An Algorithm for Fault-Tolerant Clock State&Rate Synchronization

Klaus Schossmaier

Bettina Weiss

Vienna University of Technology

Department of Automation

20 October 1999

http://www.auto.tuwien.ac.at/Projects/SynUTC

Introduction

Problem: Provide the nodes of a distributed system with

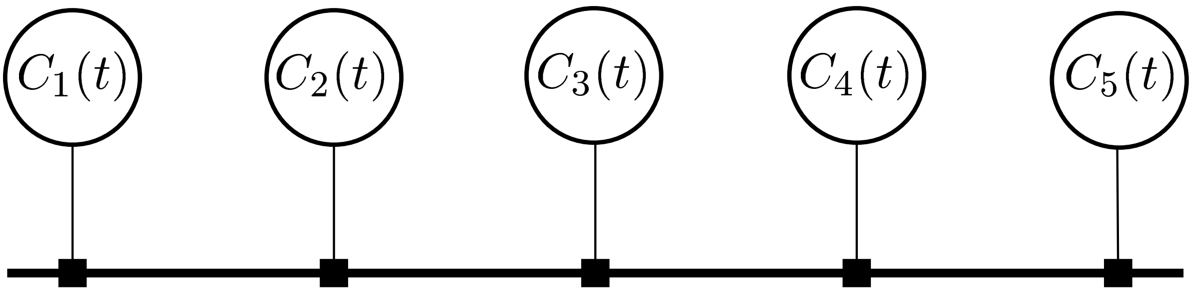
- internal ("precise")
- external ("accurate")

reliably synchronized hardware clocks.

Benefits:

- applications dealing with time
- implementation of distributed services
- improvement of distributed algorithms
- monitoring the system

SIMULATION EXPERIMENT



Parameters:

- average clock drifts:

  0 ppm, 0.5 ppm, −0.5 ppm, 2 ppm, −1 ppm

- packet transmission delay:

  $\delta = 60\ \mu$s, $\epsilon^-_{\max} = 10\ \mu$s, $\epsilon^+_{\max} = 20\ \mu$s

- algorithm:

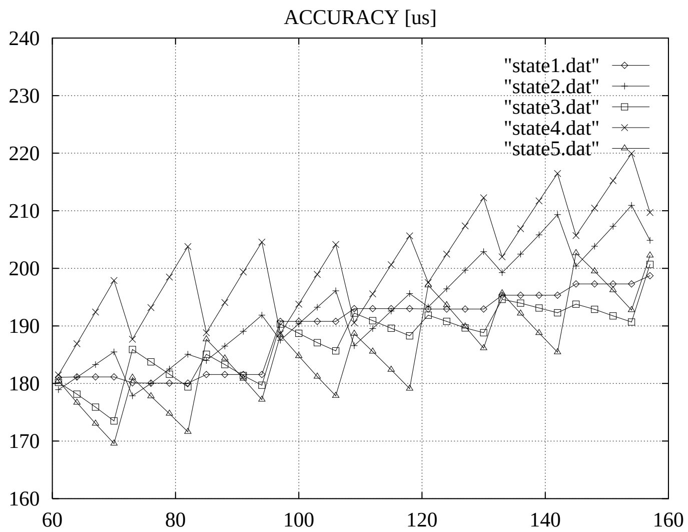
  Fault-Tolerant Average (FTA)

