Distributed Systems
MIEEC, Fall 2018

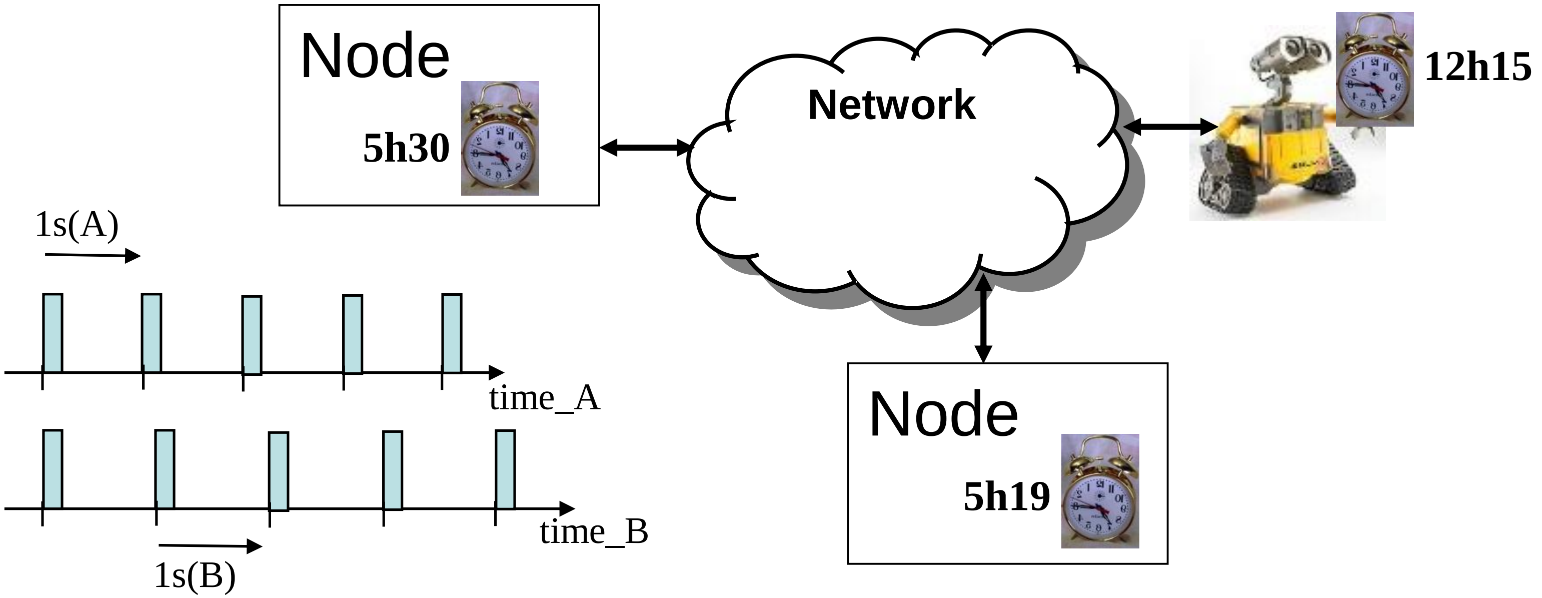
Clock Synchronization

Luis Almeida
Faculty of Engineering – **University of Porto**
Portugal

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Time across a network

- In a **distributed system** each node has its **own clock**
  - Without specific support, there is **no explicit coherent notion of time** across a distributed system
  - Worse, due to **drift**, clocks tend to **permanently diverge**



Time across a network

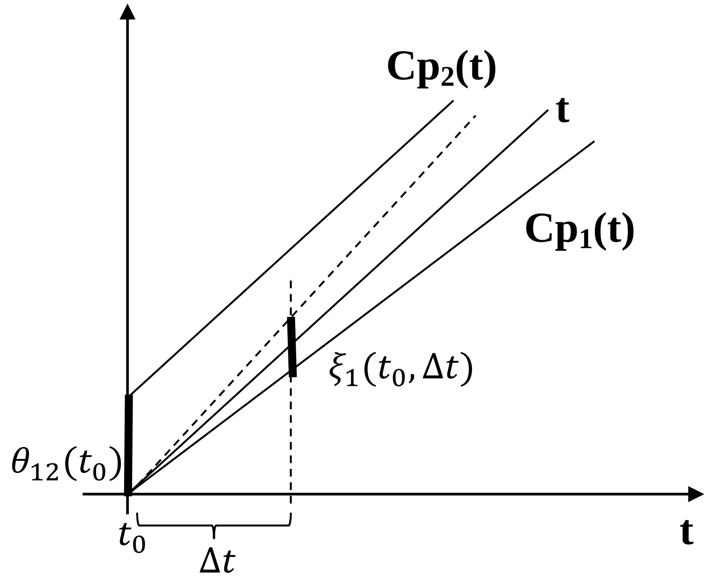
- **Why** developing a **coherent notion of time?**
  - Carry out **actions** at **desired time** instants
    - e.g. synchronous data acquisition, synchronous actuation
  - **Time-stamp** data and events
    - e.g. establish causal relationships that led to a system failure
  - Compute the **age** of data
  - **Coordinate** transmissions
    - e.g. TDMA clock-based systems

