

---

# Global Time and Clock Synchronization in Networks



## "Milestone" papers concerning clock synchronization:

---

L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):5278, July 1985

F. Cristian, H. Aghili, and R. Strong: Clock synchronization in the presence of omission and performance failures, and processor joins. In *Proc. of 16th International Symposium on Fault-Tolerant Computing Systems*, July 1986

H. Kopetz and W. Ochsenreiter: Clock synchronization in distributed real-time computer systems. *IEEE Transactions on Computers*, C-36(8):933940, August 1987.

D. L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Computers*, 39(10):14821493, October 1991

P. Verissimo, A. Casimiro, and L. Rodrigues. Cesiumspray: a precise and accurate global time service for large scale systems. *Journal of Real-Time Systems*, 12:243294, 1997

Good survey paper:

Emmanuelle Anceaume and Isabelle Puaut: Performance Evaluation of Clock Synchronization Algorithms, *Rapport de recherche N ° 3526*, Octobre 1998  
ISSN 0249-6399



# What is the function of time ?

## **Orientation:**

**Determine the position of an event in the continuum of time.**

## **Regulation**

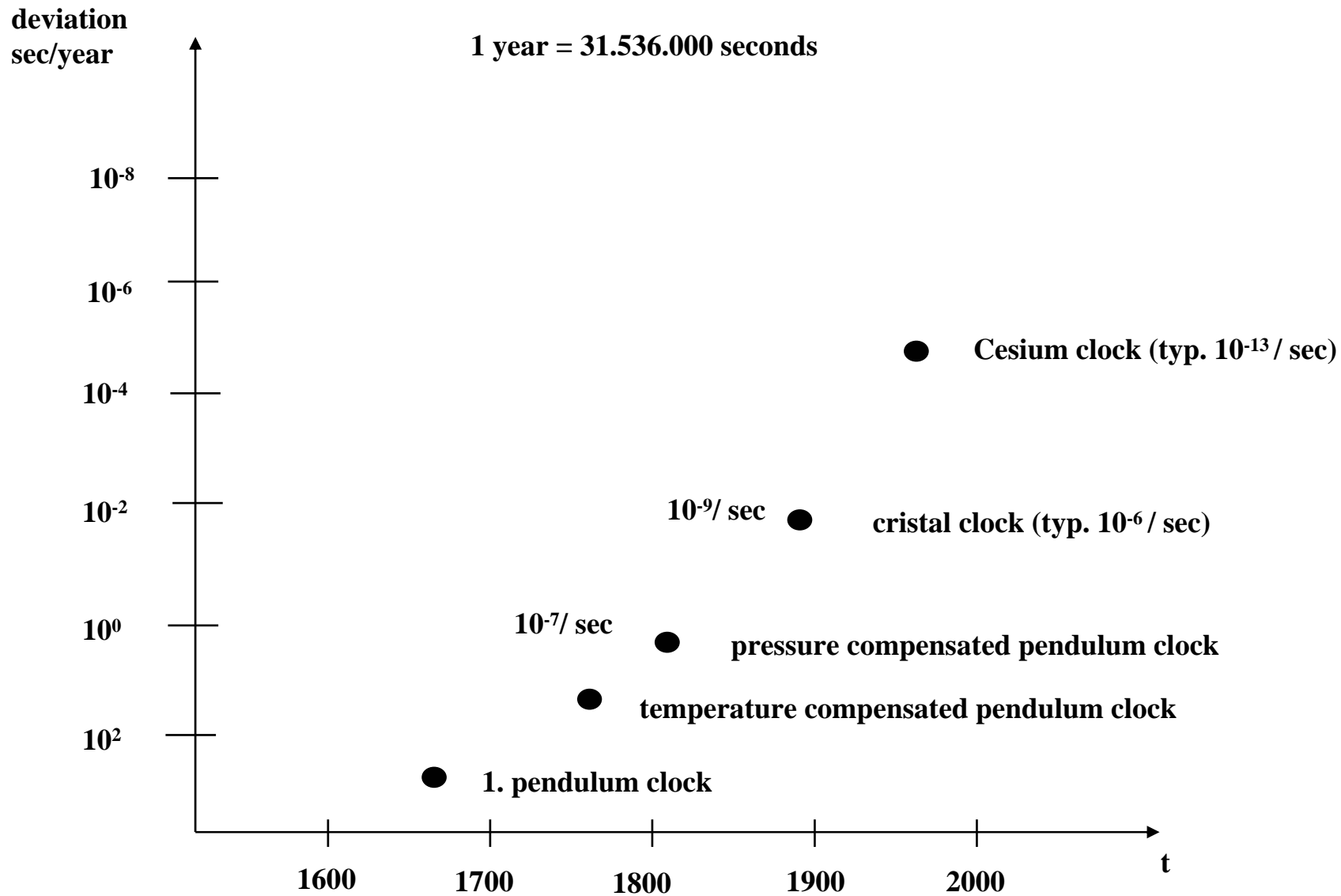
**Enforcing a time regime.**

## **Coordination**

**Synchronizing cooperative tasks.**



# Advances in Time Measurement





## The Cesium-clock of the "Physikalisch-Technischen Bundesanstalt" at Braunschweig

D. Lehmann: "Ohne Uhren keine Zeit", in: Geo - Das neue Gesicht der Erde,  
Nr.12, Dezember 1995



# Time standards

**Astronomische Zeit (AT)**

**Physikalische Zeit (PT)**

basiert auf der gleichförmigen Bewegung von Himmelskörpern

basiert auf periodischen physikalischen Prozessen (Schwingungen)

**Sonnendurchlauf:**

Höchster Punkt der Sonne während eines Tages

**Sonntag:**

Intervall zwischen zwei aufeinanderfolgenden Sonnendurchläufen

**Sonnensekunde:**

Der 1 / 86400 Teil eines Sonntages

**Mittlere Sonnensekunde:**

Sonnensekunde gemittelt über eine große Anzahl von Sonntagen

**Zeitzone:**

Erster Zeitstandard (1820) basiert auf mittlerer Sonnensekunde

Gebiete für die dieselbe Zeit festgelegt ist. 1884 wird die Welt in 24 Zeitzone aufgeteilt. Die Zeitzone unterscheiden sich von UT (GMT) ganzzahlig um jeweils 1 Stunde.

**UT (AT, 1833)**

Universal Time (UT) Mittlere Sonnenzeit, gemessen am Greenwich 0-Meridian (GMT). Basiert auf der mittleren Länge eines Sonntags, d.h. auf der Erdrotation

**ET (AT, 1955)**

Ephemeridenzeit (ET), basiert auf der Umlaufzeit der Erde um die Sonne. Harold Spencer Jones stellte 1939 fest, daß die Rotation der Erde variiert, die Umlaufzeit um die Sonne nicht. 1 Sekunde der ET wird festgelegt als der 1/31.566.925,9747 Teil des tropische Jahres, das am Mittag des 1. Januars 1900 begann. (Tropische Jahr: Periode zwischen zwei aufeinanderfolgenden Durchläufen der Sonne durch den Himmelsäquator in derselben Richtung.)

**UT0 (AT, 1960)**

Zeit, basierend auf den lokalen Beobachtungen verschiedener, über die Erde verteilter Observatorien.

**UT1 (AT, 1960)**

Zeit, basierend auf der Koordination der verschiedenen UT0-Zeiten (Mittlung).

**UT2 (AT, 1960)**

Nochmals, auf empirischer Basis korrigierte UT1-Zeit.

**TAI (PT, 1961)**

Temps Atomique International (TAI) basiert auf mehreren koordinierten Cäsium-Uhren. Fortlaufende Zeitählung, beginnend mit dem 1. Januar 1958 0 Uhr UT2-Zeit (daher konsistent mit UT2).

**UTC (PT, 1972)**

Universal Time Coordinated (UTC) basiert auf TAI, wird aber ständig an UT2 angepaßt. Immer wenn UTC und UT2 mehr als 800 ms auseinander gedrifted sind, wird eine "Schaltsekunde" eingefügt. UTC beginnt am 1. Januar 1972. Seit dieser Zeit sind (bis 1992) 15 Schaltsekunden eingefügt worden. UTC ist damit eine an AT angepaßte physikalische Zeit. Genauigkeit: ca. 1 Sek / 300000 Jahre



## **World wide, standardized time reference : *Universal Time Coordinated (UTC)***

---

**distributed by:**

- **land-based radio stations (0,1-10 msek)**
- **GPS - *Global Positioning System***
- **GEOS - *Geostationary Environmental Operational Satellites***

**Time base in distributed systems:**

- **a (single) UTC-based reference clock**
- **local clocks**
- **algorithms for clock synchronization**

**Correspondence to perfect time is dependent on:**

- **precision of the time signal**
- **distance to sender**
- **atmospheric conditions**
- **drift of local clocks**
- **predictability of network latencies (steadyness and tightness)**
- **the synchronization algorithm**



# Physical model of time

---

**Assumption: Existence of a uniform sequence of events suitable to represent a time reference**

**Properties:**

**Equivalence:** All time stamps are equivalent in the sense that there is a linear formule converting a timestamp into another one.

**Linearity:** The conversion formulas are linear.  
Different timestamps can be related by a proportional factor.

**Causality:  
(potential)** Time is aligned according to an order of events which are related in a before/after relationship.

**Reversal:** Physical processes in general cannot be reversed.

**Homogeneity:** Time flow is uniform and does not contain holes.





# Relation between internal and external time

---

To measure the occurrence of events in a system in the metric of the external physical time (e.g. TAI), there must be an internal representation of the physical time.

**Internal physical time:** Internal time reference that approximates the external physical time. All events are ordered in the global flow of time. The before/after relation between events is determined according to the internal time. Causally independent events are also temporally ordered. Given two events  $e_1$  and  $e_2$  at time  $t_{e1}$  and  $t_{e2}$  respectively. If on an internal time scale  $t_{e1} \leq t_{e2}$  holds, this must also hold on an external time scale.

**Internal synchronization:** Synchronization of internal clocks .

**External Synchronization:** Synchronization to the external physical time.



# Physical Clocks

---

**Physical clock:** Time reference that is based on periodic physical processes like oscillations of a pendulum, a crystal or an atom to measure the progression of time.

**Characterization of physical clocks:**

**Frequency/Rate:  $f$**  Number of periodic events per time frame. (oscillations, clock pulses)  
Metric: Hz, KHz, MHz, GHz, . . .

**Period  $p=1/f$**  Time interval between two consecutive periodic events.  
Metric: sec, ms,  $\mu$ s, ns . .

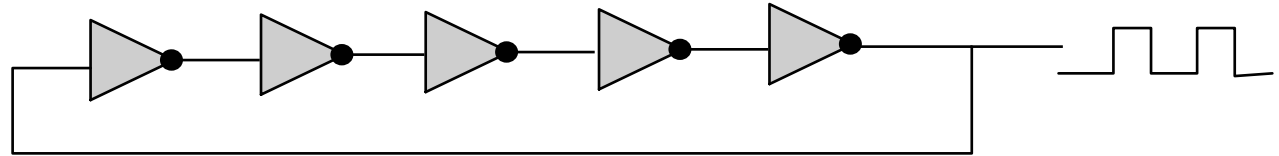
**Granularity:  $g$**  time interval between two clock pulses (corresponds to a period).  
Can only be determined by a clock of higher granularity. The granularity determines the lower bound of the temporal distance between two events that are not recognized as simultaneous events.

Metric: sec, ms,  $\mu$ s, ns . . .

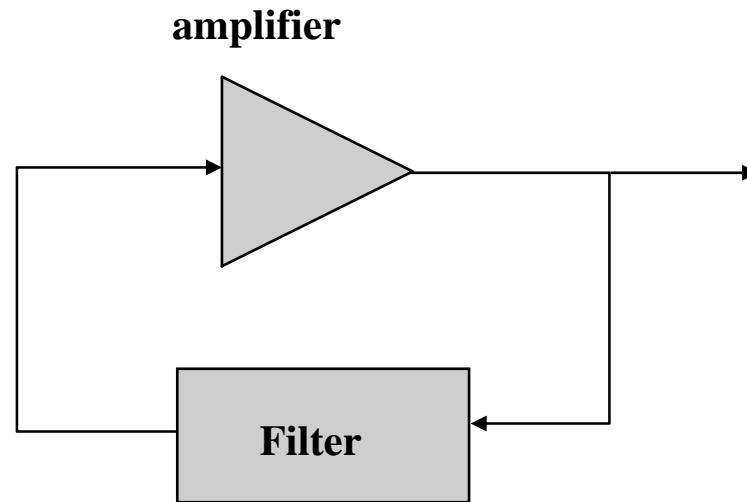


# Physical Clocks

Simple oscillator



Frequency controlled oscillator



alternatives, e.g.:



# Terminology

---

<b>stability</b>	:	<b>deviation of the specified frequency</b>
<b>accuracy <math>\alpha</math></b>	:	<b>deviation wrt. a standard time</b>
<b>granularity (resolution) <math>g</math></b>	:	<b>smallest measurable time difference</b>
<b>precision <math>\pi</math></b>	:	<b>long term difference between two clocks</b>
<b>convergence</b>	:	<b>difference of two clocks immediately after a synchronization</b>
<b>offset <math>\delta</math></b>	:	<b>time difference between two clocks</b>
<b>skew</b>	:	<b>frequency difference between two (clock) oscillators</b>
<b>drift <math>\rho</math></b>	:	<b>slowly increasing difference to a standard time</b>



# Terminology

---

Let  $C(t)$  be the function that maps the time instance  $t$  of the reference time to the internal clock.

**Def. 1:**

a clock is a reference clock or perfect clock if :  $\forall t: C(t) = t$

**Def. 2:**

a clock is correct at time  $t_0$  , if  $C(t_0) = t_0$  is satisfied.

**Def. 3:**

a clock rate is perfect at  $t_0$  , if at  $t_0$  :  $C'(t) = 1$  (  $dC(t) / dt = 1$  )

$C'(t)$  denotes the (rate) stability of a clock.

We consider 3 cases:

$dC/dt > 1$ : *fast clock*

$dC/dt = 1$ : *perfect clock*,

$dC/dt < 1$ : *slow clock*

**Drift(rate)  $r$  :** deviation of the clock/time

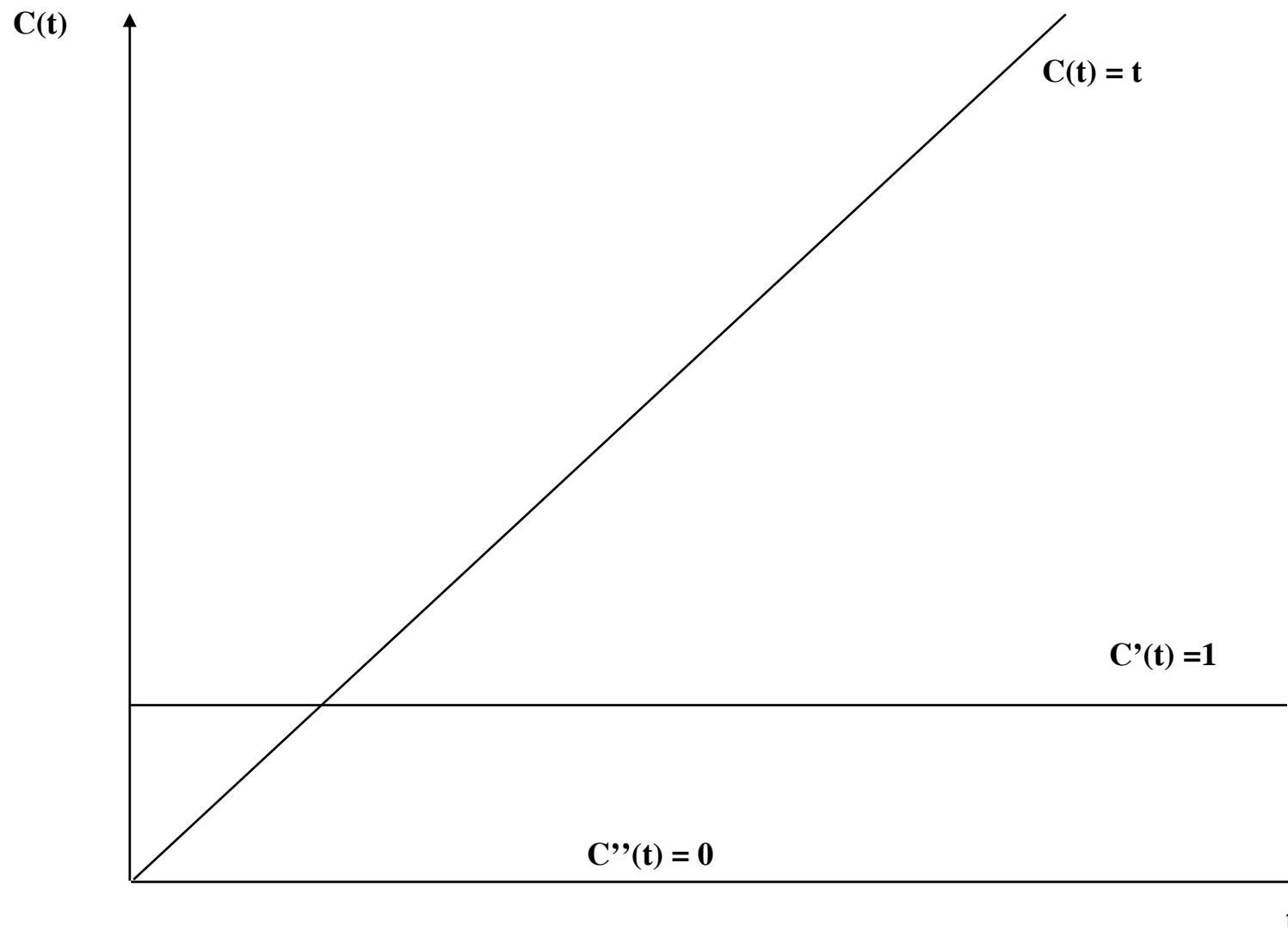
The drift is assumed to be bounded by:

