjustments. The remaining parameters for the convergence functions are as follows:

$\mathcal{OP}_{n-f}^{\boldsymbol{\pi}^o,\boldsymbol{\pi}^H,\Upsilon_{\max}}(\cdot)$	$\mathcal{OP}_{n-f}^{\boldsymbol{\gamma}^o,\boldsymbol{\gamma}^H,\Theta_{\max}}(\cdot)$
$\boldsymbol{\pi}^o = [-220, 220]\ \mu$s	$\boldsymbol{\gamma}^o = [-4, 4]$ ppm
$\boldsymbol{\pi}^H = [-220, 220]\ \mu$s	$\boldsymbol{\gamma}^H = [-4, 4]$ ppm
$\Upsilon_{\max} = 300\ \mu$s	$\Theta_{\max} = 10$ ppm

A segment of the simulation run is shown in the eight plots of Figure 4, where the upper four plots correspond with those in Figure 1. It is immediately apparent that the previous sawtooth-like shape of the traditional accuracies $\alpha_p(t)$ has almost vanished, in fact, the synchronization is now dominated by the transmission delay uncertainties $\varepsilon^{\pm}_{\max}$. This is also the reason for the ragged precision $\pi(t)$, which is well below 15 μs. The effects of rate synchronization can be observed by the drift plots $\theta_p(t)$, which are now squeezed between 0 and 1.1 ppm, reducing the consonance $\gamma(t)$ to a maximum of 0.7 ppm. We can only reach smaller values when the oscillators are more stable.

The plots in the third row of Figure 4 display the evolution of the accuracy intervals $\boldsymbol{A}_p(t)$ and rate intervals $\boldsymbol{R}_p(t)$. The way of drawing them is the same as in Figure 2, but a (small) fixed t-offset is imposed for each node to separate them at simultaneous snapshot times. The average growth of the accuracy intervals reads about ± 1.8 μs/s and of the rate intervals about ± 0.01 ppm/s.

The state&rate adjustments during the execution of our algorithm are shown in the remaining plots of Figure 4. In particular, the bottom left plot depicts the difference $\Upsilon'_p = A'_p - (k_S P_S + \Delta + E_p)$ between the new and the old clock state, and the bottom right one the ratio $\Theta'_p(t) = S'_p(t)/S_p(t)$ between the new and the old coupling factor. State adjustments happen at each state period $P_S = 12$ s without exceeding the 12 μs mark, whereas rate adjustments happen at each rate period $P_R = mP_S = 48$ s and are delimited by 0.4 ppm in this run.

4.4. Discussion

The algorithm from Definition 9 aims to achieve a small precision/accuracy as well as a small consonance/drift, but there are mutual dependencies between these synchronizations. Evidently, the state synchronization takes advantage of the disciplined clock rates, whereas the rate synchronization subsists on the round structure. In the view of the rate feature, the "incoming" maximum precision $\pi_{\max}$ relies on the "outgoing" maximum consonance $\gamma_{\max}$ and maximum drift $\theta_{\max}$. Note that the clocks' accuracies are not relevant for the rate synchronization. Fortunately, the inspections in [15] reveal that $\gamma_{\max}$ depends only loosely on $\pi_{\max}$, which allows us to have a separated analysis, putting the state part on top of the rate one.

Based on this observation we claim that the rate part of our state&rate algorithm from Definition 9 can be combined with any one for state synchronization, e.g., the *optimal* algorithm by [18], the *a-posteriori agreement* algorithm by [20], the *probabilistic* algorithm by [2], or the framework by [13]. It only requires a rate adjustable clock and the possibility to exchange rate intervals within rounds to extend those approaches, since they all rely on the ordinary hardware drift rate and not on the oscillator stability.

Let us review a few results from the analysis [15] and how they relate to our simulation run in Figure 4. First of all, the simplified expression for the maximum consonance turns out as

$$\gamma_{\max} = 6P_R\sigma_{\max} + 4\epsilon_{\max}/P_R, \tag{26}$$

therefore $\gamma_{\max} = 2.88 \text{ ppm} + 2.5 \text{ ppm} = 5.38$ ppm. The contribution from the oscillator stability and from the transmission delay uncertainty is almost in balance, thus our rate period P_R is close to the optimum $\sqrt{(2\epsilon_{\max})/(3\sigma_{\max})} = 44.72$ s. The maximum consonance from (26) overestimates the *average consonance* $\gamma_{\text{avg}} = 0.45$ ppm in our simulation run by around one order of magnitude, which is not surprising, since the above formula arises from worst case scenarios and the stability parameter is too conservative. Furthermore, the analysis provides a worst case rate adjustment of $\pm 2P_R\sigma_{\max} = \pm 0.96$ ppm and a worst case growth of rate intervals of $\pm 3\sigma_{\max} = \pm 0.03$ ppm, which are both not very tight bounds.