But how to synchronize the clocks across the network?

Few definitions

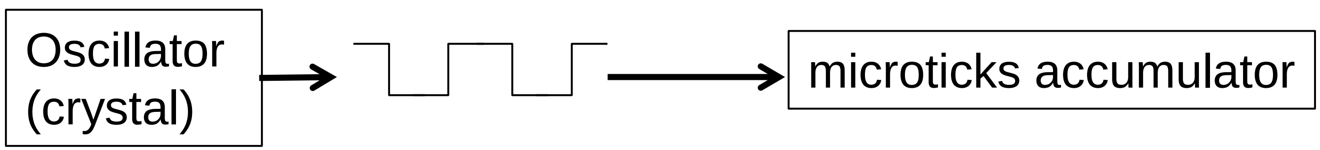
- **Offset**
  - $\theta_{ij}(t) = \left|Cp_i(t) - Cp_j(t)\right|$
- **Drift rate** and **drift**
  - $\rho_i(t) = \left|\frac{(Cp_i(t+\Delta t) - Cp_i(t))}{\Delta t} - 1\right|$
  - $\xi_i(t, \Delta t) = 2 * \rho_i(t) * \Delta t$
- **Accuracy** α
  - $\max_{it}\left|Cp_i(t) - t\right| \leq \alpha$
- **Precision** δ
  - $\max_{ijt}\left|Cp_i(t) - Cp_j(t)\right| \leq \delta \Leftrightarrow \max_{ijt}\left(\theta_{ij}(t)\right) \leq \delta$

$Cp_i(t)$ is the **clock** of **node i** at **instant t**



Digital clocks

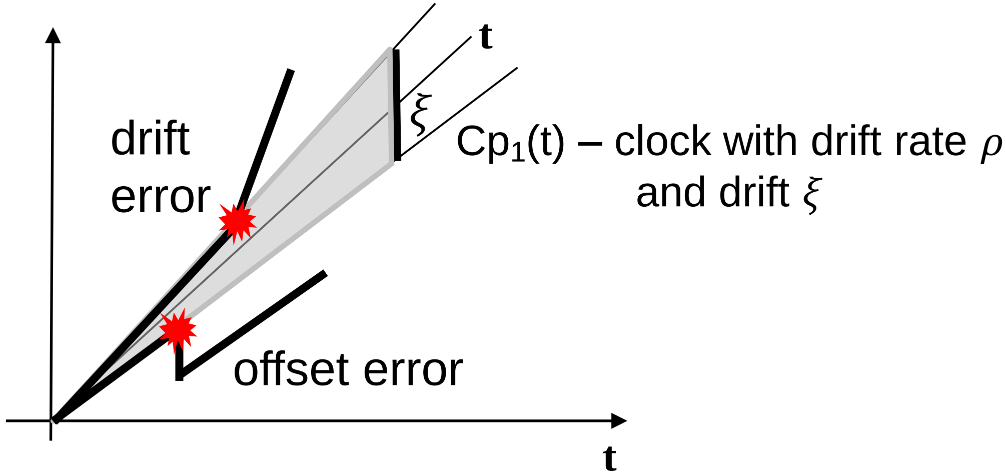
- A digital **clock** is a **counter** incremented every **tick** (fixed interval)
  - A **tick** is implemented counting a **fixed number** (n) of **microticks** that represent oscillator pulses



- Main **clock parameter** is the **drift rate**
  - Real clocks have drift rates between **10^{-2}** and **10^{-8}** depending on the quality of their oscillators