$$1-\rho \leq dC/dt \leq 1+\rho$$

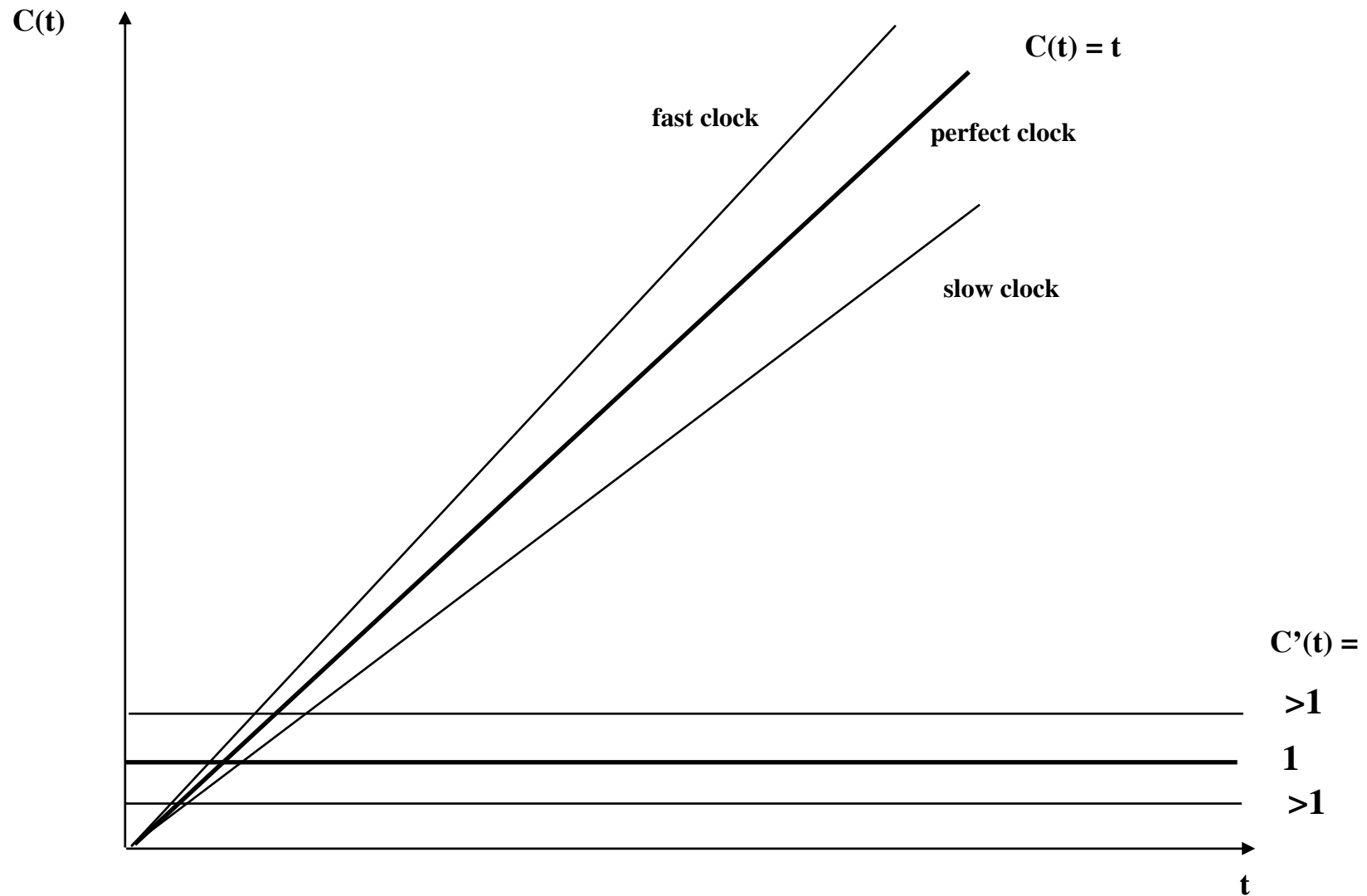


# The perfect Clock

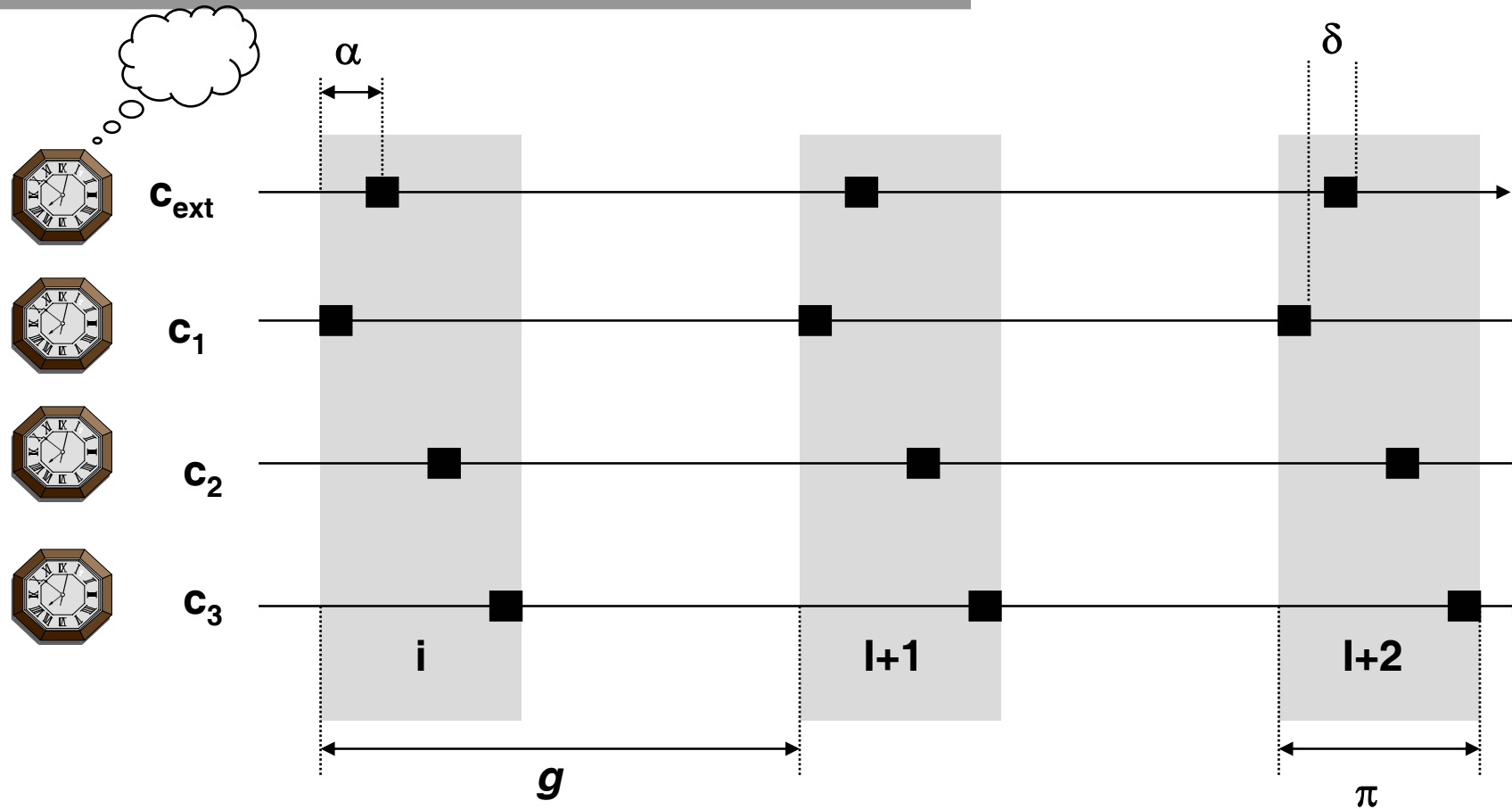
---



# Clock with constant Drift Rate



# Relation between the main distributed clock parameters

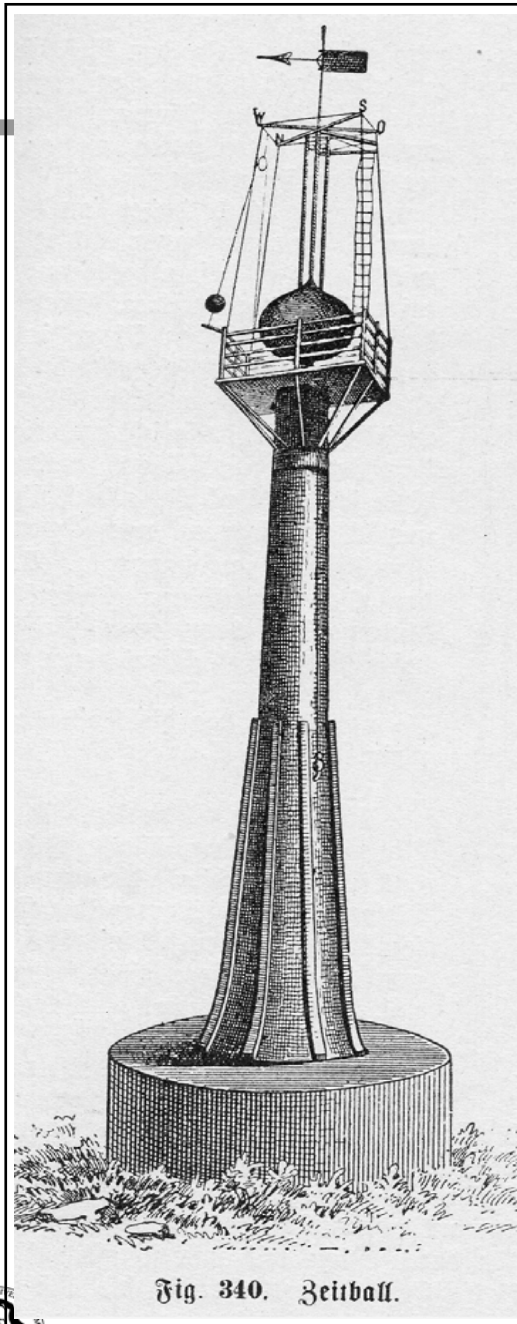


$\alpha$  : accuracy  
 $\pi$  : precision  
 $g$  : granularity  
 $\delta$  : offset

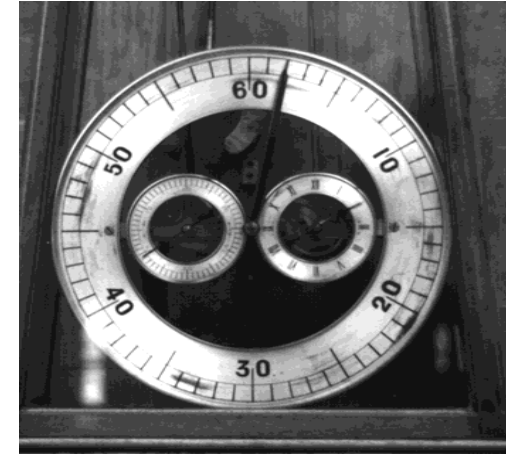
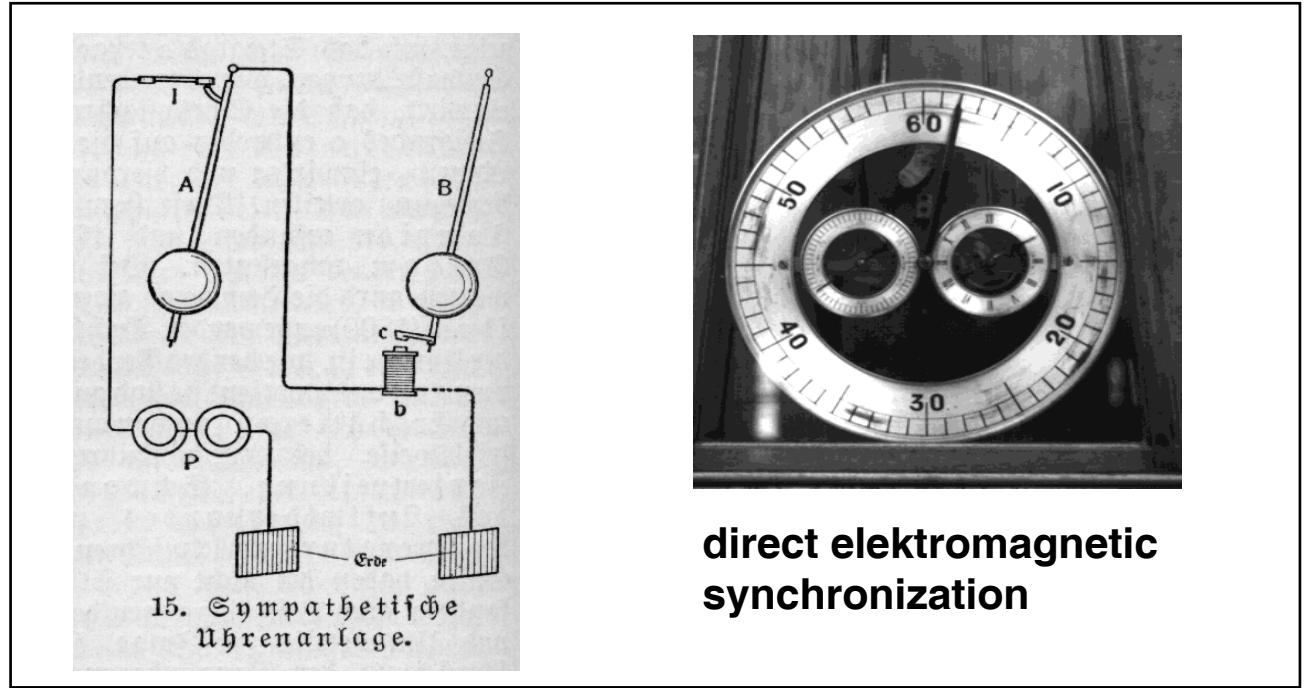




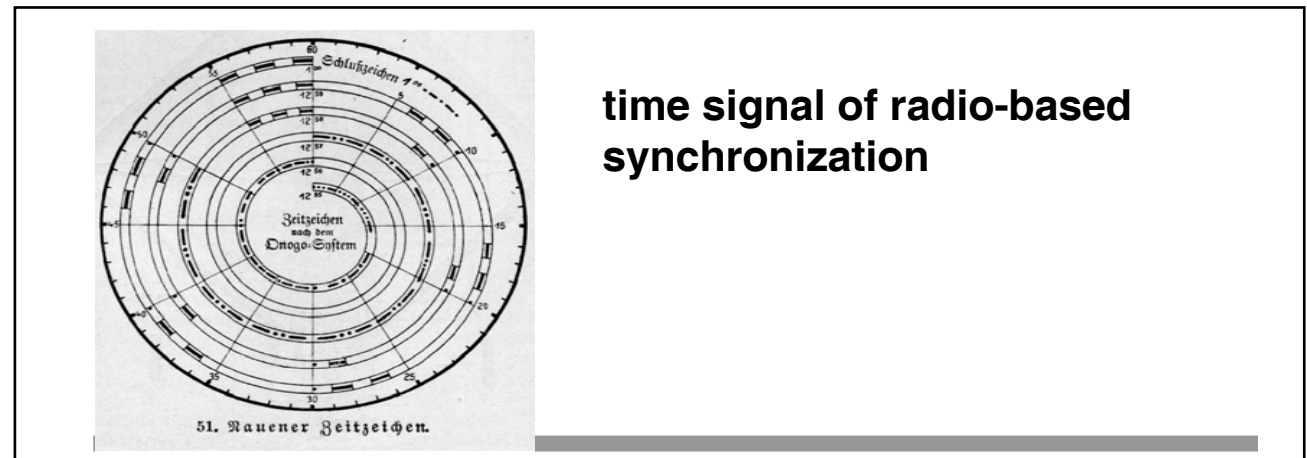
# Clock Synchronization



Embedded Networks 07



direct elektromagnetic synchronization



time signal of radio-based synchronization



# Synchrony between Clocks

Two clocks  $c_i$  and  $c_j$  are synchronous at time  $T$ , if:

$$|c_i(T) - c_j(T)| < \delta$$

**measure**

**objective**

**frequency synchronization**

**stability/min. skew**

**time synchronization**

**accuracy / min. offset**



# When to re-synchronize ?

---

Maximum drift rate  $\rho$ :