  $P_S = 12$ s

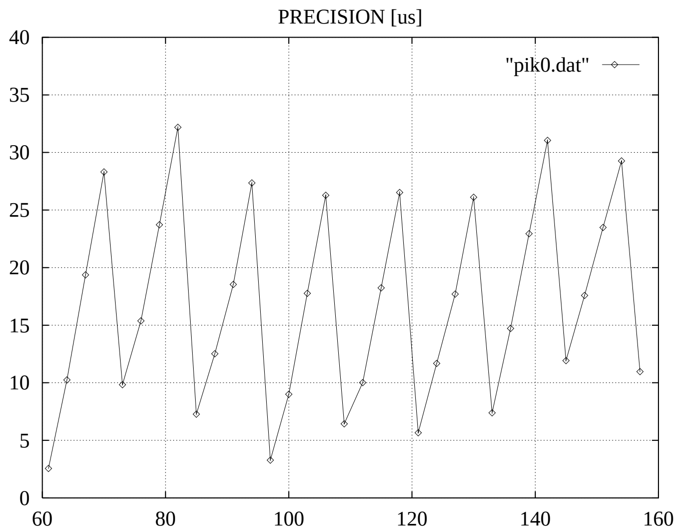
Toolkit: SimUTC

Clock State Synchronization

Accuracy: $\alpha_p(t) = C_p(t) - t$

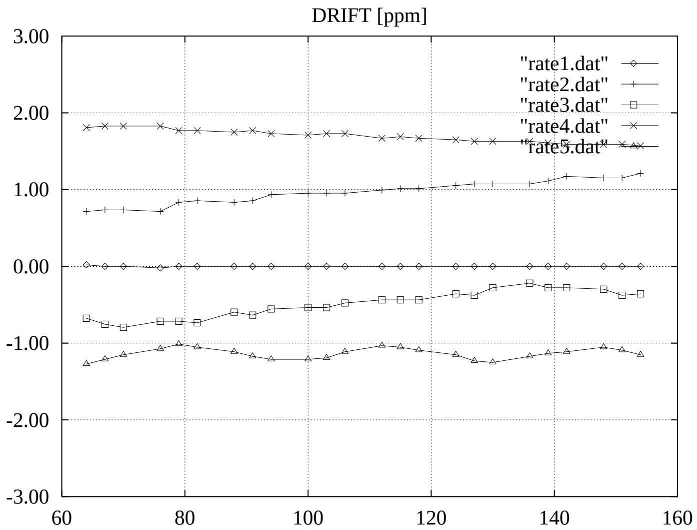


Precision: $\pi(t) = \max_{p,q}\{C_p(t) - C_q(t)\}$

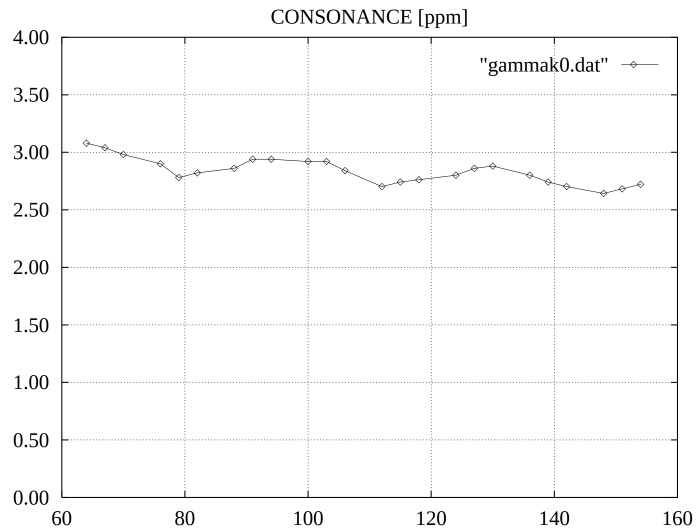


CLOCK RATE SYNCHRONIZATION

Drift: $\theta_p(t) = v_p(t) - 1$



Consonance: $\gamma(t) = \max_{p,q}\{v_p(t) - v_q(t)\}$



Clock State&Rate Synchronization

Our algorithm achieves state and rate synchronization and offers the following novelties:

Exchanging State&Rate information

$\longrightarrow$ based on intervals

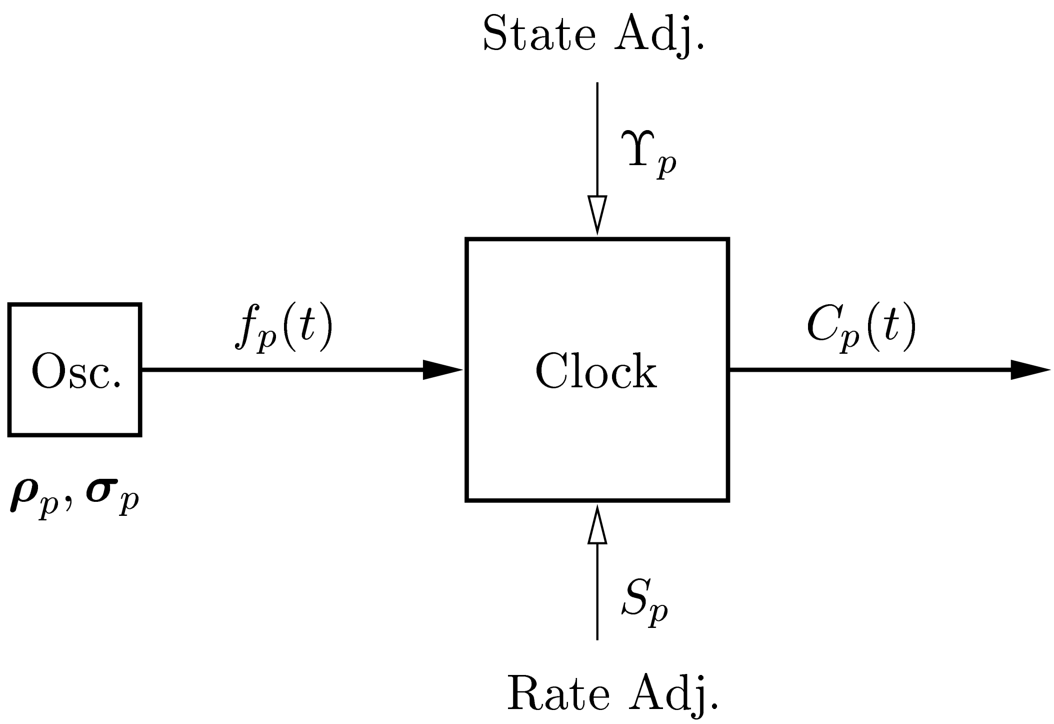
Computing State&Rate adjustments

$\longrightarrow$ advanced convergence function and fault model

Synchronizing State&Rate of clocks

$\longrightarrow$ Oscillator-clock model

Oscillator-Clock Model



Hardware: UTCSU-ASIC placed on an M-Module

Oscillator Assumptions

Oscillator drift:

$$\frac{f_p(t)}{f_p} \in [1 \pm \boldsymbol{\rho}_p]$$

Oscillator stability:

$$\frac{f_p(t + \Delta t)}{f_p(t)} \in [1 \pm \boldsymbol{\sigma}_p \Delta t]$$

Example:

Drift Interval $\boldsymbol{\rho}_p = \left[-10^{-6}, 10^{-6}\right]$

Stability Interval $\boldsymbol{\sigma}_p = \left[-10^{-11}, 10^{-11}\right]$

Local Intervals

Each node maintains two intervals:

Accuracy interval: $\boldsymbol{A}_p = [C_p(t) \pm \boldsymbol{\alpha}_p(t)]$

to capture the clock state such that

$$t \in \boldsymbol{A}_p$$

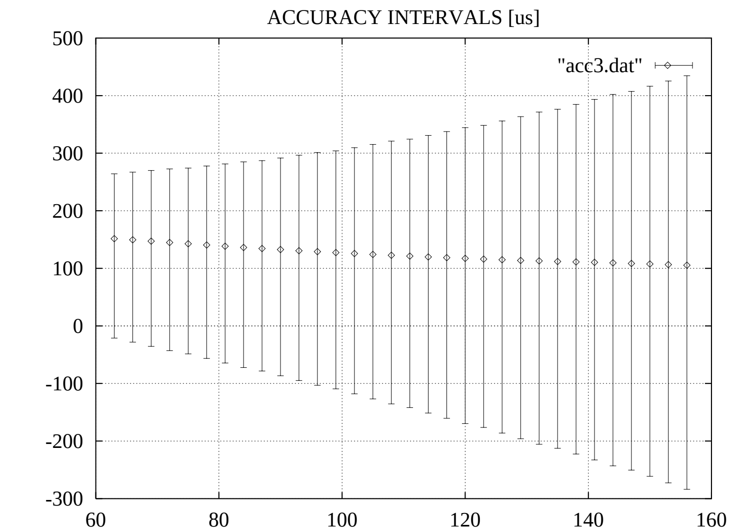
Rate interval: $\boldsymbol{R}_p = [1 \pm \boldsymbol{\theta}_p(t)]$