Fault model of digital clocks

- Clocks can suffer **offset errors** and **drift errors**



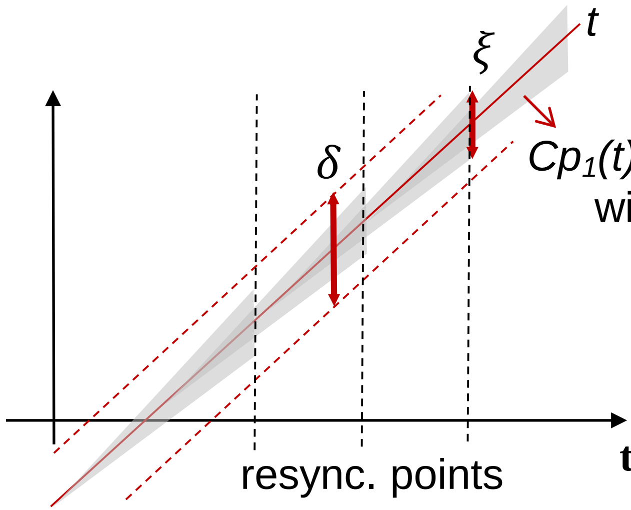
  - Offset errors are stochastic errors in tick counting
  - Drift errors can be **sistematic** (due to inherent drift rate) and **stochastic**
  - **Sistematic drift >>** stochastic drift
    - allows algorithmic correction → **clock synchronization**

Clock synchronization

- **Clocks can be synchronized:**
  - **Externally** – an external source sends a time update regularly (e.g. GPS)
    - Quality metric: **accuracy**
  - **Internally** – nodes exchange messages to come up with a global clock
    - **Master-Slave** – The time master spreads its own clock to all other nodes
    - **Distributed** – All nodes perform a similar role and agree on a common clock, typically an average
    - Quality metric: **precision**
  - Both methods are complementary
    - **Internal** synchronization provides **high availability** and **good short-term stability**
    - **External** synchronization provides **long-term stability** but has lower availability
- Standards:
  - NTP, SNTP, IEEE 1588 (PTP)

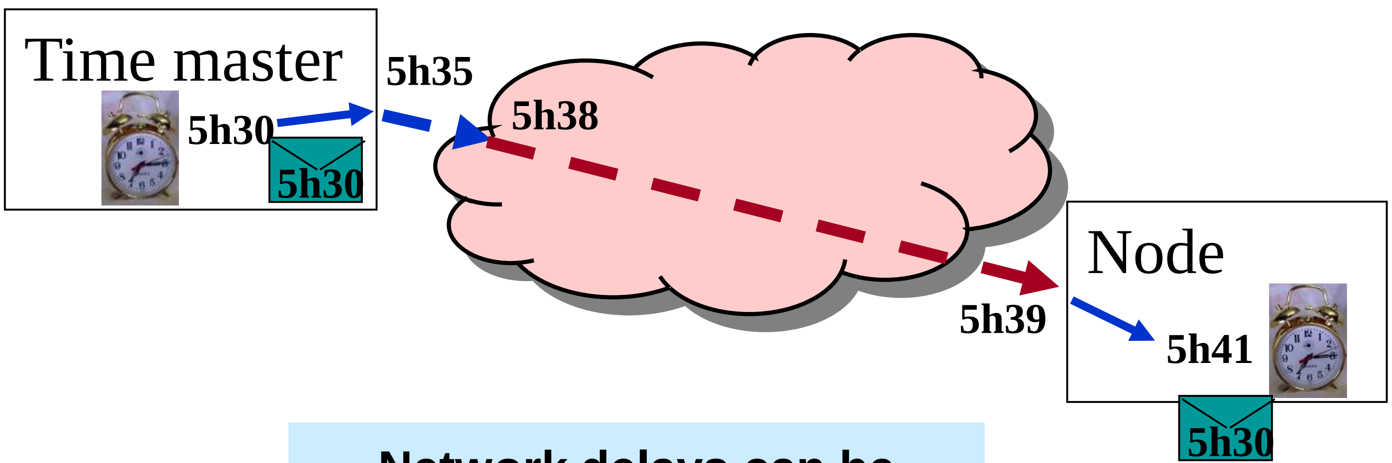
Synchronizing clocks

- Requires **regularly**
  - **Exchanging** clock values
  - Measuring **differences**
  - Computing and applying a **correction term**
    - In the form of **increment/decrement** of the **microticks** counter (n)
    - (not so common) Directly in the clock tick value



