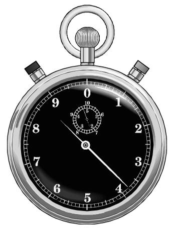
Clock Synchronization

Henrik Lönn
Electronics & Software
Volvo Technological Development



Parallel and Distributed Systems: Clock Synchronization

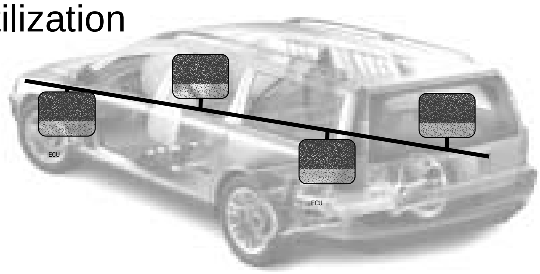
Contents

- General
- Types of Synchronisation
- Faults and problems to cope with
- Example algorithms
- Transmission delays
- Derivation of criterion n>3m
- Derivation of clock precision

VTD Electronics and Software

Why do we need an agreed time?

- Consistent inputs/outputs
- Control loops
- Rollback recovery
- Time-triggered task execution
- Resource utilization



Clock Synchronization

- Internal vs. External
- Master-Slave vs. Distributed
- Hardware vs. software

Synchronization Goals:

- High Precision -Small deviation, or skew, between clocks
- High Accuracy - Small deviation to external time
- Monotonic and “Continuous” time

Clock drift

$$\left|\tfrac{d}{dt}C_i - 1\right| < \rho$$

or

$$(1-\rho)(t_2 - t_1) \le t_2 - t_1 \le (1+\rho)(t_2 - t_1)$$

- Why should clock drift be small?

  Generally: All users of the clock service require a certain accuracy & precision

  => Clock synchronization must be performed more often with large drift

How often should we re-synchronize?

- ρ: A rate deviation from external time (s/s)
- R: Time between synchronizations
- Δ_s: Precision after synchronization

Skew increase by R*2ρ between two synchronizations

$\Rightarrow$ Choose R so that $\Delta_s + R*2\rho <$ Req. precision

How often should we re-synchronize?

- Loosely coupled systems:
  - Inaccurate, expensive clock readings=>
    Use high accuracy clocks, synchronize seldom
- Tightly coupled systems:
  - Accurate clock readings=>
    Use low accuracy clocks and synchronize often

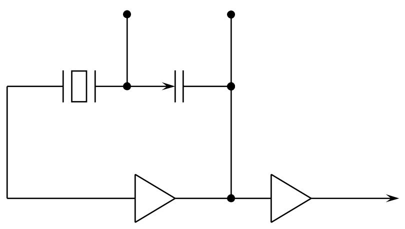
($\Delta_s + R*2\rho <$ Req. precision)

Rate adjustment

- Clock rate is adjusted
  - Required adjustment: Δ
  - Time between adjustments: R
  - => Adjust rate with Δ/R

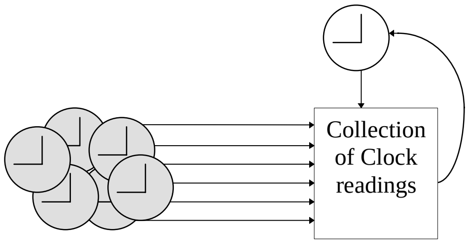
Oscillators

- Inverters
- RC oscillators 10^{-3}
- Quartz oscillators 10^{-6}



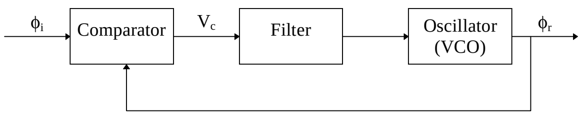
Hardware clock synchronization

- Use a “clock distribution network”
  - Much hardware required
  - Very good precision



Phase Locked Loop

≈feedback control loop



Full connectivity not required

A *cluster* is fully connected internally and connected to at least one clock in each cluster

$p_i + M - 1 \geq 3m+1$

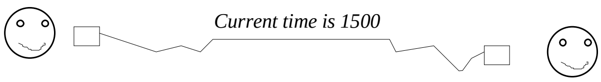
"Number of Clock inputs > 3 times number of faults"

- M number of clusters
- p_i clocks in cluster i
- m faulty clocks are tolerated

Parallel and Distributed Systems: Clock Synchronization

Software clock synchronisation:

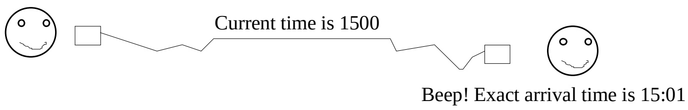
- Use messages
  - Cheap in hardware
  - Flexible
  - Low precision



VTD Electronics and Software

Hardware assisted Software clock synchronisation:

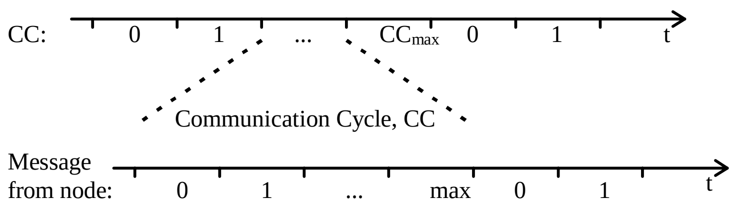
- Use messages whose exact point of arrival is recorded by hardware
  - Cheap in hardware
  - high precision on suitable topology



Parallel and Distributed Systems: Clock Synchronization

Use of pre-scheduled events as clock readings

TDMA: Time Division Multiple Access



VTD Electronics and Software

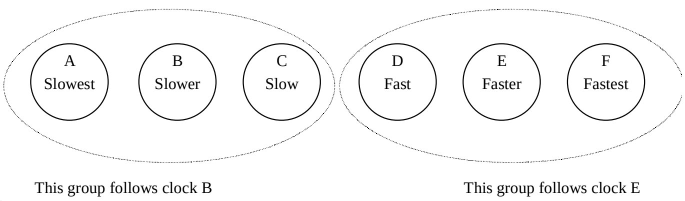
Faults

- Excessive drift
- Clock reading errors
- Byzantine faults (“Dual-faced” clocks)
  - The faulty clock gives different readings to different nodes

Cliques

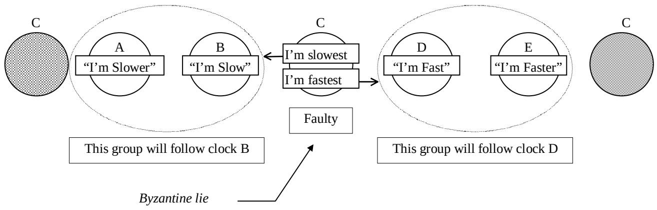
Clock groups that are mutually unsynchronized

- Clock synch. algorithm must prevent this!



Byzantine faults

- Effect on a ”sensible” algorithm: *(Syncronize to median of clock readings):*



- Solution: Use a fault tolerant algorithm.
- 3m+1 clocks to tolerate m clock failures

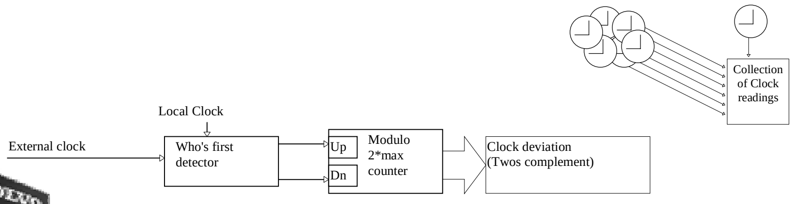
Convergence vs Consistency Algorithms

The presented algorithms are of this type

- Interactive convergence algorithms
  - Adjustment function guarantees limited clock skew in spite of faulty clocks and different sets of clock readings
- Interactive consistency algorithms
  - Agreement algorithm guarantees that all sets of clock readings are identical. Adjustment function is “arbitrary”, take e. g. median

Interactive convergence algorithm (CA1)

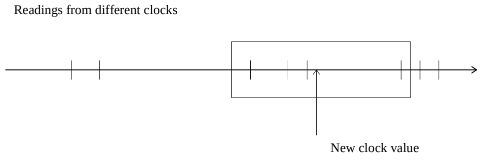
- Collect clock readings
- Replace values larger than max with an arbitrary value smaller than max
- Take average of clock readings



Parallel and Distributed Systems: Clock Synchronization

Fault Tolerant Average algorithm (CA2)

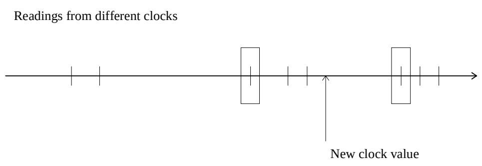
- Mean of clocks excluding t fastest and t slowest



VTD Electronics and Software

Midpoint algorithm

- Mean of fastest and slowest excluding t fastest and t slowest

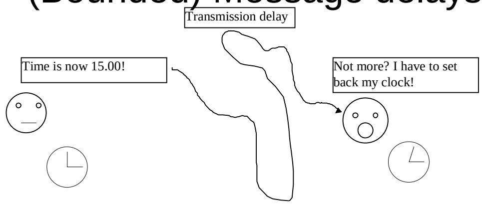


Convergence Nonaveraging Algorithm (CNA)

- Resynchronization interval=R
- At synchronization i:
  - If clock reaches iR, send synchronization message
  - Adjust time to iR if synchronization message arrive in time, and relay message
  - Ignore synchronization messages that arrive too early or too late (adjustment i has been done)

Transmission delays:

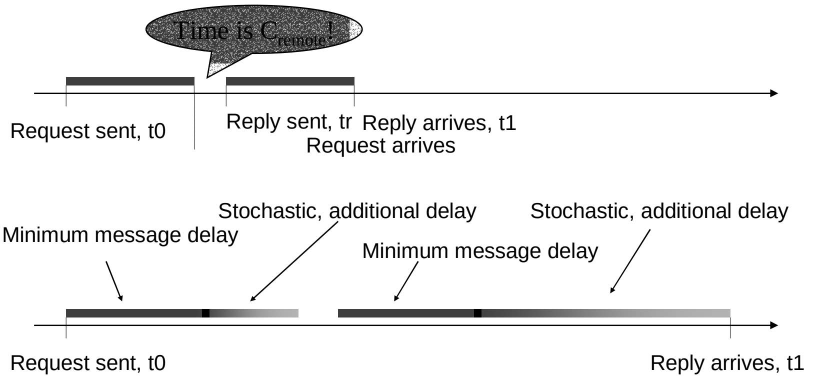
- Hardware/Hybrid environment:
  Speed of light 3 ns/m
- Software environment:
  (Bounded) Message delays



Reducing effects of message delay in loosely coupled systems:

- Incoming messages receive a timestamp by hardware
- Before a message is relayed, its clock value is increased by the time spent in the node

Probabilistic Clock Synch.



Parallel and Distributed Systems: Clock Synchronization

Probabilistic Clock Synch.

1) Request reference clock reading from master

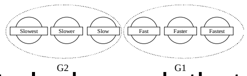
- Request sent at t_0 and clock reading C_{remote} received at t_1

2) Assume that reading instant is $C_{local}=(t_0+t_1)/2$ and calculate clock deviation $C_{local} - C_{remote}$

Make multiple readings until t_1-t_0 < threshold

- Total delay=t_1-t_0
- Total stochastic delay, $d_{stoch}=(t_1-t_0) -2*d_{min}$
- $t_0+d_{min}< t_r<t_0+d_{min}+ d_{stoch}$
- Choose midpoint => error<$d_{stoch}/2$

Correctness Criteria

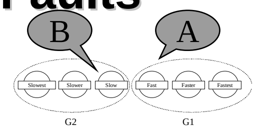


For all partitions G1 and G2 of correct clocks such that all clocks in G1 are faster than those in G2:

C1: If all clocks in G1 use a reference that is faster than all clocks in G2
then there must be at least one clock in G2 that uses a reference that is at least as fast as the slowest clock in G1 *and vice versa*

C2: If a clock uses a reference from a faulty clock, there must be correct signals that are slower and faster than the reference.
i. e. Faulty clocks that are used as a reference must be sandwiched between correct clocks

Tolerating Byzantine Faults



- Correctness criterion C1=>

If $\max_{x \in G1} f_{p(x)}(N,m) \le \|G1\| + m$

then $\min_{y \in G2} f_{p(y)}(N,m) \le \|G1\|$

x

(If "no clock in G1 use reference in G2" then "a clock in G2 must use ref. in G1")

If $\min_{y \in G2} f_{p(y)}(N,m) \ge \|G1\| + 1$ (1)

then $\max_{x \in G1} f_{p(x)}(N,m) \ge \|G1\| + m + 1$ (2)

(If "no clock in G2 use reference in G1" then "a clock in G1 must use ref. in G2")

$f_{p(i)}(N,m)$: Number of the reference clock used by clock i

(Clock readings numbered "fastest first". There are N clocks and m Byz. faults)

Tolerating Byzantine Faults

(2)-(1) $\Rightarrow$

$$\max_{x \in G1} f_{p(x)}(N,m) - \min_{y \in G2} f_{p(y)}(N,m) \geq m \quad (3)$$

Avoid faulty clocks (Criterion C2) $\Rightarrow$

$$\max_{x \in G1} f_{p(x)}(N,m) \leq N - m$$

$$\min_{x \in G2} f_{p(y)}(N,m) \geq m + 1$$

Substitute in (3) $\Rightarrow$

N-m - (m+1) ≥ m; <u>N ≥ 3m+1</u>

Attainable precision, CA2

Precision after synchronization: Δ_s

Clock precision: δ

Reading error: ε

Synchronization interval: R

$\Delta_s = m\delta/(N-2m) + \varepsilon$

$\delta = \Delta_s + 2R\rho = m\delta/(N-2m) + \varepsilon + 2R\rho$

$\Rightarrow \delta = (\varepsilon + 2R\rho)((N-2m)/(N-3m))$

Attainable precision, CA2

Include uncertainty in R and time when correction is applied $\Rightarrow$

$\delta = \Delta_s + \rho\Delta_s + 2\rho(1+\rho)R = (m\delta/(N-2m) + \varepsilon + 2R\rho)\ (1+\rho)$

$\Rightarrow \delta = (1+\rho)(\varepsilon + 2R\rho)\ ((N-2m)/(N-3m-m\rho))$

Parallel and Distributed Systems: Clock Synchronization

Summary

- Reasons for clock synchronisation
- Some problems to cope with
- Types of clock synchronisaton
- Example algorithms
- Derivation of attainable clock precision for one algorithm

VTD Electronics and Software