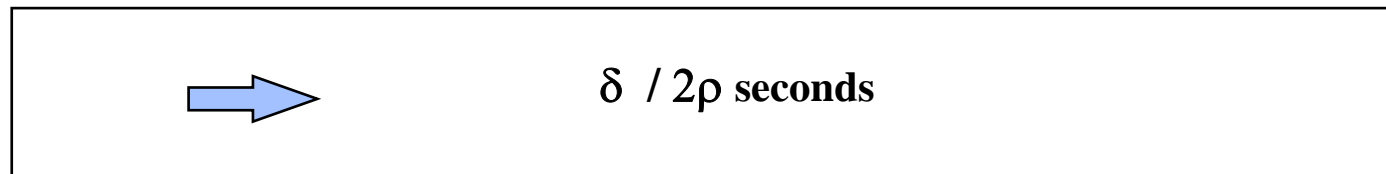
$$1-\rho \leq dC/dt \leq 1+\rho$$

constant  $\rho$  is a part of the physical clock specification; typical:  $10^{-5}$  to  $10^{-6}$

difference of two un-synchronized clocks  $\Delta t$ :

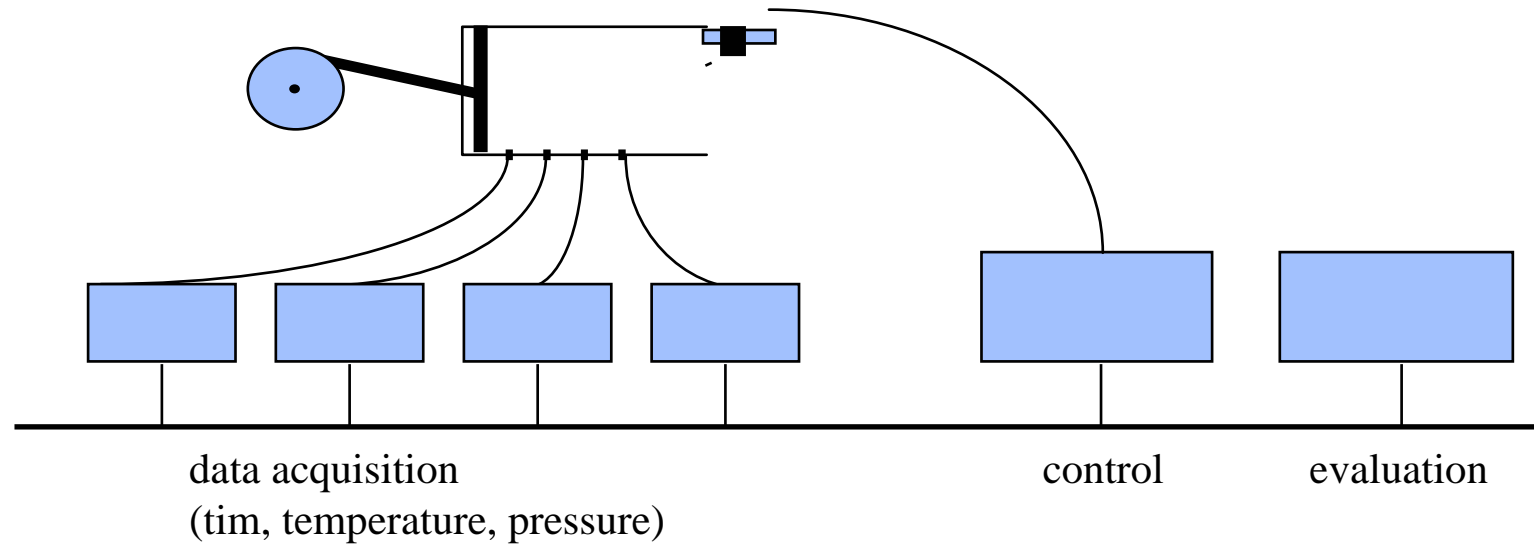
$$\leq 2\rho\Delta t$$

re-synchronization interval if two clocks should not deviate more than  $\delta$  seconds after



# When to re-synchronize ?

typical drift rate:  $10^{-5}$  -  $10^{-6}$  (~1sek in 1 - 11 days).



10 Mhz clock; Period = 100ns

after 10 sec., the worst-case difference between clocks can be calculated as:

$$10 * 2 * 10^{-6} = 2 * 10^{-5} = 20 \mu\text{sec} = 20.000 \text{ nsec}$$



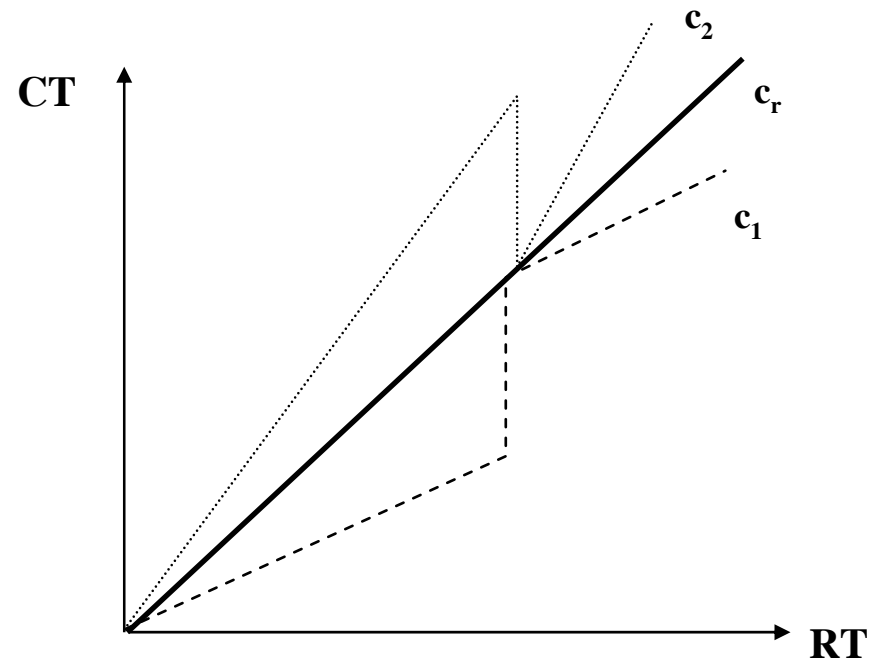
200 times the clock resolution!!



# How to synchronize ?

---

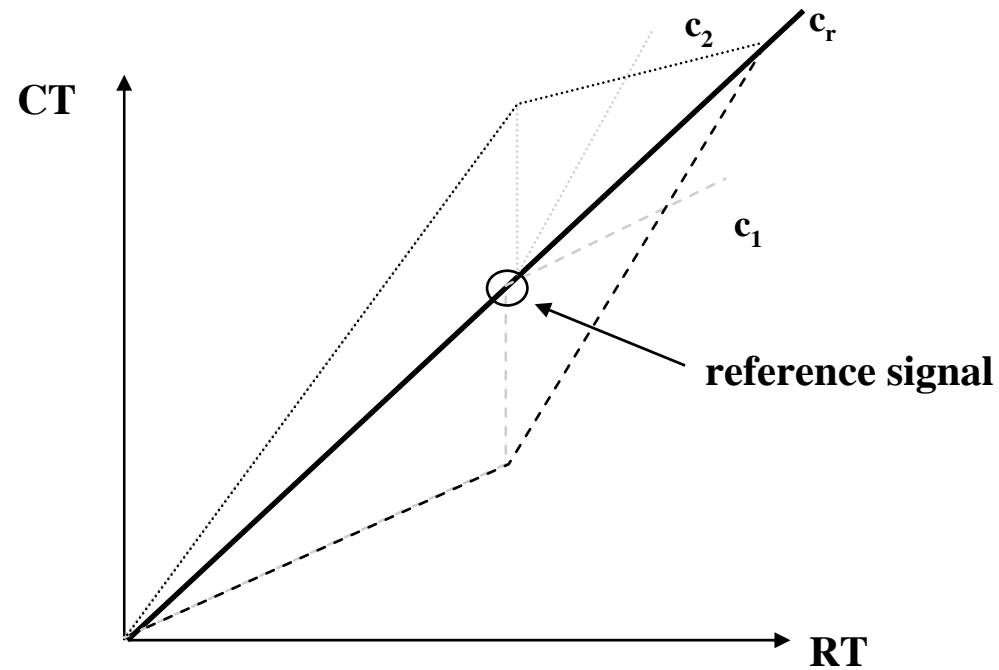
## Adapting clock values



# How to synchronize ?

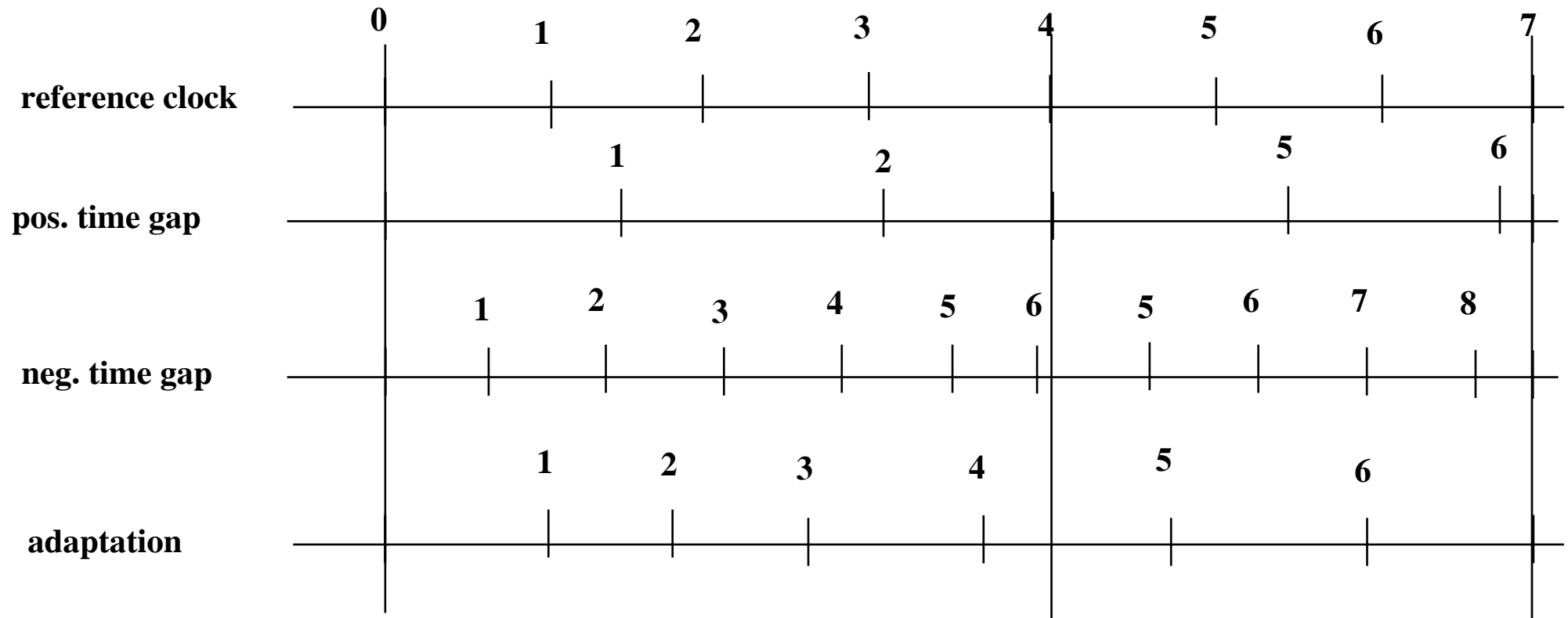
---

## Adapting clock rates

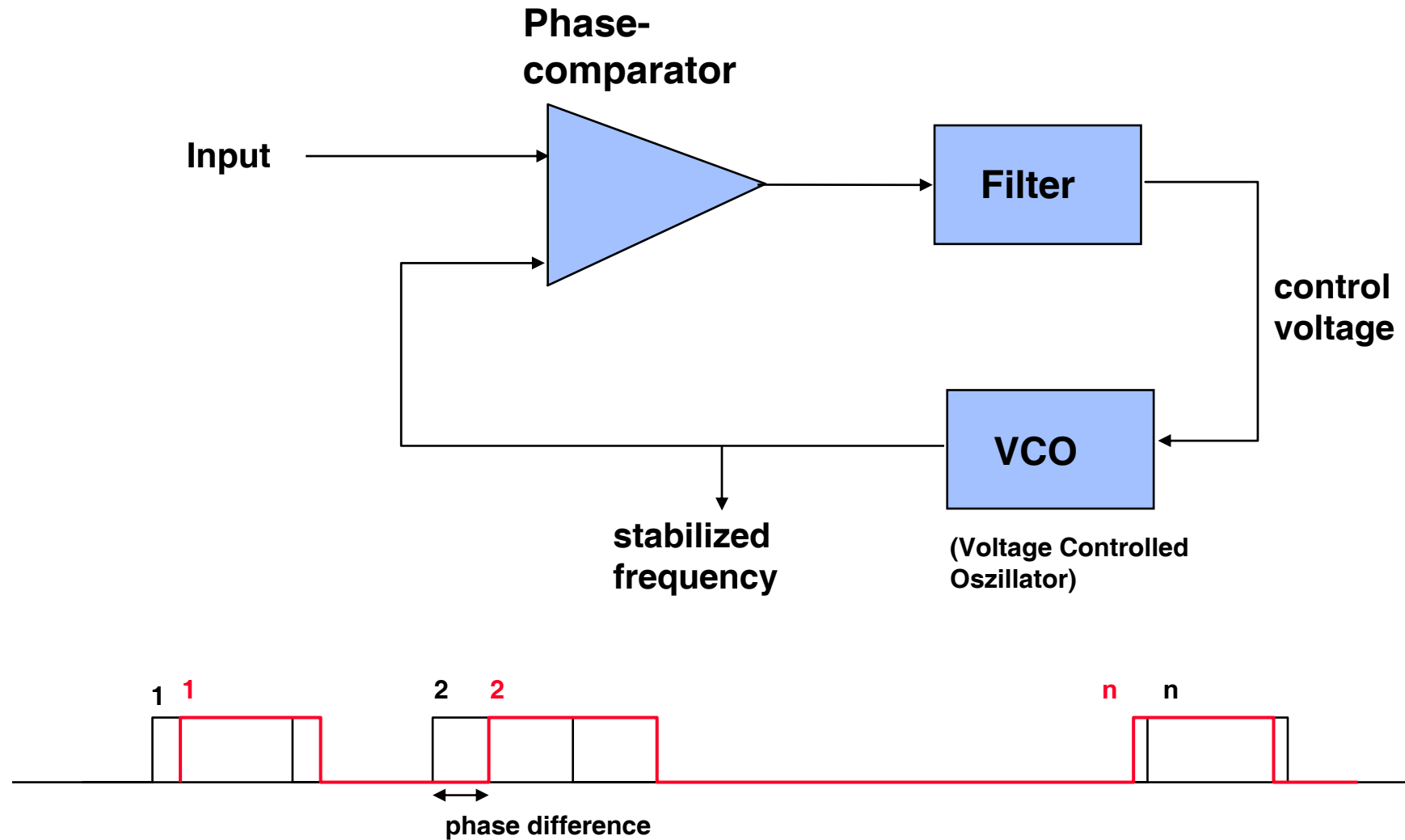


# How to synchronize ?

- counter correction
- drift correction



# Adapting the clock rate: PLL (Phase Locked Loop)





---

# Clock synchronization via messages in communication networks

- **time server**
- **Master- Slave synchronisation**
- **Decentralized and co-operative sync.**



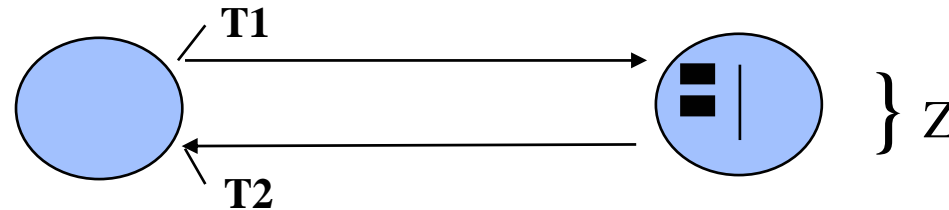
# Sever-based clock synchronization

---

## Cristian's Algorithm:

Every node synchronizes its time base periodically with a time server.

period:  $\frac{\delta}{2\rho}$  to keep error below  $\delta$



- (a) send request for a time stamp  $Z$  to the time server
- (b) calculate a correction factor  $K$  to correct the local clock

Problems:

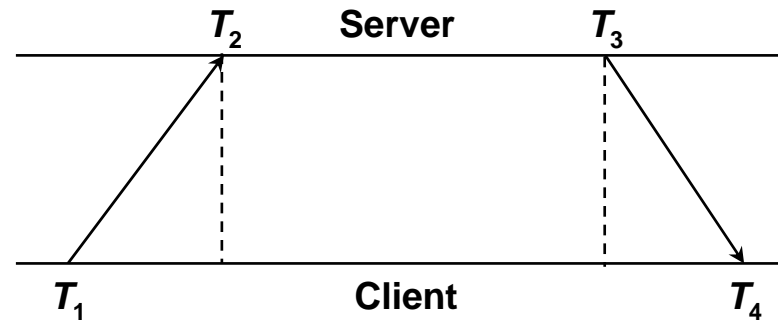
- Monotony properties of timee
- Measurements are based on roundtrips ( $T2-T1$ ). A roundtrip may vary depending on network condidtions.
  - > simple correction:  $[(T2-T1)/2 - Z]$
  - > Refinement 1: statistical methods to calculate the average of  $T2-T1$ .
  - > Refinement 2: statistical methods to estimate the delay in the server.