The deviation between the *average precision* $\pi_{\text{avg}} = 8.85$ μs in our simulation run and the analytical maximum is even worse, since the latter is given by the simplified expression

$$\pi_{\max} = 2\varepsilon_{\max} + 2P_S\gamma_{\max}, \tag{27}$$

therefore $\pi_{\max} = 60\ \mu\text{s} + 129\ \mu\text{s} = 189\ \mu$s. As expected, the maximum precision is determined by the maximum uncertainty from the transmission delay and the maximum consonance. The overestimation of $\gamma_{\max}$ leads to a very bad value of $\pi_{\max}$, however, by replacing it with the average consonance, we get a contribution of not more than 11 μs, which indicates a reasonable rate synchronization compared to the achievable state synchronization. Finally, the analysis yields a worst case state adjustment of $\pm P_S\gamma_{\max} = \pm 64.56\ \mu$s, which is again a weak bound. Note that the growth of accuracy intervals is immediately linked to the lengths of the local rate intervals.

5. Conclusions

The precision of traditional clock state synchronization approaches can be enhanced by an additional clock rate synchronization, because without it the consonance of the clocks in a distributed system might be as much as $2\rho_{\max}$. However, we discovered that the effort to synchronize the clock rates is primarily limited by the oscillator stability $\sigma_{\max}$. In this paper we have constructed a fault-tolerant algorithm for synchronizing both the state and rate of clocks, which is based on intervals, has the advantage of being cheap in terms of communication as well as computation cost, and could be implemented without difficulties.

Future work is devoted to study simulation experiments with the help of our SimUTC toolkit including the injections of faults. This is a rather complex undertaking, since faults can impact external/internal and state/rate synchronization in interdependent ways and with different degrees of severity. Moreover, we intend to investigate the usage of other convergence functions including the integration of rate synchronization with existing clock state algorithms, to improve the characterization of the oscillator stability, and to prove optimality results for state&rate synchronization.

6. Acknowledgement

We gratefully acknowledge Ulrich Schmid, the head of project SynUTC, for his thorough comments and suggestions to enrich this paper. The authors are also thankful for the valuable input given by the anonymous reviewers.

References

[1] M. H. Azadmanesh and R. M. Kieckhafer. New hybrid fault models for asynchronous approximate agreement. *IEEE Transactions on Computers*, 45(4):439–449, 1996.

[2] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158, 1989.

[3] C. Fetzer and F. Cristian. An optimal internal clock synchronization algorithm. In *Proceedings of the 10th Annual IEEE Conference on Computer Assurance*, pages 187–196, Gaithersburg, MD, June 1995.

[4] L. Lamport. Synchronizing time servers. Technical Report 18, Digital System Research Center, June 1987.

[5] B. Liskov. Practical uses of synchronized clocks in distributed systems. *Distributed Computing*, 6(4):211–219, 1993.

[6] J. Lundelius-Welch and N. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2/3):190–204, 1984.

[7] J. Lundelius-Welch and N. Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and Computation*, 77(1):1–36, 1988.

[8] K. Marzullo. *Maintaining the time in a distributed system. An example of a loosely-coupled distributed service*. PhD thesis, Department of Electrical Engineering, Stanford University, February 1984.

[9] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems*, 8(4):284–304, November 1990.

[10] U. Schmid. Interval-based clock synchronization with optimal precision. Technical Report 183/1-78, Department of Automation, Vienna University of Technology, December 1997.

[11] U. Schmid, J. Klasek, T. Mandl, H. Nachtnebel, G. R. Cadek, and N. Kerö. A network time interface M-Module for distributing GPS-time over LANs. *Real-Time Systems*, 17(2):–, September 1999.

[12] U. Schmid and K. Schossmaier. Interval-based clock synchronization. *Real-Time Systems*, 12(2):173–228, March 1997.

[13] F. B. Schneider. Understanding protocols for byzantine clock synchronization. Technical Report 87-859, Department of Computer Science, Cornell University, August 1987.

[14] K. Schossmaier. An interval-based framework for clock rate synchronization. In *Proceedings of the 16th Annual ACM Symposium on Principles on Distributed Computing (PODC)*, pages 169–178, Santa Barbara, CA, 21–24 August 1997.

[15] K. Schossmaier. *Interval-based Clock State and Rate Synchronization*. PhD thesis, Faculty of Technical and Natural Sciences, Vienna University of Technology, September 1998.

[16] K. Schossmaier and J. Klasek. Implementing the optimal precision algorithm for clock state&rate synchronization. Technical Report 183/1-91, Department of Automation, Vienna University of Technology, February 1999.

[17] K. Schossmaier, U. Schmid, M. Horauer, and D. Loy. Specification and implementation of the Universal Time Coordinated Synchronization Unit (UTCSU). *Real-Time Systems*, 12(3):295–327, May 1997.

[18] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, July 1987.

[19] D. B. Sullivan, D. W. Allan, D. A. Howe, and F. Walls. Characterization of clocks and oscillators. *National Institut of Standards and Technology (NIST) Technical Note*, 1337:1–342, March 1990.

[20] P. Veríssimo and L. Rodrigues. A-posteriori agreement for fault-tolerant clock synchronization on broadcast networks. In *Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing (FTCS)*, pages 527–535, July 1992.

[21] B. Weiss. Simulation environment for clock synchronization. Technical Report 183/1-88, Department of Automation, Vienna University of Technology, February 1999.

[22] B. Weiss, G. Gridling, U. Schmid, and K. Schossmaier. The SimUTC fault-tolerant distributed systems simulation toolkit. In *Proceeding of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, College Park, MD, 24–28 October 1999.