Network delay and precision

- Impact of **network delay jitter** on the **achievable precision**



Network delays can be rather variable and large

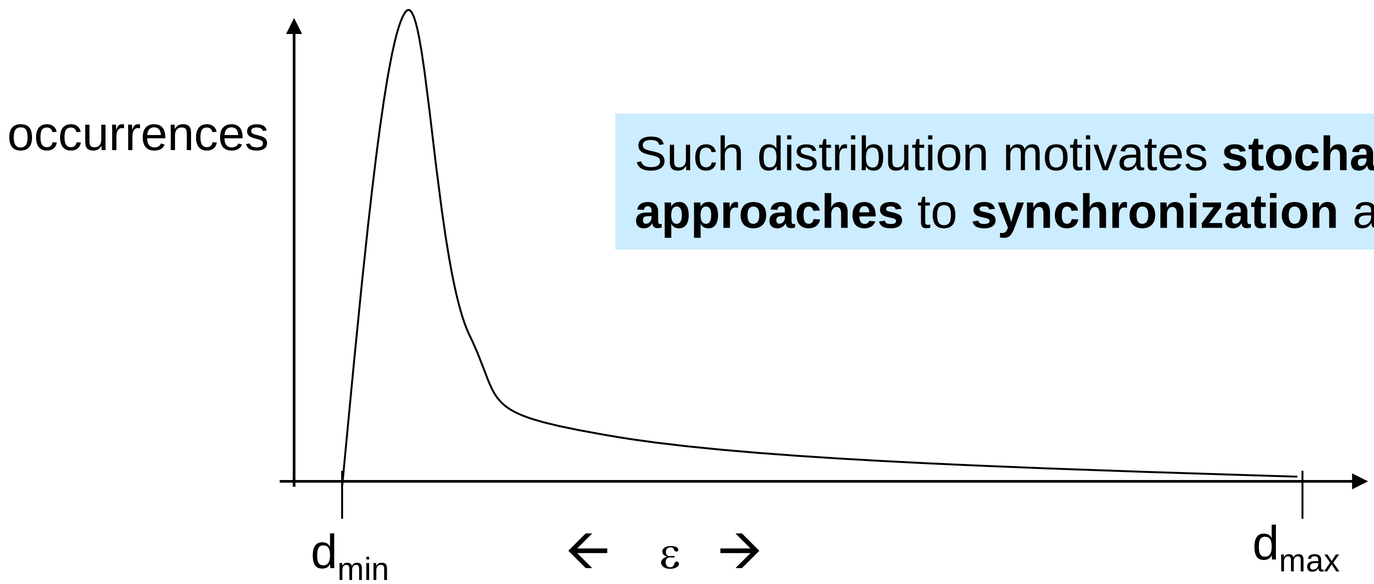
Network delay and precision

- **Network delay jitter** (ε) limits the achievable **precision** (δ)
  - **Clock synchronization** algorithms allow correcting **systematic errors**
    - not the impact of ε (**$\varepsilon = d_{max} - d_{min}$** → jitter in the network delay d)
  - Typical precision with SW methods in small networks is worse than 10μs
    - In LANs it is common to achieve 1-5ms precision
  - With special HW support, it is possible to reach 1μs or better
  - **Ludelius and Lynch** showed that the **precision** δ achievable in a network with **N nodes** (with **drift-free** oscillators) and **ε network delay jitter** is never better (smaller) than

$$\delta \geq \varepsilon \left(1 - \frac{1}{N}\right)$$

Network delay and precision

- In **shared networks** with collisions and back-off/retry mechanisms
  - Typical distribution of the network-induced delay (d)

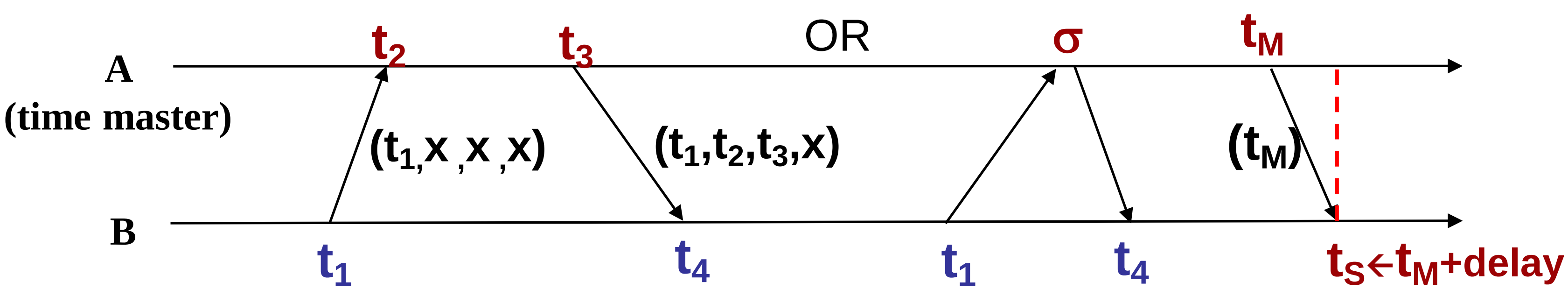


Such distribution motivates **stochastic approaches** to **synchronization** and **precision**