# Calculating offset and delay

---

Offsets  $o$   
Delay  $d$



$$o = [(T_2 - T_1) + (T_3 - T_4)] / 2$$

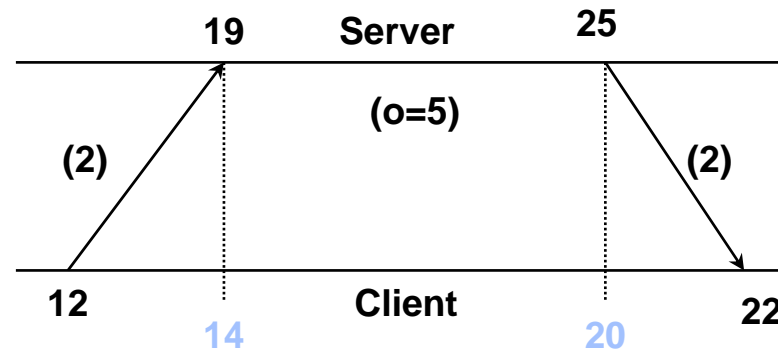
$$d = (T_4 - T_1) - (T_3 - T_2)$$



# Calculating offset and delay

---

Offsets  $o$   
Delay  $d$



$$o = [(19 - 12) + (25 - 22)] / 2$$

$$o = 5$$

$$d = (22 - 12) - (25 - 19)$$

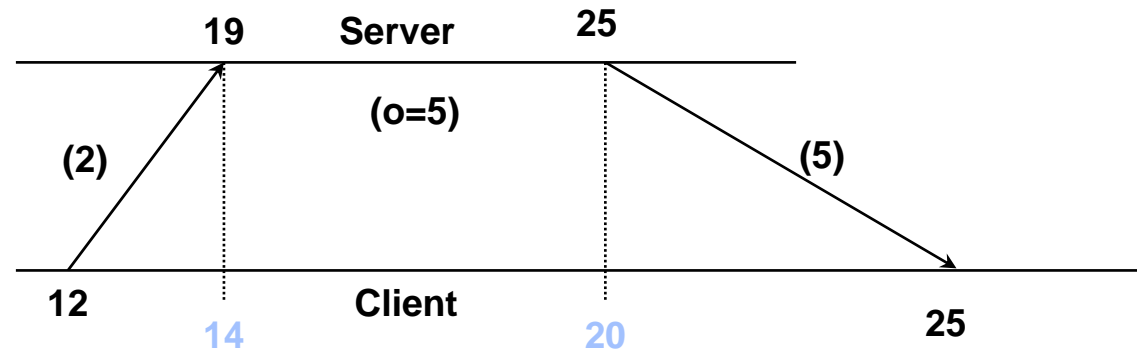
$$d = 4$$



# Calculating offset and delay

---

Offsets  $o$   
Delay  $d$



$$o = [(19 - 12) + (25 - 25)] / 2$$
$$o = 3,5$$

$$d = (25 - 12) - (25 - 19)$$
$$d = 7$$



# Parameter for the calculation of time

---

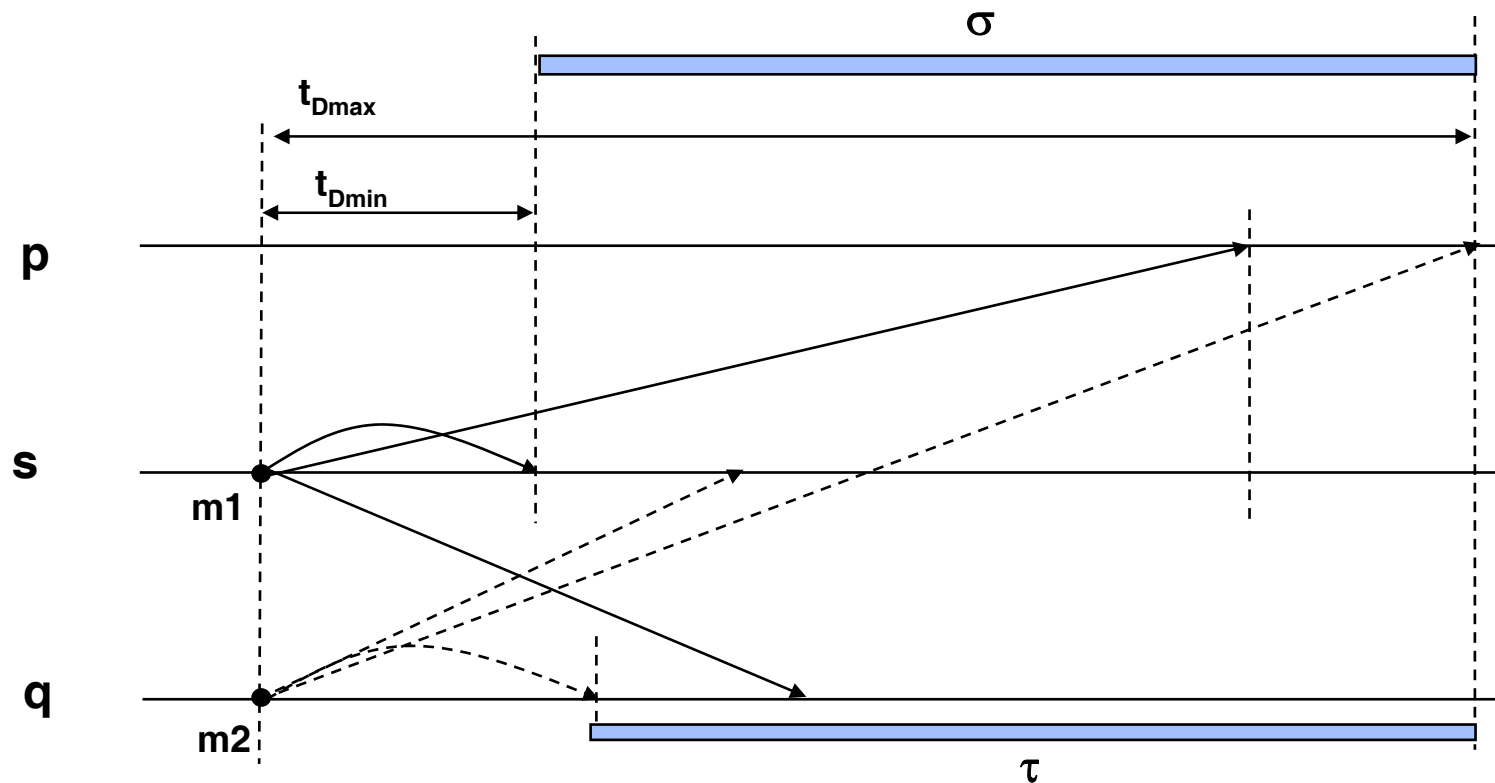
- ➔ **message delays in the communication network  
(steadyness + tightness)**
- ➔ **Time until the message is sent  
Delays in the Reply Queue**
- ➔ **Delay when reading the local clock  
Delay in the Request Queue**
- ➔ **Physical parameters of the server clock:  
Drift, Offset**



# to remember:

Steadyness  $\sigma$  : measures the maximal difference of delivery times of **DIFFERENT** messages.

Tightness  $\tau$  : measures the difference of transmission times of **ONE** message to **DIFFERENT** nodes.



# The *Network Time Protocol* (Internet NTP)

---

## Properties:

- Synchronization of physical time in LANs/WANs with UTC;  
Statistical techniques to:
  - consider the variance and delay of message transmissions
  - consider the quality of time on different time servers
- High availability by redundant servers and connections
- Scalability concerning the number of time servers, clients and the synchronization rate (adapted to Internet characteristics).
- Authentication of messages and time information to protect time services against manipulation

David L. Mills : „Internet Time Synchronization: The Network Time Protocol“, IEEE TC on Communications, 39(10), 1991

<http://www.ee.udel.edu/~mills/ntp.html> (20.5.07)





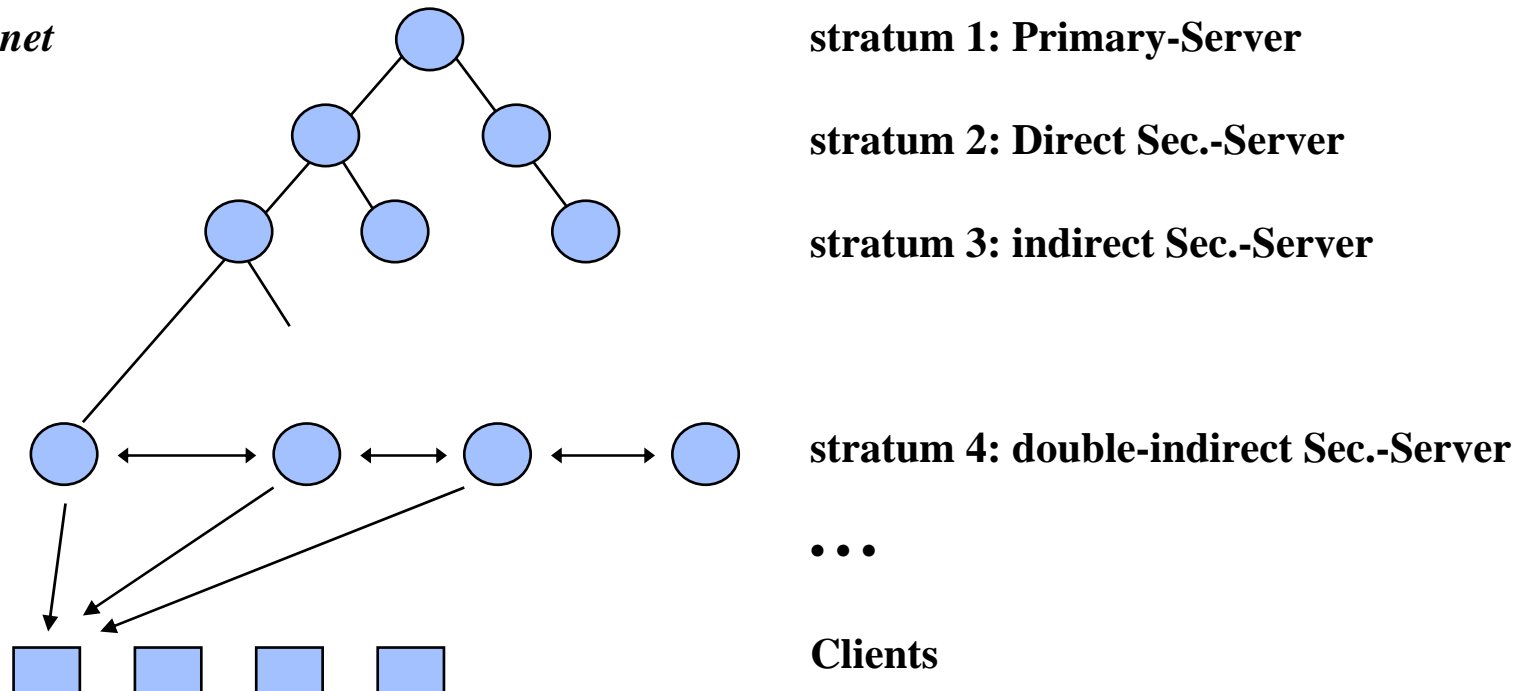
# The *Network Time Protocol* (Internet NTP)

---

## Hierarchy of time servers:

- Primary server: UTC-clock
- Secondary server: directly or indirectly connected with a primary server. May not have a UTC-clock.

## *Time Synchronisation Subnet*



# The *Network Time Protocol* (Internet NTP)

---

## **Reconfiguration**

- Clock failure in a primary server → server becomes secondary server

## **Communication**

- Cristian's algorithm: RPC-based communication, server send time-stamped replies
- on lower strata, symmetric relations between servers mutually exchanging time information
- in LANs: Multicast is used to distribute time information

## **High precision by exploiting the history of time messages:**

Every server adds to the time message  $t_A$

- the time stamp when the last request  $r_L$  was received
- the time stamp when the corresponding reply  $t_L$  has been sent
- the time stamp when the actual message  $t_A$  has been sent



---

# NTP Architecture, Protocol and Algorithms



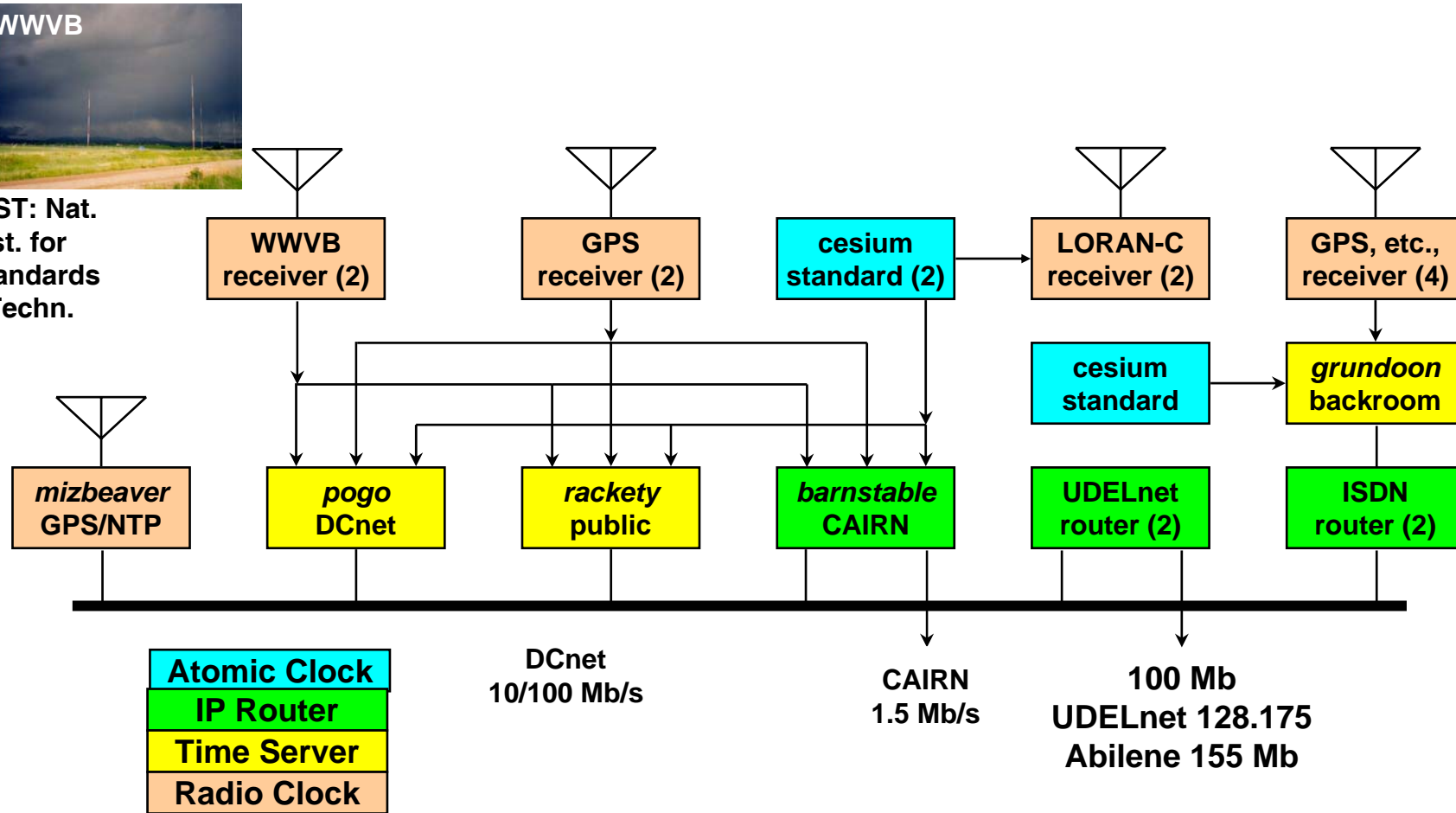
David L. Mills  
University of Delaware  
<http://www.ee.udel.edu/~mills>  
<mailto:mills@udel.edu>



# DCnet timekeeping facilities (January 2001)



NIST: Nat. Inst. for Standards & Techn.