to capture the clock rate such that

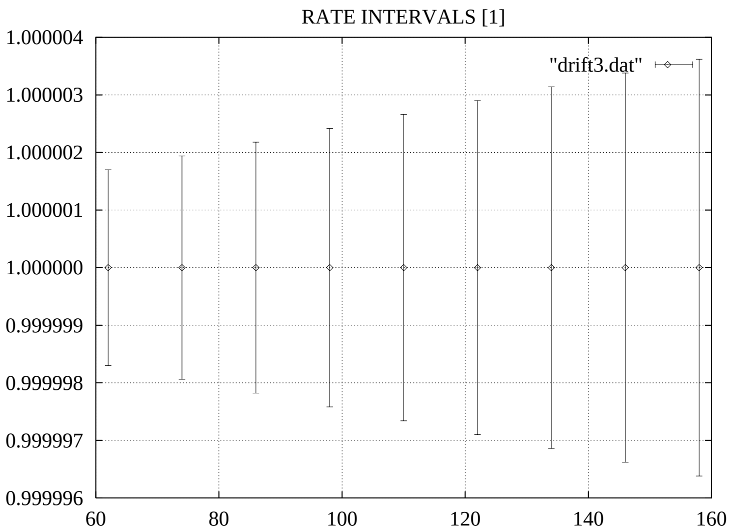
$$1/v_p(t) \in \boldsymbol{R}_p$$

Maintenance

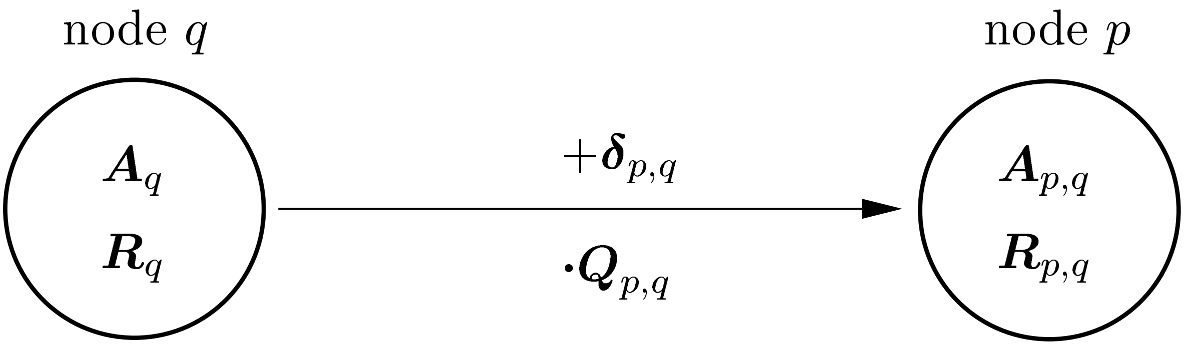
Drift compensation: $\boldsymbol{A}_p + \Delta T \boldsymbol{R}_p$