  - **Ludelius and Lynch**'s bound is a deterministic bound
  - **Stochastic approaches** can achieve **unbounded precision**

Measuring the network delay

- **Round-trip delay - RTD**
  - estimated from $((t_4 - t_1) - (t_3 - t_2))/2$ on node B
  - used to correct delay at B upon reception of time marks from A

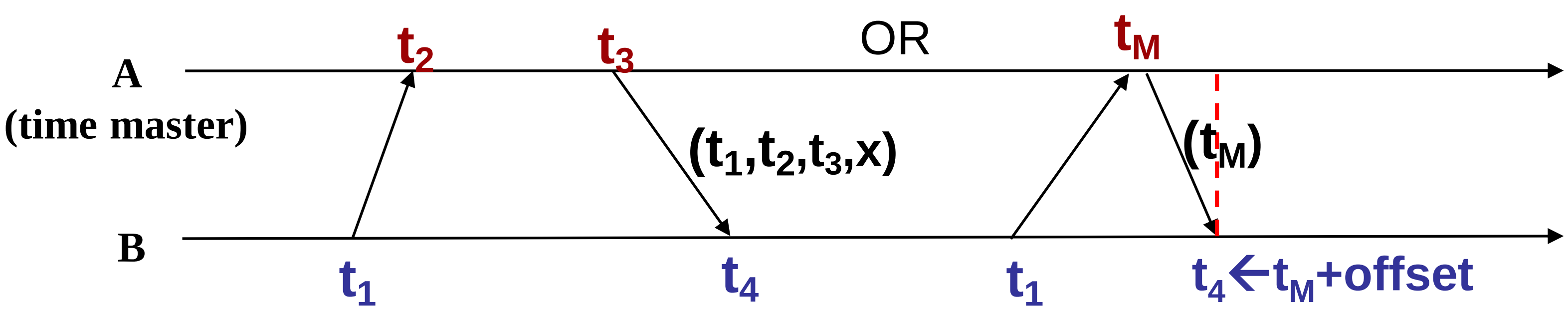


$$delay \approx \frac{RTD}{2} = \left\langle \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \middle| \frac{(t_4 - t_1) - \sigma}{2} \right\rangle$$

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Measuring the clocks offset

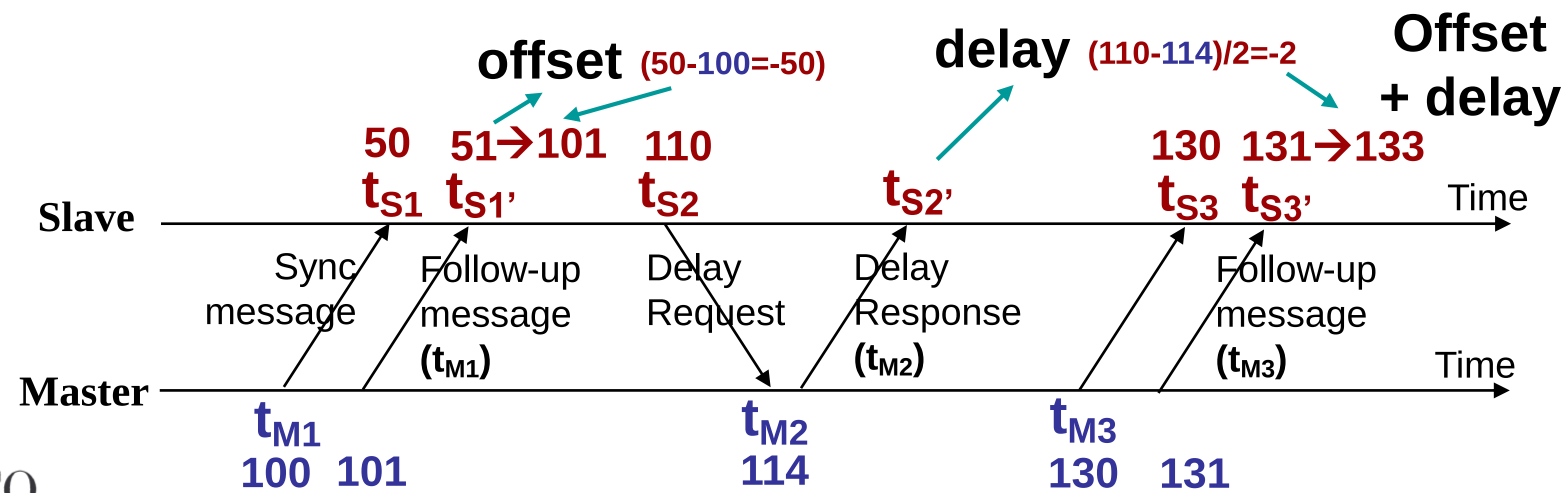
- **Offset – Compensating the clocks average difference**
  - estimated from $((t_2+t_3)-(t_4+t_1))/2$ on node B
  - used to correct offset at B upon reception of time marks from A