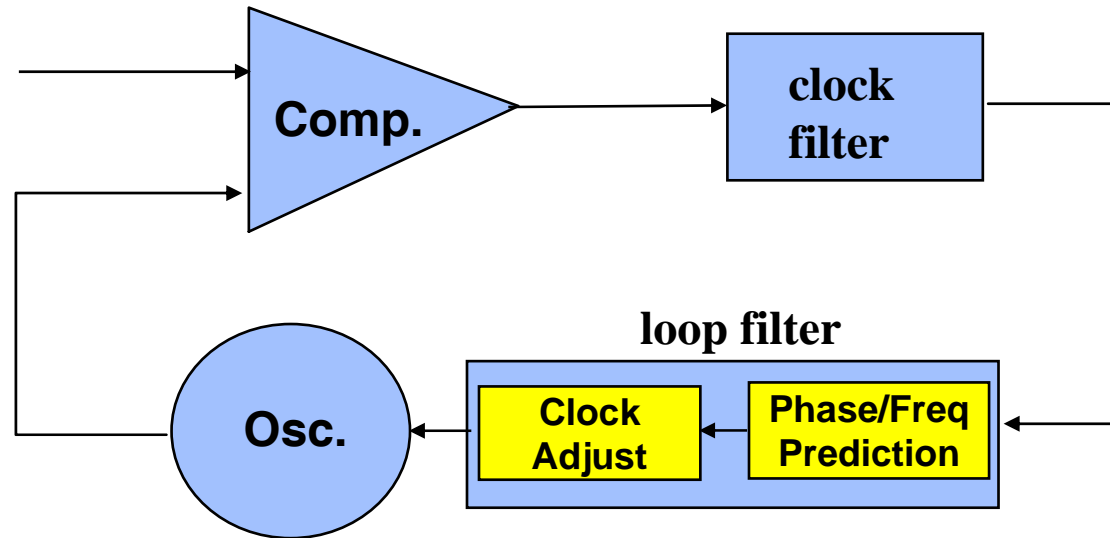


D. Mills/02



# PLL-Modell of NTP

---



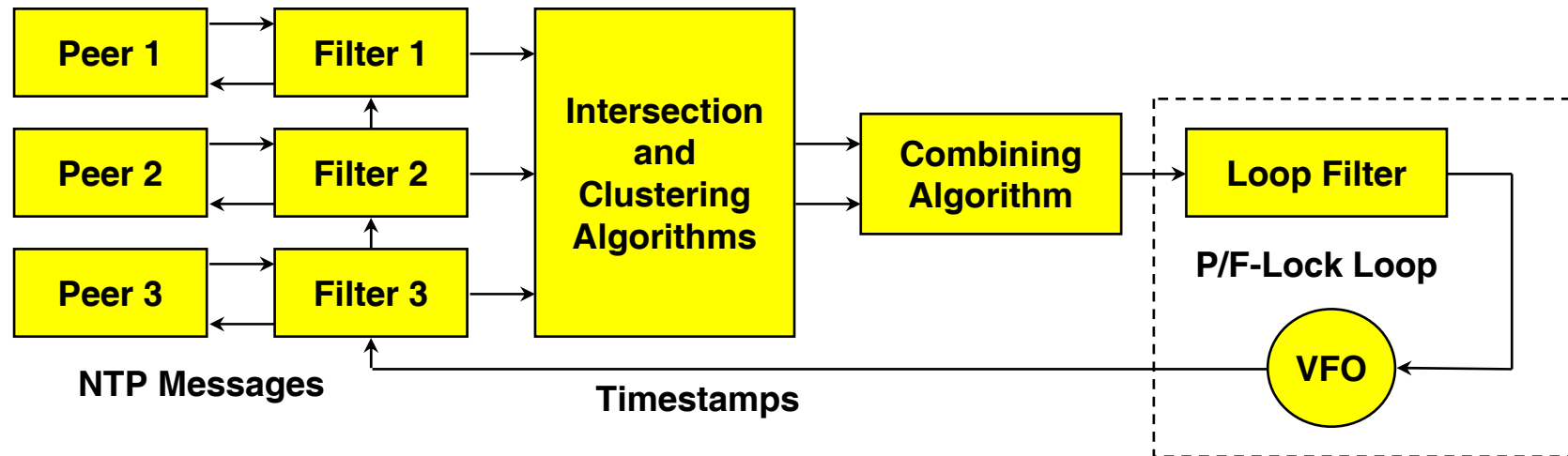
**NTP controls the "oszillator frequency"**

**clock filter:** selects appropriate time messages from the set of received time messages

**loop filter:** calculates the correction factor for the oszillator



# How NTP works

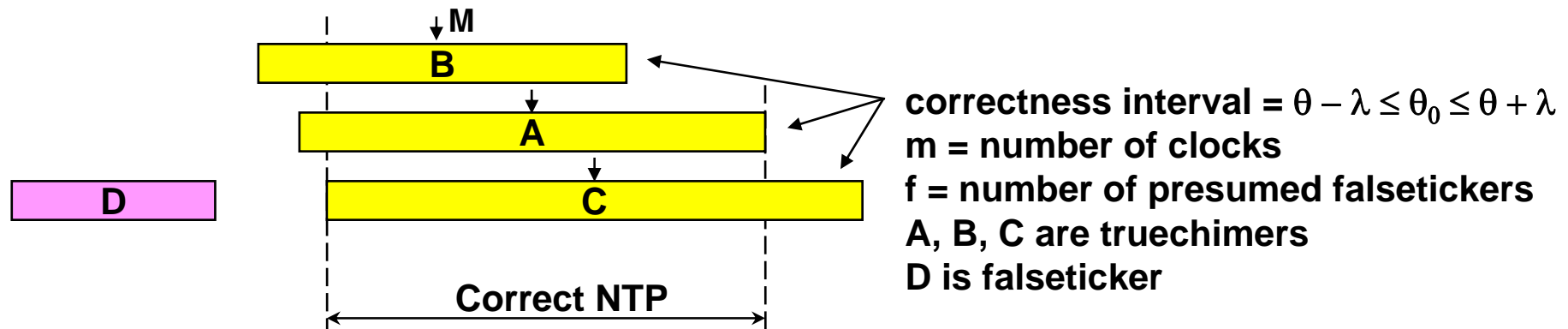


- Multiple synchronization peers provide redundancy and diversity
- Clock filters select best from a window of eight clock offset samples
- Intersection and clustering algorithms pick best subset of servers believed to be accurate and fault-free
- Combining algorithm computes weighted average of offsets for best accuracy
- Phase/frequency-lock feedback loop disciplines local clock time and frequency to maximize accuracy and stability

D. Mills/02



# Intersection algorithm

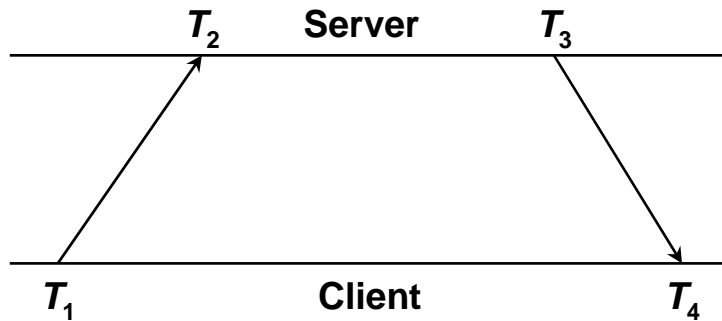


- NTP algorithm requires the *midpoint of the intervals* to be in the intersection
  - Initially, set falsetickers  $f$  and counters  $c$  and  $d$  to zero
    - Scan from far left endpoint: add one to  $c$  for every lower endpoint, subtract one for every upper endpoint, add one to  $d$  for every midpoint
    - If  $c \geq m - f$  and  $d \geq m - f$ , declare success and exit procedure
  - Do the same starting from the far right endpoint
    - If success undeclared, increase  $f$  by one and try all over again
    - if  $f \geq m/2$ , declare failure

D. Mills/02

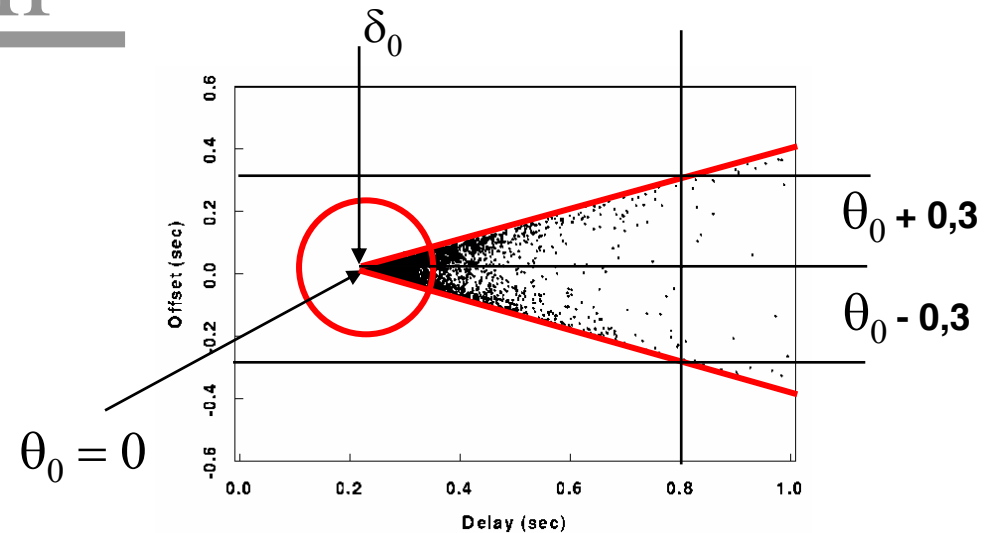


# Clock filter algorithm



$$\theta = \frac{1}{2} [(T_2 - T_1) + (T_3 - T_4)]$$

$$\delta = (T_4 - T_1) - (T_3 - T_2)$$



$$\frac{(\delta - \delta_0)}{2}$$

$$= \frac{(0,8 - 0,2)}{2}$$

$$= 0,3$$

- The most accurate offset  $\theta_0$  is measured at the lowest delay  $\delta_0$  (apex of the wedge scattergram).
  - The correct time  $\theta$  must lie within the wedge  $\theta_0 \pm (\delta - \delta_0)/2$ .
  - The  $\delta_0$  is estimated as the minimum of the last eight delay measurements and  $(\delta_0, \theta_0)$  becomes the offset and delay output.
  - Each output can be used only once and must be more recent than the previous output.
- The distance metric  $\lambda$  is based on delay, frequency tolerance and time since the last measurement.

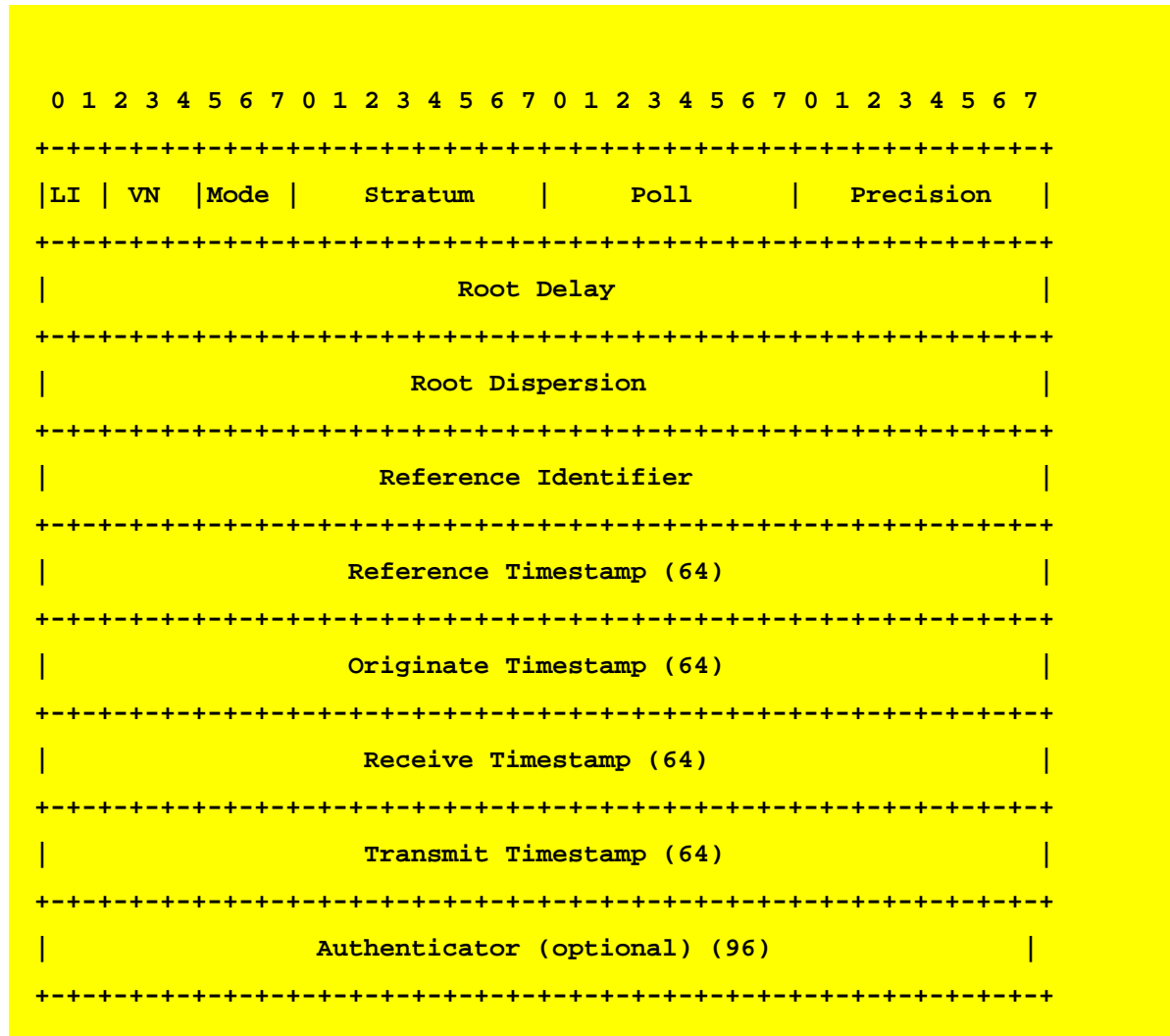
D. Mills/02







# NTP Packet Header



**LI** : Leap indicator, indicates a „leap second“ because of UT/UTC difference

**VN**: Version number of NTP Impl.

**Mode**: Operational smode, e.g. client or server

**Stratum**:  
 1: primary reference  
 2-15: secondary reference  
 16-255: reserved

**Poll**: Max. interval between consecutive time messages.

**Precision**: precision of local clock

**Root Delay**: Roundtrip Delay

**Root Dispersion**: Statistical error relative to the primary reference

**Ref. ID**: ID of the reference clock (e.g. DCF77)

**Ref.TS**: time of last synchronization

**Originate, Rx. Tx.:** Client req., Server Rx. Sever Tx to calculate roundtrip.



**Authenticator**: for optional authentication of time stamps.

# Control characteristic

---

## Phase error

**100ms**

corrected after 39 Min.

overshoot (overcompensation) 7ms after 54 Min.

stabilizes to error < 1ms after 6h

## Frequency error

**50 ppm**

1 ppm after 16h

0,1 ppm after 26 h

=> measurements: 99% of all time information deviates less than 30 msek from UTC.

If the clock is outside the capture of the PLL, the clock will be adjusted rather than the Oscillator frequency.