Stability compensation: $\boldsymbol{R}_p + \Delta T \boldsymbol{\sigma}_p$



REMOTE INTERVALS



Delay interval: $\boldsymbol{\delta}_{p,q} = [\delta_{p,q} \pm \boldsymbol{\epsilon}_{p,q}]$

to capture the transmission delay such that

$$\Delta t_{p,q} \in \boldsymbol{\delta}_{p,q}$$

Quotient rate interval: $\boldsymbol{Q}_{p,q} = \left[\frac{\Delta T_q}{\Delta T_p} \pm \boldsymbol{\nu}_{p,q}\right]$

to capture the relative rate such that

$$\frac{v_q}{v_p} \in \boldsymbol{Q}_{p,q}$$

State&Rate Algorithm

Each node p performs periodically:

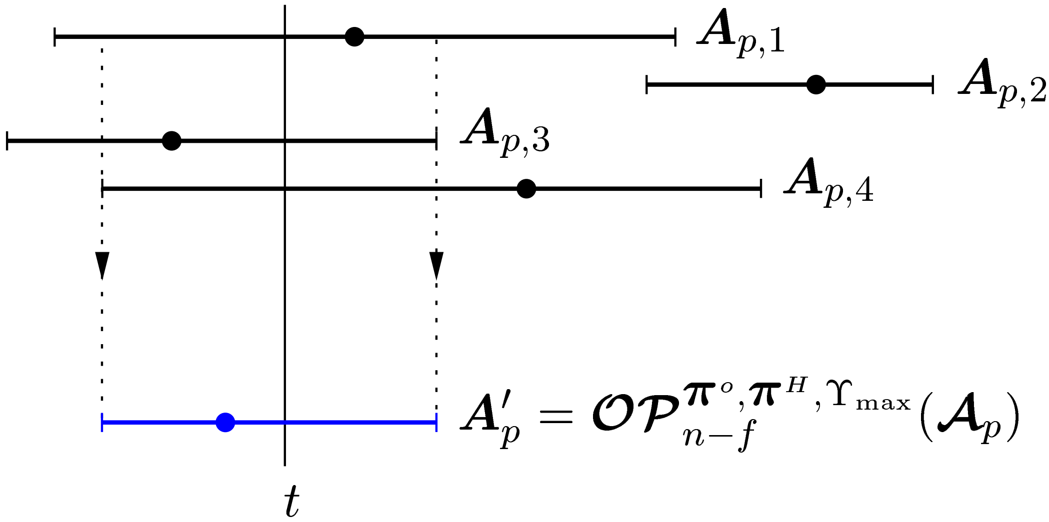
Send: broadcast packets $\langle \boldsymbol{A}_p, \boldsymbol{R}_p, U_p \rangle$

Preprocessing: receive packets from remote nodes q and compute set $\boldsymbol{\mathcal{A}}_p$ of remote accuracy intervals and set $\boldsymbol{\mathcal{R}}_p$ of remote rate intervals

Computation: calculate new accuracy interval $\boldsymbol{A}'_p$ and new rate interval $\boldsymbol{R}'_p$

Adjustment: enforce new coupling factor S_p and state increment Υ_p

Convergence Function



Optimal precision function $\mathcal{OP}$:

- extension of Marzullo's function
- parameters $\boldsymbol{\pi}^o, \boldsymbol{\pi}^H$ to set the reference point
- maximal adjustment $\Upsilon_{\max}$
- fault-tolerance parameter f

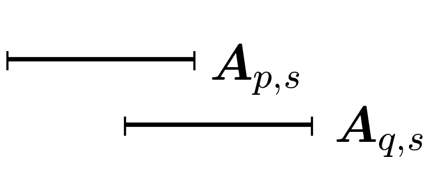
Abstract Fault Model

Faults are not viewed in terms of system failures, but rather as incorrect pairs of preprocessed intervals:

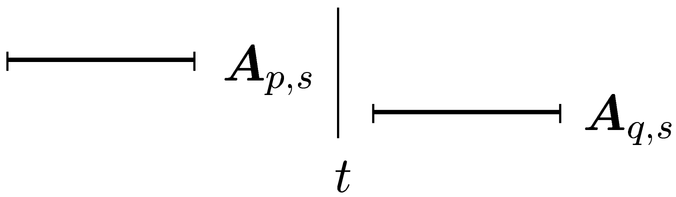
node s sends interval $\boldsymbol{A}_s$ to node p and q

crash fault $\boldsymbol{A}_{p,s} = \emptyset, \boldsymbol{A}_{q,s} = \emptyset$

symmetric fault



asymmetric fault

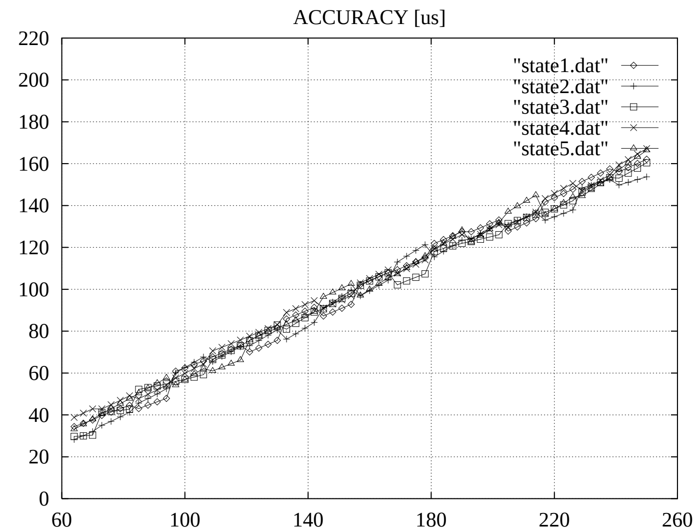


restricted fault when symmetric or crash

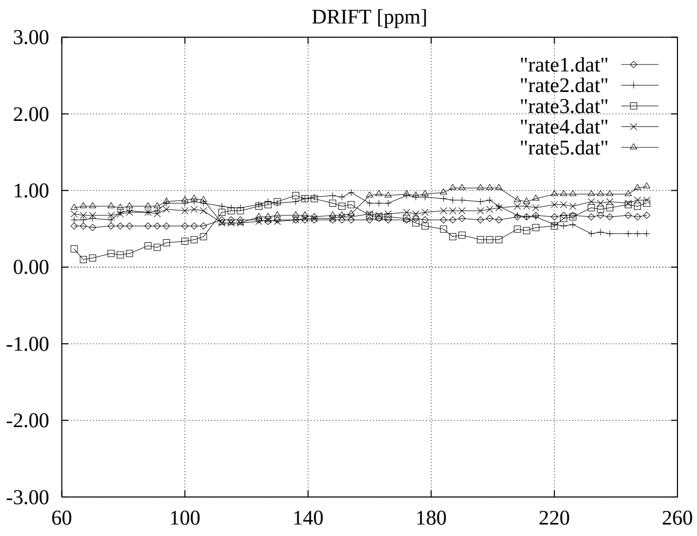
arbitrary fault when asymmetric or restricted

Simulation Results

State: $\pi_{\text{avg}} = 8.85\ \mu\text{s}$ $(16.83\ \mu\text{s})$



Rate: $\gamma_{\text{avg}} = 0.45$ ppm $(2.83$ ppm$)$



Analytical Results

Maximum consonance:

$$\gamma_{\max} = \underbrace{6P_R\sigma_{\max}}_{2.88\ \mathrm{ppm}} + \underbrace{4\epsilon_{\max}/P_R}_{2.5\ \mathrm{ppm}}$$

Maximum precision:

$$\pi_{\max} = \underbrace{2\varepsilon_{\max}}_{60\ \mu s} + \underbrace{2P_S\gamma_{\max}}_{129\ \mu s}$$

Maximum adjustment:

$$|\Upsilon_{\max}| = \underbrace{P_S\gamma_{\max}}_{64.5\ \mu s}$$

Final Comments

Conclusions:

- rate synchronization required for high-precision
- analogy between state and rate synchronization
- consonance is limited by oscillator stability
- algorithm is cheap and easy to implement

Future work:

- other convergence functions
- simulation experiments with fault injection
- optimality results for State&Rate synchronization