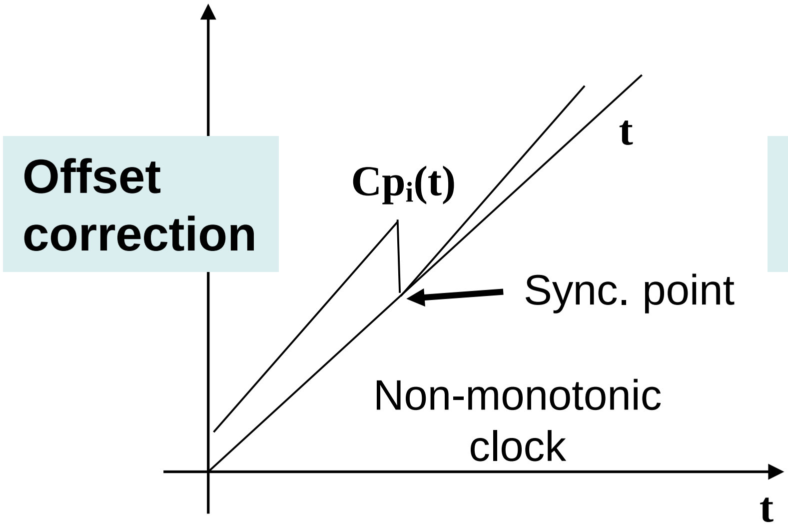
$$offset \approx \left\langle \frac{(t_2+t_3)-(t_4+t_1)}{2} \middle| t_M - \frac{(t_4+t_1)}{2} \right\rangle$$

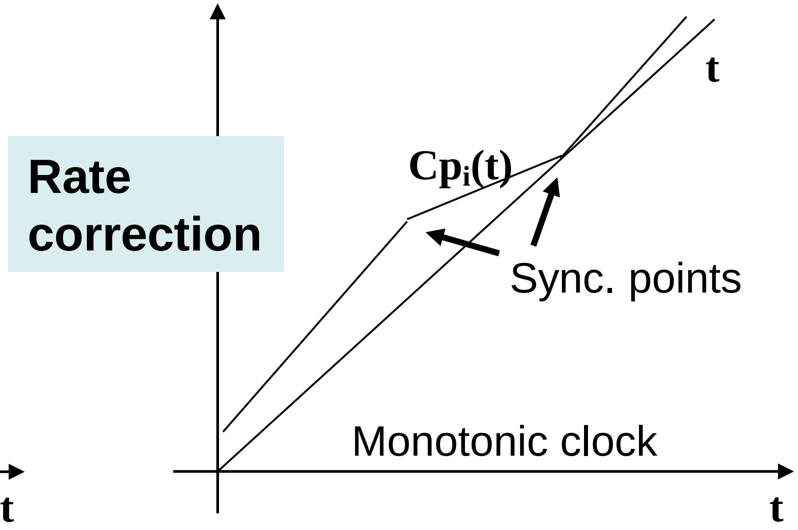
Synchronization with IEEE 1588

- **Follow up messages**
  - ***Timestamps on "end of transmission"***
  - ***Synchronization messages do not carry timestamps***
- **Slave to Master offset:** estimated from $(t_{S1}-t_{M1})$ at $t_{S1'}$
  corrected at $t_{S1'} \leftarrow t_{S1'}-(t_{S1}-t_{M1})$
- **Network induced delay:** estimated from $(t_{S2}-t_{M2})/2$ at $t_{S2'}$
  corrected at $t_{S3'} \leftarrow t_{S3'}-(t_{S3}-t_{M3})-\text{delay}$