# Decentralized and cooperative approaches for clock synchronization

## Averaging:

L. Lamport, P. Melliar-Smith: "Synchronizing Clocks in the Presence of Faults",  
Journal of the ACM, 32(1): 52-78, 1985

J. Lundelius, N. Lynch: "A new Fault-Tolerant Algorithm for Clock Synchronization",  
Proc. 3rd ACM SIGACT-SIGOPS Symp. On Principles of Distr. Comp., 75-88, Vancouver,  
1984

## Non-averaging:

J. Halpern, B. Simons, R. Strong, D. Dolev: "Fault-Tolerant Clock Synchronization",  
Proc. 3rd ACM SIGACT-SIGOPS Symp. On Principles of Distr. Comp., 89-102, Vancouver,  
1984

T. Srikanth, S. Toueg: "Optimal Clock Synchronization", Journal of the ACM, (34)3,  
627-645, 1987

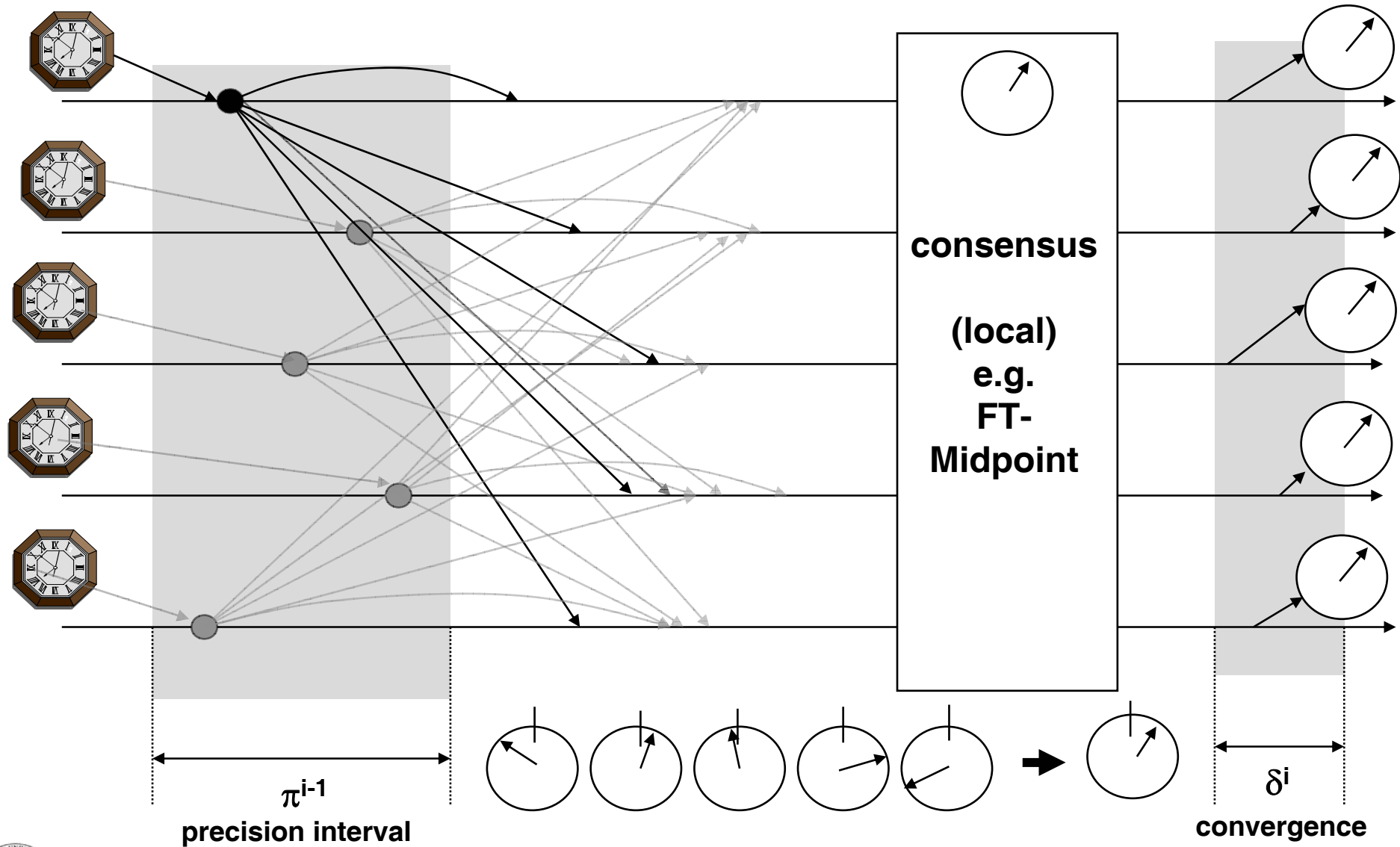
## Hybrid:

P. Verissimo, L. Rodrigues: "A posteriori Agreement for Fault-Tolerant Clock  
Synchronization on Broadcast Networks", Digest of papers, 2nd IEEE Int'l Symp. On  
Fault-Tolerant Computing, Boston, 1992

M. Clegg, K. Marzullo: "Clock Synchronization in Hard Real-Time Distributed Systems",  
Technical report CS96-478, UCSD, Dept. Of Comp. Science, 1996



# Averaging clock synchronization



# Fault-tolerant convergence algorithms:

- **Fault-tolerant Midpoint and**
- **Fault-Tolerant Average**

## – Fault-tolerant Midpoint:

The  $k$  largest and the  $k$  smallest values will not be considered.

The average is computed from the values  $k+1$  und  $n-k$ :

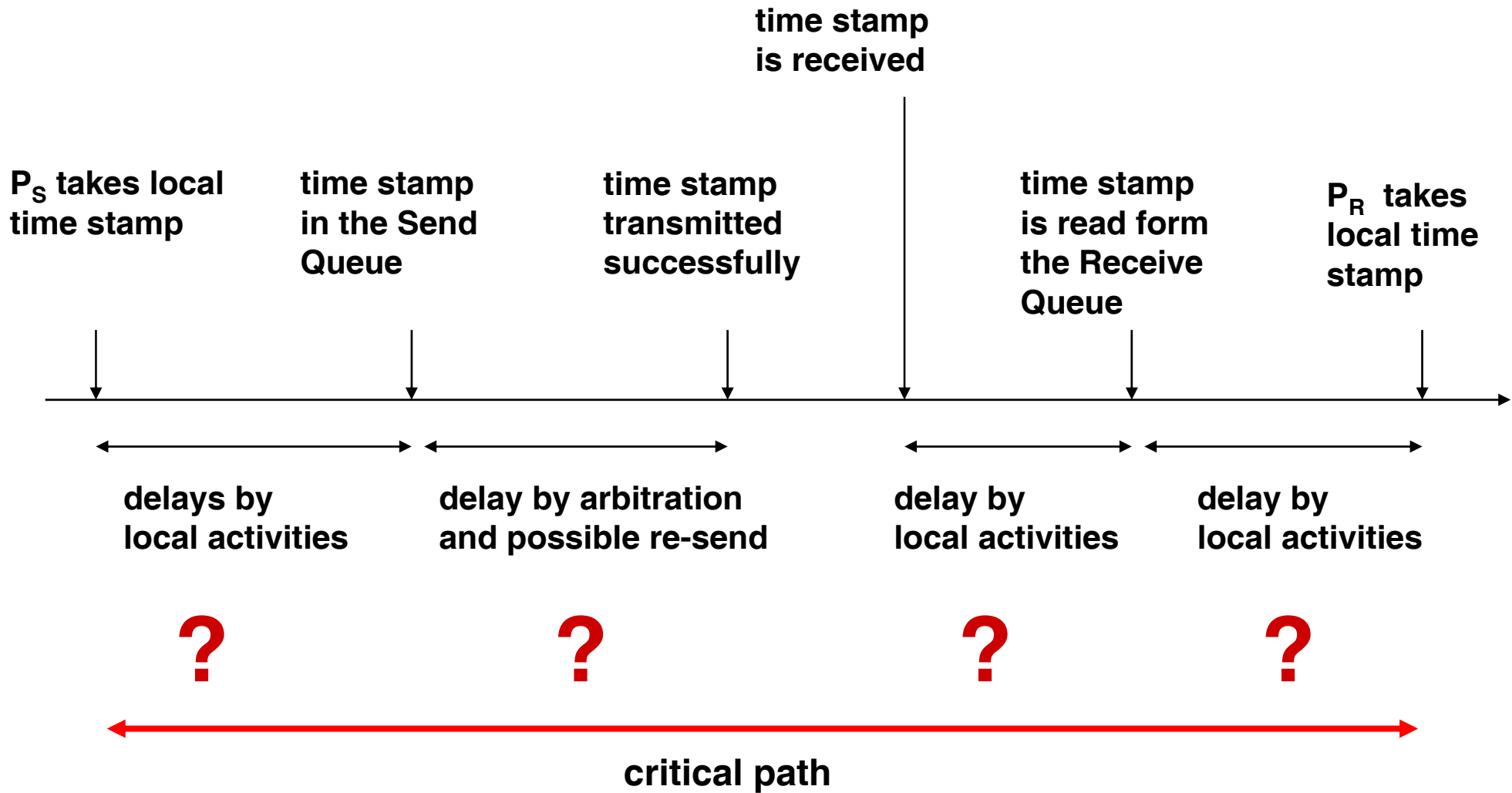
- **Midpoint:  $(\text{max\_value} + \text{min\_value}) / 2$**
- this is NOT the MEAN value of the sorted list of values!!!!

## – Fault-tolerant Average:

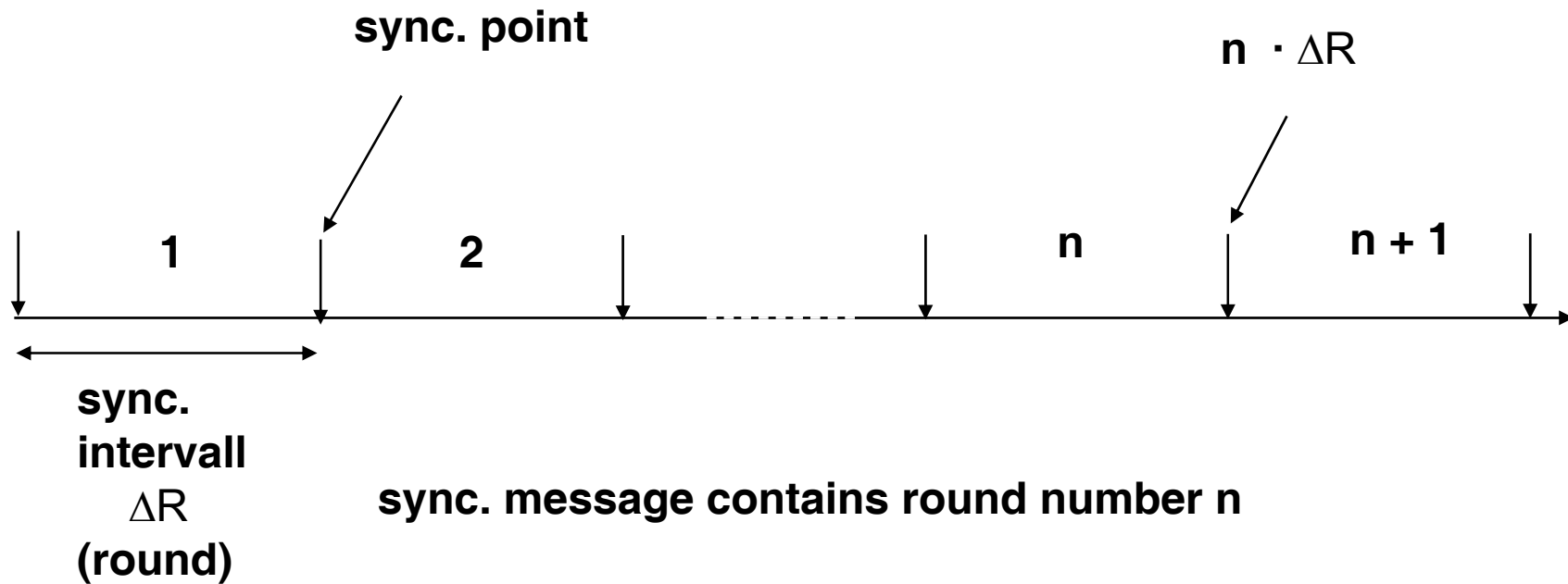
The  $k$  largest and the  $k$  smallest values will not be considered. The average will be computed from the remaining values.



# Practical Considerations:

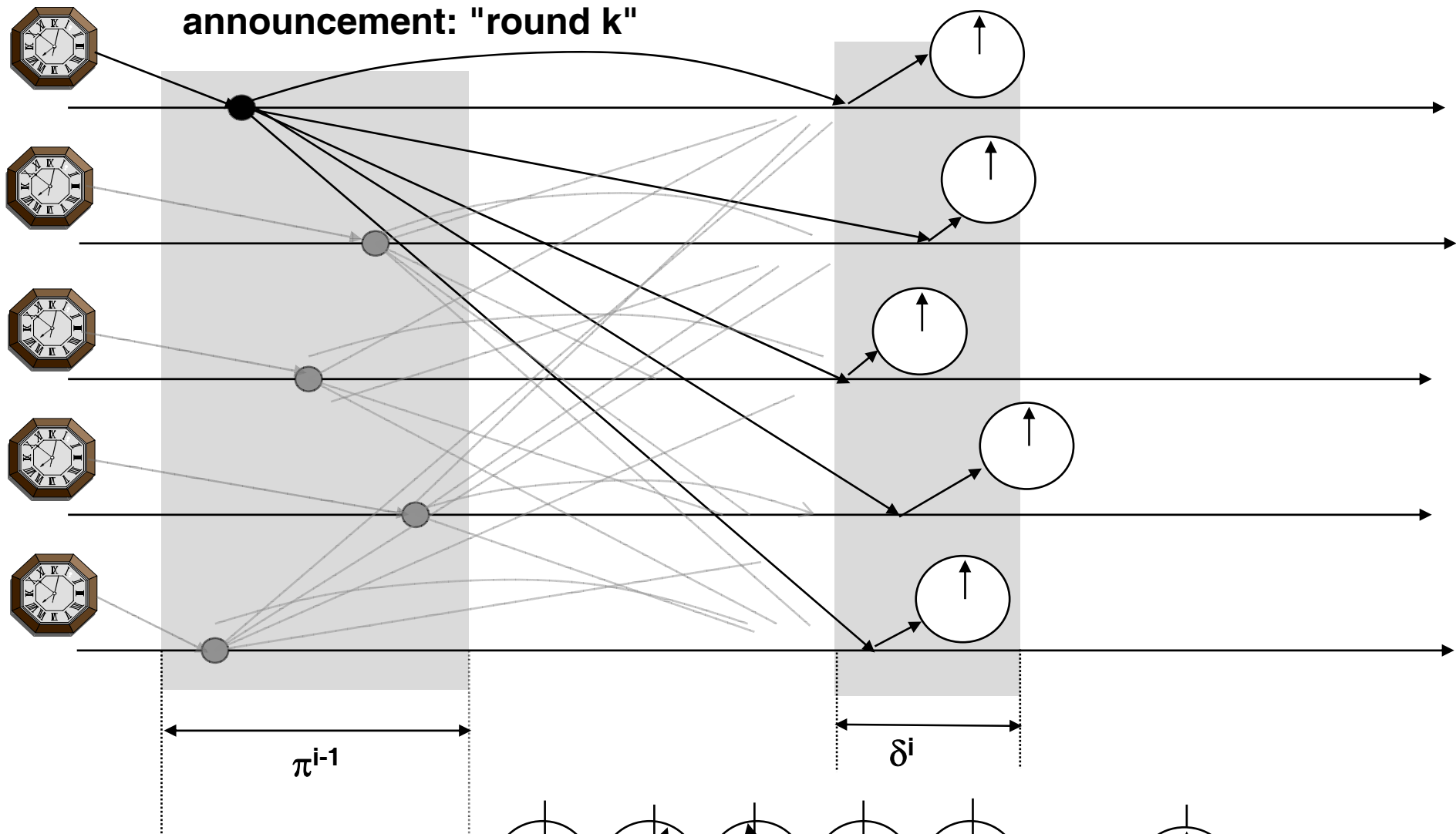


# Non-averaging synchronization

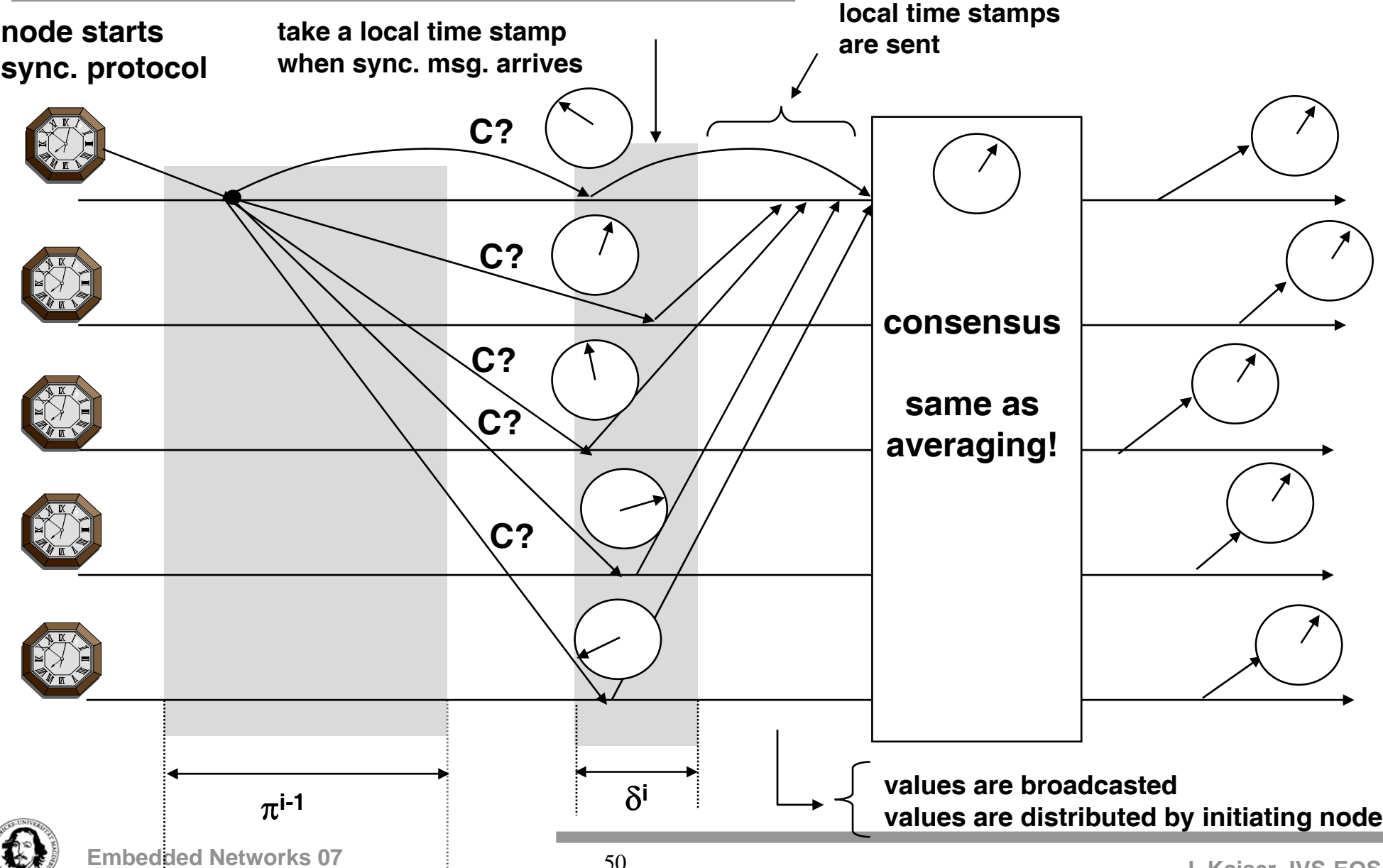


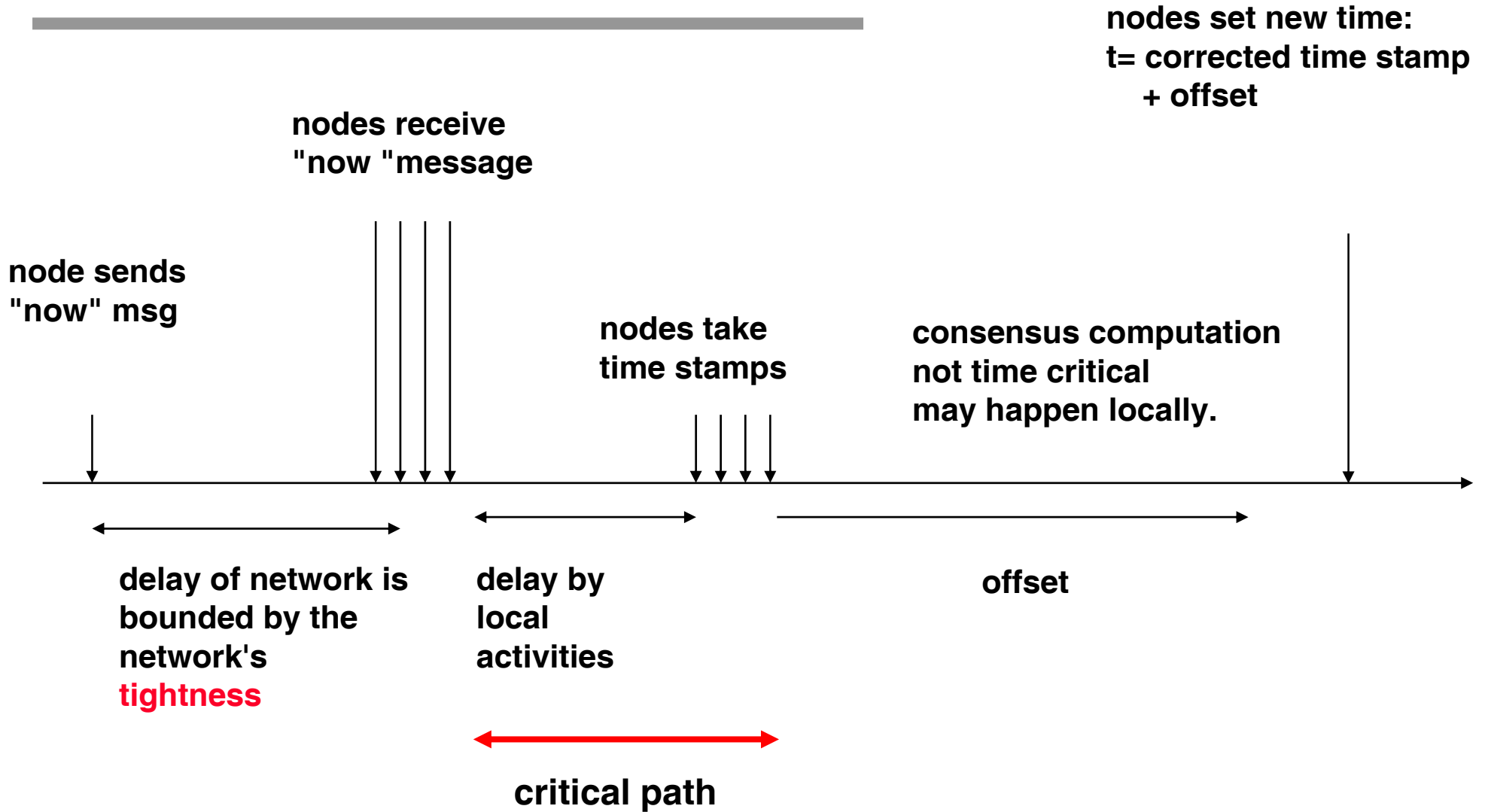


# non averaging synchronization



# Hybride synchronization





## Co-operating clock synchronization (Kopetz, Ochsenreiter)\*

$t_{i,j}$  : time when the synchronization message sent by node  $i$  was received by node  $j$

For  $n$  nodes, the local matrix of received time stamps can be constructed as:

$$C_j = \begin{pmatrix} t_{1,1} & \cdot & \cdot & \cdot & \cdot & t_{1,j} & \cdot & \cdot & \cdot & \cdot \\ \cdot & t_{2,2} & \cdot & \cdot & \cdot & t_{2,j} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & t_{3,j} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & t_{j,j} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & t_{n,j} & \cdot & \cdot & \cdot & t_{n,n} \end{pmatrix}$$

\*

H. Kopetz, W. Ochsenreiter  
Clock Synchronisation in Distributed Real-Time Systems  
IEEE Transactions on Computers, Vol. C-36, No.8  
August 1988

The diagonal contains the local time stamps ( $i=j$ , i.e. the sending and receiving node are identical).  
The column  $j$  contains the receive times of all messages sent by other nodes on node  $j$ .

For all good clocks the following equation holds:

$$t_{i,j} = t_{i,i} + (O_{i,j} + md + E_{i,j}) \cdot N \quad (N = 1 \text{ if } i \neq j, N = 0 \text{ if } i = j)$$

$O_{i,j}$  (offset) : offset between clocks on nodes  $i$  and  $j$

$md$  : message delay

$E_{i,j}$  : sum of the reading errors on nodes  $i$  and  $j$



# Co-operating clock synchronization

correction vector for node j

$$\mathbf{K}^t = \left( \begin{array}{ccccccc}
 \mathbf{0} & \mathbf{k}_{1,2} & \dots & \mathbf{k}_{1,j} & \dots & \mathbf{k}_{1,n} & \\
 \mathbf{k}_{2,1} & \mathbf{0} & \dots & \mathbf{k}_{2,j} & \dots & \mathbf{k}_{2,n} & \\
 \cdot & \cdot & & \mathbf{k}_{3,j} & & \cdot & \\
 \cdot & \cdot & \mathbf{0} & & & & \\
 \cdot & \cdot & \dots & \mathbf{0} & \dots & \cdot & \\
 \cdot & \cdot & & & & \cdot & \\
 & & & & \mathbf{0} & & \\
 \mathbf{k}_{n,1} & \mathbf{k}_{n,2} & \dots & \mathbf{k}_{n,j} & \dots & \mathbf{0} & 
 \end{array} \right)$$

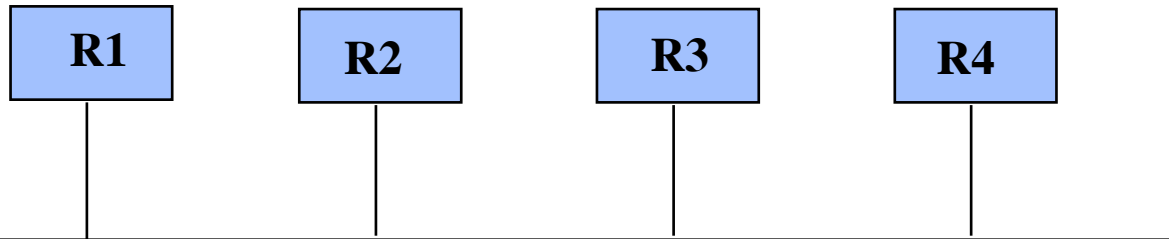
For every element  $k_{i,j}$  :

$$k_{i,j} = t_{i,j} - t_{i,i} - md \cdot N \quad ( N = 1 \text{ if } i \neq j, N = 0 \text{ if } i = j )$$



# Co-operating clock synchronization

**Example:**



**md=6**

$t_{1,1} = 32, 32$	$t_{1,2} = 37, 32$	$t_{1,3} = 38, 32$	$t_{1,4} = 41, 32$
$t_{2,1} = 69, 60$	$t_{2,2} = 60, 60$	$t_{2,3} = 67, 60$	$t_{2,4} = 66, 60$
$t_{3,1} = 95, 87$	$t_{3,2} = 94, 87$	$t_{3,3} = 87, 87$	$t_{3,4} = 98, 87$
$t_{4,1} = 20, 16$	$t_{4,2} = 19, 16$	$t_{4,3} = 21, 16$	$t_{4,4} = 16, 16$

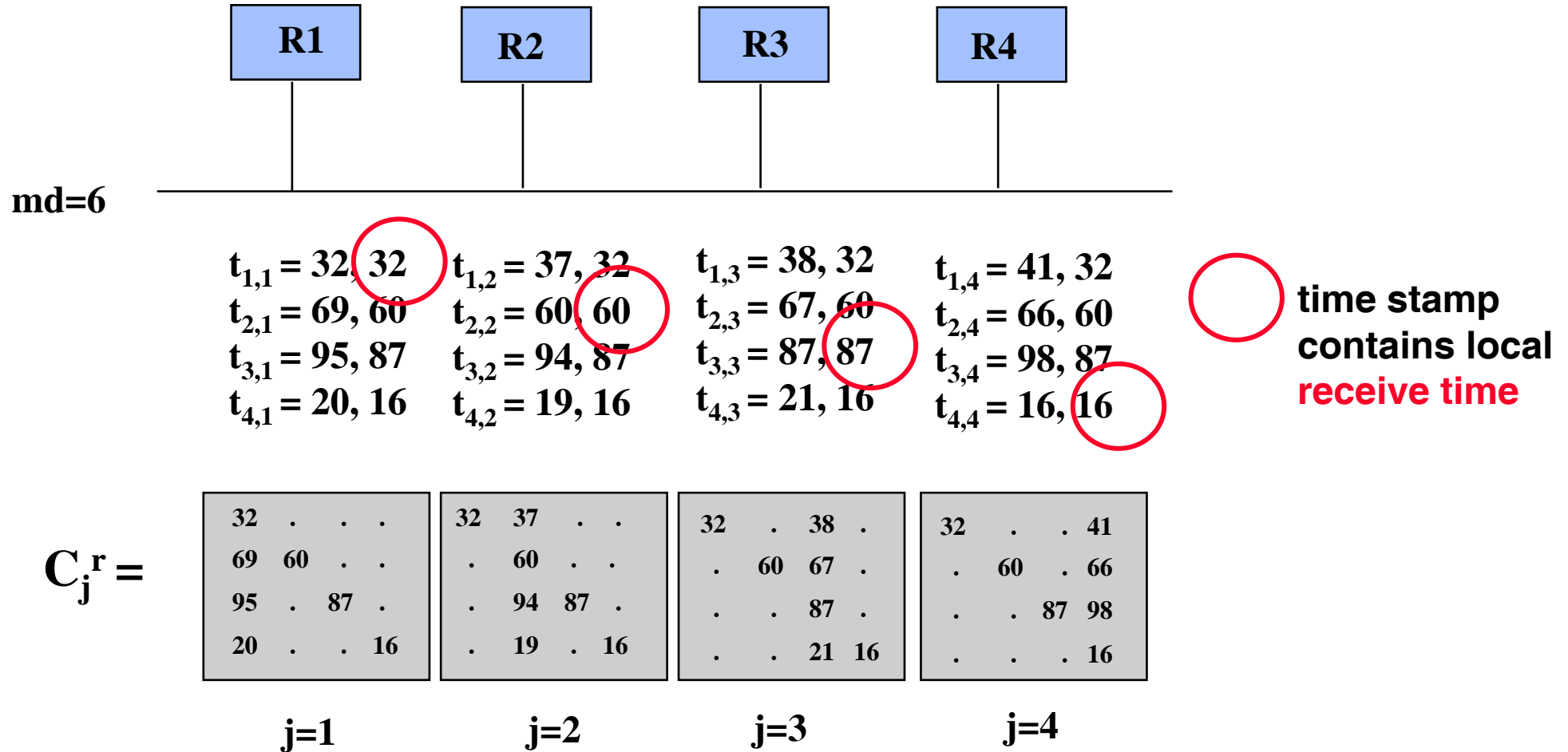
$C_j^r =$

<table border="0"> <tr><td>32</td><td>.</td><td>.</td><td>.</td></tr> <tr><td>69</td><td>60</td><td>.</td><td>.</td></tr> <tr><td>95</td><td>.</td><td>87</td><td>.</td></tr> <tr><td>20</td><td>.</td><td>.</td><td>16</td></tr> </table>	32	.	.	.	69	60	.	.	95	.	87	.	20	.	.	16	<table border="0"> <tr><td>32</td><td>37</td><td>.</td><td>.</td></tr> <tr><td>.</td><td>60</td><td>.</td><td>.</td></tr> <tr><td>.</td><td>94</td><td>87</td><td>.</td></tr> <tr><td>.</td><td>19</td><td>.</td><td>16</td></tr> </table>	32	37	.	.	.	60	.	.	.	94	87	.	.	19	.	16	<table border="0"> <tr><td>32</td><td>.</td><td>38</td><td>.</td></tr> <tr><td>.</td><td>60</td><td>67</td><td>.</td></tr> <tr><td>.</td><td>.</td><td>87</td><td>.</td></tr> <tr><td>.</td><td>.</td><td>21</td><td>16</td></tr> </table>	32	.	38	.	.	60	67	.	.	.	87	.	.	.	21	16	<table border="0"> <tr><td>32</td><td>.</td><td>.</td><td>41</td></tr> <tr><td>.</td><td>60</td><td>.</td><td>66</td></tr> <tr><td>.</td><td>.</td><td>87</td><td>98</td></tr> <tr><td>.</td><td>.</td><td>.</td><td>16</td></tr> </table>	32	.	.	41	.	60	.	66	.	.	87	98	.	.	.	16
32	.	.	.																																																																
69	60	.	.																																																																
95	.	87	.																																																																
20	.	.	16																																																																
32	37	.	.																																																																
.	60	.	.																																																																
.	94	87	.																																																																
.	19	.	16																																																																
32	.	38	.																																																																
.	60	67	.																																																																
.	.	87	.																																																																
.	.	21	16																																																																
32	.	.	41																																																																
.	60	.	66																																																																
.	.	87	98																																																																
.	.	.	16																																																																
<b>j=1</b>	<b>j=2</b>	<b>j=3</b>	<b>j=4</b>																																																																



# Co-operating clock synchronization

Example:



# Co-operating clock synchronization

$C_j^r =$

32	.	.	.	32	38	.	.	32	.	38	.	32	.	.	41
69	60	.	.	.	60	.	.	.	60	67	.	.	60	.	71
95	.	87	.	.	94	87	.	.	.	87	.	.	.	87	98
20	.	.	16	.	19	.	16	.	.	21	16	.	.	.	16