Clock correction





- **Most applications** require **monotonic clocks**

$t_1 < t_2 \Rightarrow Cp_i(t_1) < Cp_i(t_2) \quad \forall_{i,\, t_1,\, t_2}$ **chronoscopic behavior**

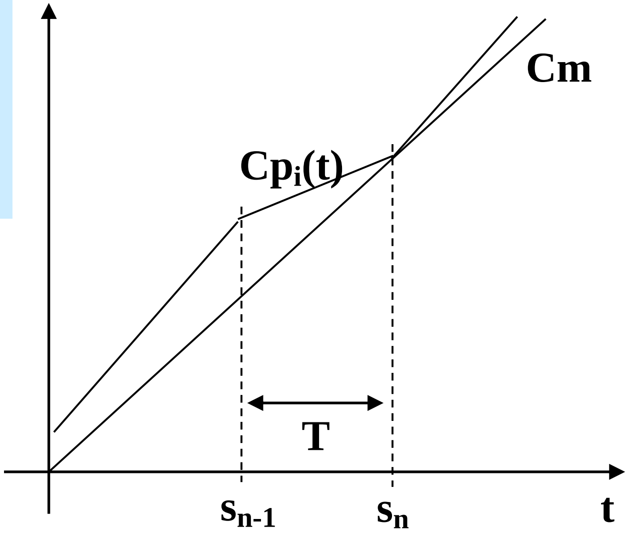
- A **chronoscopic behavior** implies **rate correction**

Correcting the local clock

- **Rate correction with feedback** (*servo-clock*)
  - **Cp_i** = **local clock** (as seen by the local applications)
  - **Cm** = **master clock** (as received in master messages)
    - $\rho_n = 1 - (\mathbf{Cp_i}(s_n) - (\mathbf{Cm}(s_n)+d)) / T$
      - ρ_n is the microticks (rate) correction term
      - T = sync interval, s_n = synchronization point n

– $\mathbf{Cp_i}(t) = \mathbf{Cp_i}(s_n) + \rho_n*(t-s_n)$ $\quad \mathbf{s_n < t < s_{n+1}}$

$$\delta = \xi + \varepsilon$$

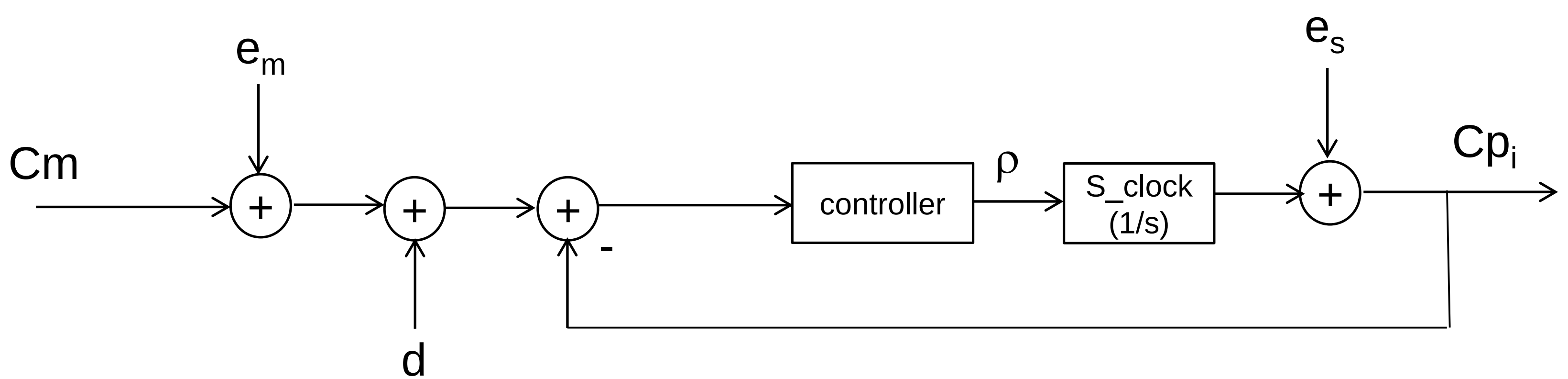


<u>It is common to consider:</u>

- $\rho_{min} < \rho_n < \rho_{max}$ to bound the clock growth
- **k*T** (k>1) instead of **T** to stabilize the local clock (less reactive)

Correcting the local clock

- **Rate correction with feedback** (*servo-clock*)
  - The clock correction as a **feedback control loop**
    - e_m → delays affecting the time stamping in the master
    - e_s → errors affecting the oscillator in the slave
    - d → network delay



Distributed clock synchronization

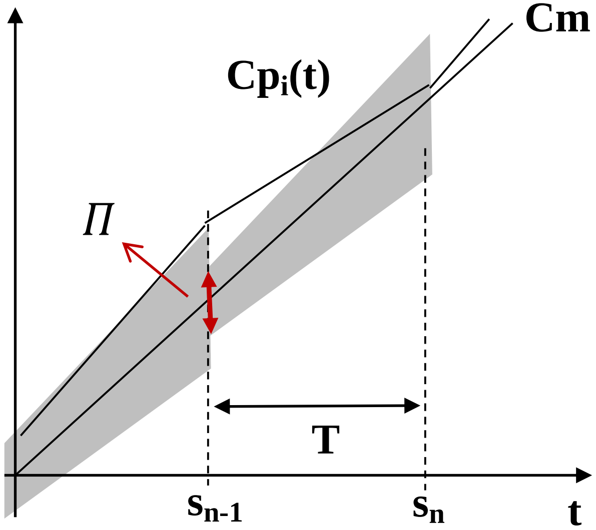
- There is **no master clock**
  - All nodes exchange their clock values among themselves
  - A **virtual reference clock Cm_i** is built averaging **all N clocks**

$$Cm_i(s_n) = \frac{1}{N}\sum_{k=1}^{N} Cp_k(s_n)$$ → error divided by N (convergence factor Π)

  - Under certain conditions, all nodes converge to this virtual ref

$$\delta_{dist} \geq \Pi + \xi + \varepsilon \quad \text{and} \quad \Pi = \frac{\delta_{dist}}{N}$$

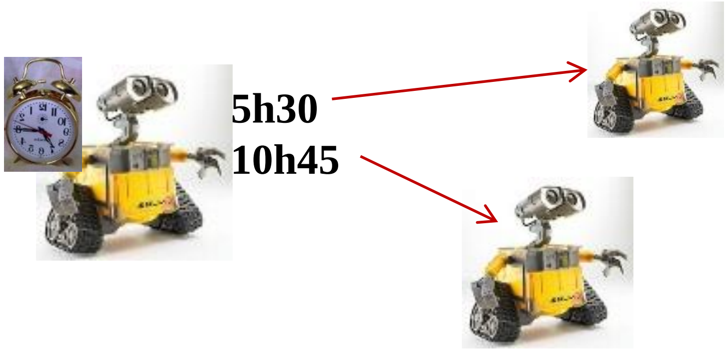
$$\delta_{dist} \geq (\xi + \varepsilon)\frac{N}{N-1}$$



  - Crashed nodes can be removed from the group and do not affect the global clock

Distributed clock synchronization

- **Fault-Tolerant Average (FTA)**
  - The **usual average** is **sensitive to very poor clocks** that diverge a lot from the others, in the presence of network errors
    - Good clocks can end up with less weight than poor clocks
    - In sparsely connected networks, normal average can lead to different nodes computing different virtual references
  - The **usual average** is also **sensitive to Byzantine** errors
    - Nodes that send inconsistent information to different destinations
    - In this case, the whole set of nodes will not converge to a virtual reference



Distributed clock synchronization

- **Fault-Tolerant Average** (FTA)
  - **sort differences** between local clocks and average
  - **eliminate** the clocks with the **k highest** and **k lowest** differences to the average
  - Use the average of the remaining clocks as the **virtual reference**
  - This allows **tolerating k Byzantine** clocks

$$\Pi = k \frac{\delta_{dist}}{N - 2k}$$

$$\delta_{dist} \geq (\xi + \varepsilon) \frac{N - 2k}{N - 3k}$$

~~Node i – 15h30~~
Node j – 15h40
Node k – 15h41
Node l – 15h41
~~Node m – 15h45~~

FTA
ref clock

Distributed clock synchronization

- **Interactive Consistency**
  - **All nodes:**
    - Send a **vector** with **their view** of all other clocks
    - Build a **local matrix** with **all views of all clocks**
  - Allows immediate detection of Byzantine clocks
    - Can be readily excluded
    - Remaining ones averaged to generate the virtual reference
  - **Tolerant to any Byzantine clocks**
  - Requires more communication

Clock of node 4 received by others

	0	1	2	3	4	5
0	21	20	21	22	19	20
1	31	30	29	30	29	30
2	41	41	40	39	**60**	39
3	50	51	50	50	51	49
4	61	60	59	60	**24**	60
5	71	70	69	70	71	70

Clocks received by node 4 in one round