$j = 1$

$$\begin{aligned}
 k_{1,1} &= 0 \\
 k_{2,1} &= t_{2,1} - t_{2,2} - md = 69 - 60 - 6 = 3 \\
 k_{3,1} &= t_{3,1} - t_{3,3} - md = 95 - 87 - 6 = 2 \\
 k_{4,1} &= t_{4,1} - t_{4,4} - md = 20 - 16 - 6 = -2
 \end{aligned}$$

$j = 2$

$$\begin{aligned}
 k_{1,2} &= t_{1,2} - t_{1,1} - md = 37 - 32 - 6 = -1 \\
 k_{2,2} &= 0 \\
 k_{3,2} &= t_{3,2} - t_{3,3} - md = 94 - 87 - 6 = 1 \\
 k_{4,2} &= t_{4,2} - t_{4,4} - md = 19 - 16 - 6 = -3
 \end{aligned}$$

$j = 3$

$$\begin{aligned}
 k_{1,3} &= t_{1,3} - t_{1,1} - md = 38 - 32 - 6 = 0 \\
 k_{2,3} &= t_{2,3} - t_{2,2} - md = 67 - 60 - 6 = 1 \\
 k_{3,3} &= 0 \\
 k_{4,3} &= t_{4,3} - t_{4,4} - md = 21 - 16 - 6 = -1
 \end{aligned}$$

$j = 4$

$$\begin{aligned}
 k_{1,4} &= t_{1,4} - t_{1,1} - md = 41 - 32 - 6 = 3 \\
 k_{2,4} &= t_{2,4} - t_{2,2} - md = 71 - 60 - 6 = 5 \\
 k_{3,4} &= t_{3,4} - t_{3,3} - md = 98 - 87 - 6 = 5 \\
 k_{4,4} &= 0
 \end{aligned}$$

$K^t =$

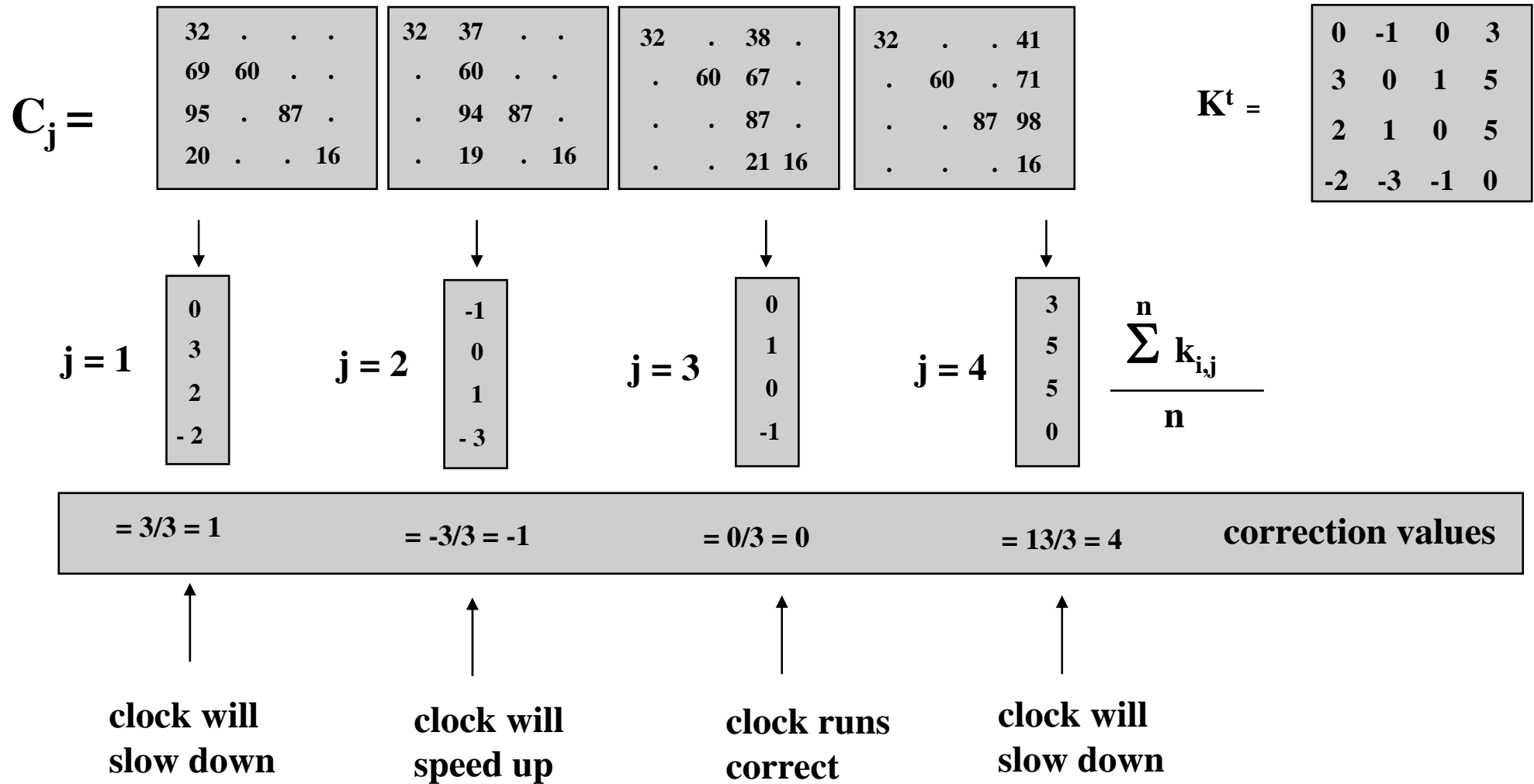
0	-1	0	3
3	0	1	5
2	1	0	5
-2	-3	-1	0





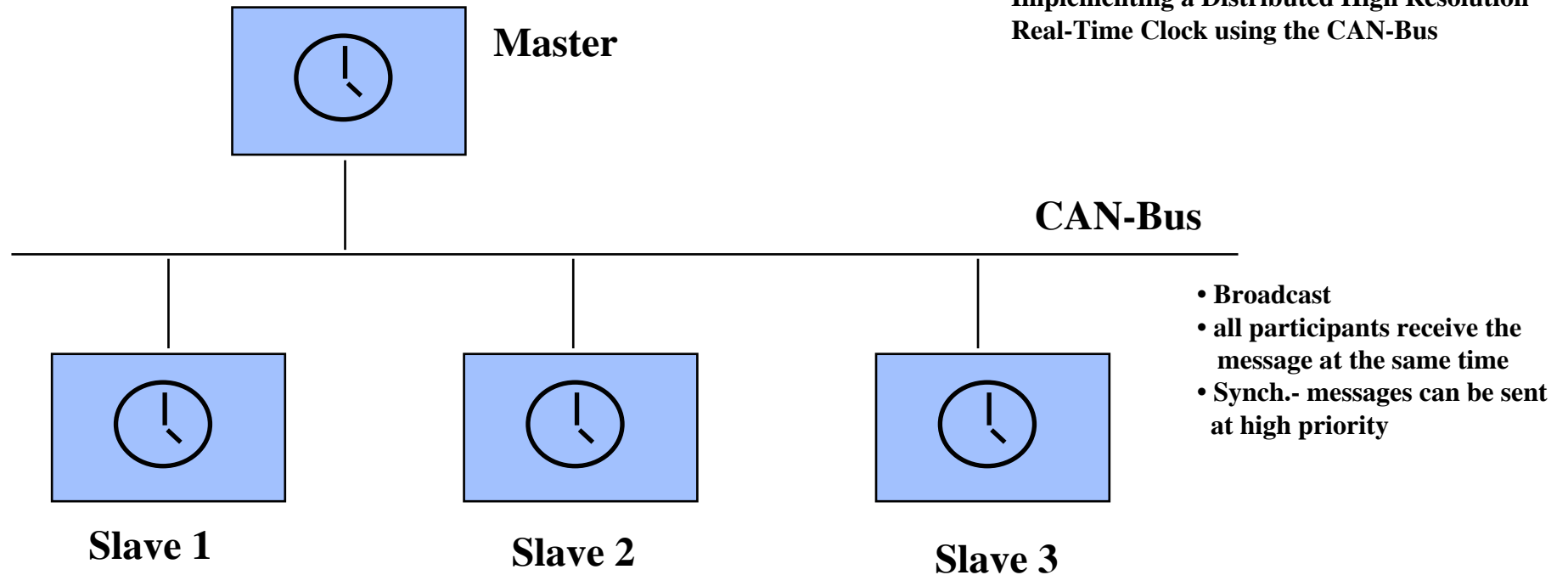
# Co-operating clock synchronization

## Example:



# Clock synchronization on the CAN-Bus\*

\* M. Gergeleit, H. Streich  
Implementing a Distributed High Resolution  
Real-Time Clock using the CAN-Bus



## System:

- 1 master, many slaves
- every participant has a local clock
- local clocks can be adjusted
- all clocks produce monotonically increasing values

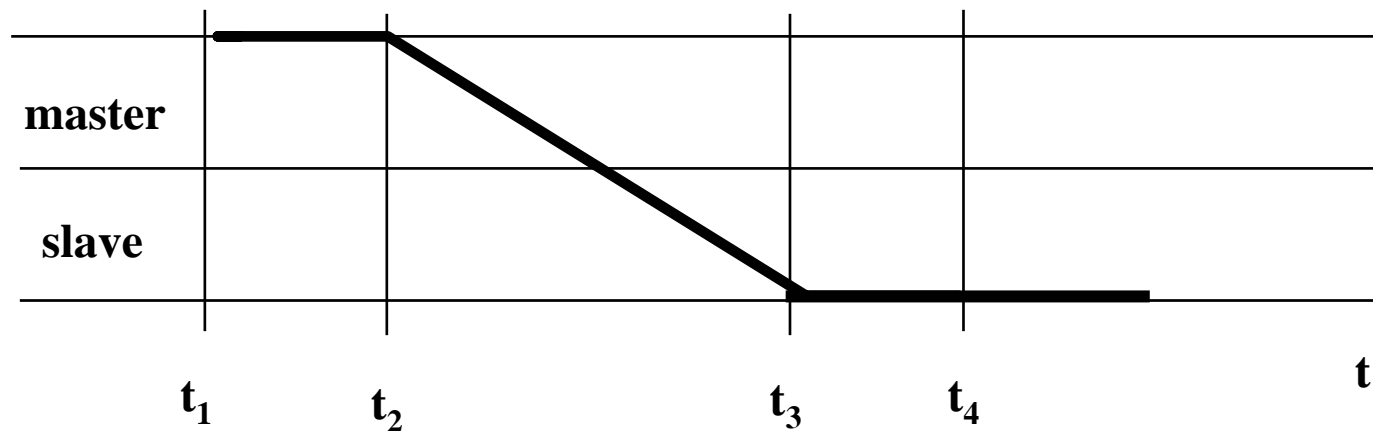


# Clock synchronization on the CAN-Bus

---

## simple protocol

- **master sends a time stamp**
- **slaves receive the message**
- **slaves adjust local clocks**

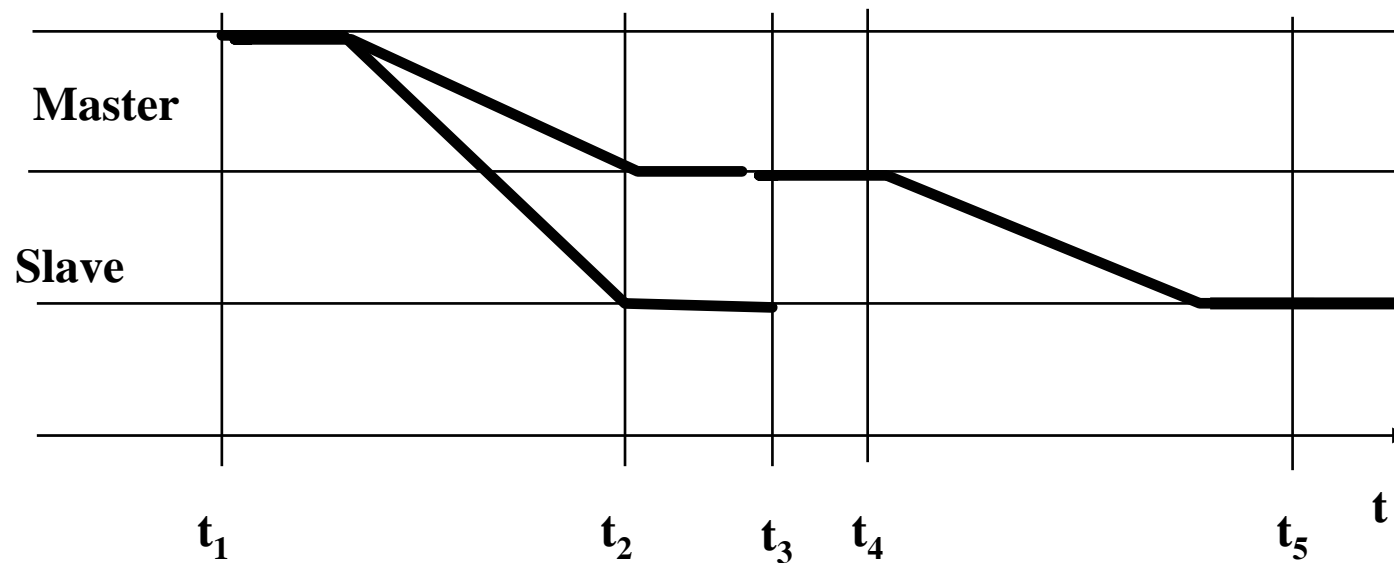


# Clock synchronization on the CAN-Bus\*

---

## improved protocol

- some node  $i$  sends the message "now" ( $t_1$ )
- master and slaves read their local clocks when receiving the message at ( $t_2$ )
- master sends its time stamp ( $t_3, t_4$ )
- slaves adjust their clocks according to the master time stamp ( $t_5$ )

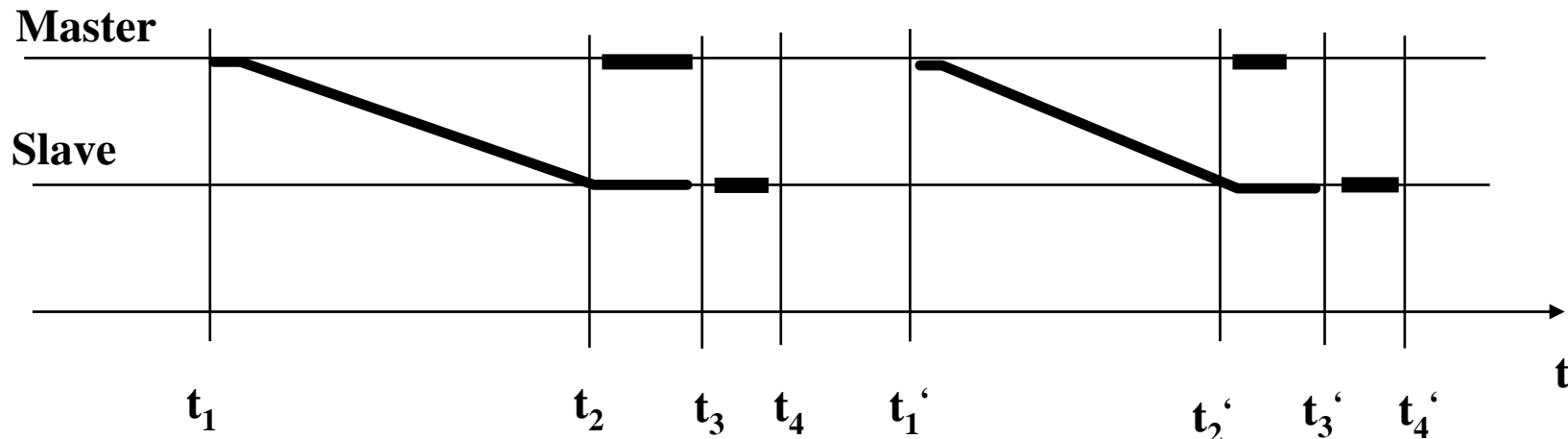


# Clock synchronization on the CAN-Bus\*

---

## optimized protokol

- master sends message "now" ( $t_1$ )
- this message contains the time stamp of the last synchronization round
- master reads its clock on the receive interrupt of the message ( $t_3$ )
- slaves read their clocks on the receive interrupt of the message ( $t_3$ )
- slaves adjust their clocks according to the time stamps of the last round ( $t_4$ )



# Clock synchronization on the CAN-Bus\*

- ◎ **average precision**
  - about 30  $\mu$ sec
- ◎ **synchronization with external time**
  - e.g. via GPS receiver
- ◎ **no additional cost**
  - implemented on embedded controller
- ◎ **Application areas:**
  - co-operative data acquisition
  - co-operative control

