

---

# Real (physical) 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 papers:

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

Bharath Sundaraman, Ugo Buy, Ajay D. Kshemkalyani: "Clock Synchronization for Wireless Sensor Networks: A Survey", *Ad-Hoc Networks*, Volume 3, May 2005



# What is time ?

---

**Informal short introduction to the concept of time and the evolving understanding of the notion of time (in German).**

J. Kaiser

As Time Goes By - Das Konzept der Zeit, ihre Messung und ihre Rolle  
in Echtzeitsystemen

in: K.G. Gaida: ZEITVERTREIB, Wo sind Wir stehengeblieben,  
Salon Verlag, ISBN 3-932189-63-9, 1998

**Long version available on the web**



# What is the function of time ?

## **Orientation:**

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

## **Regulation**

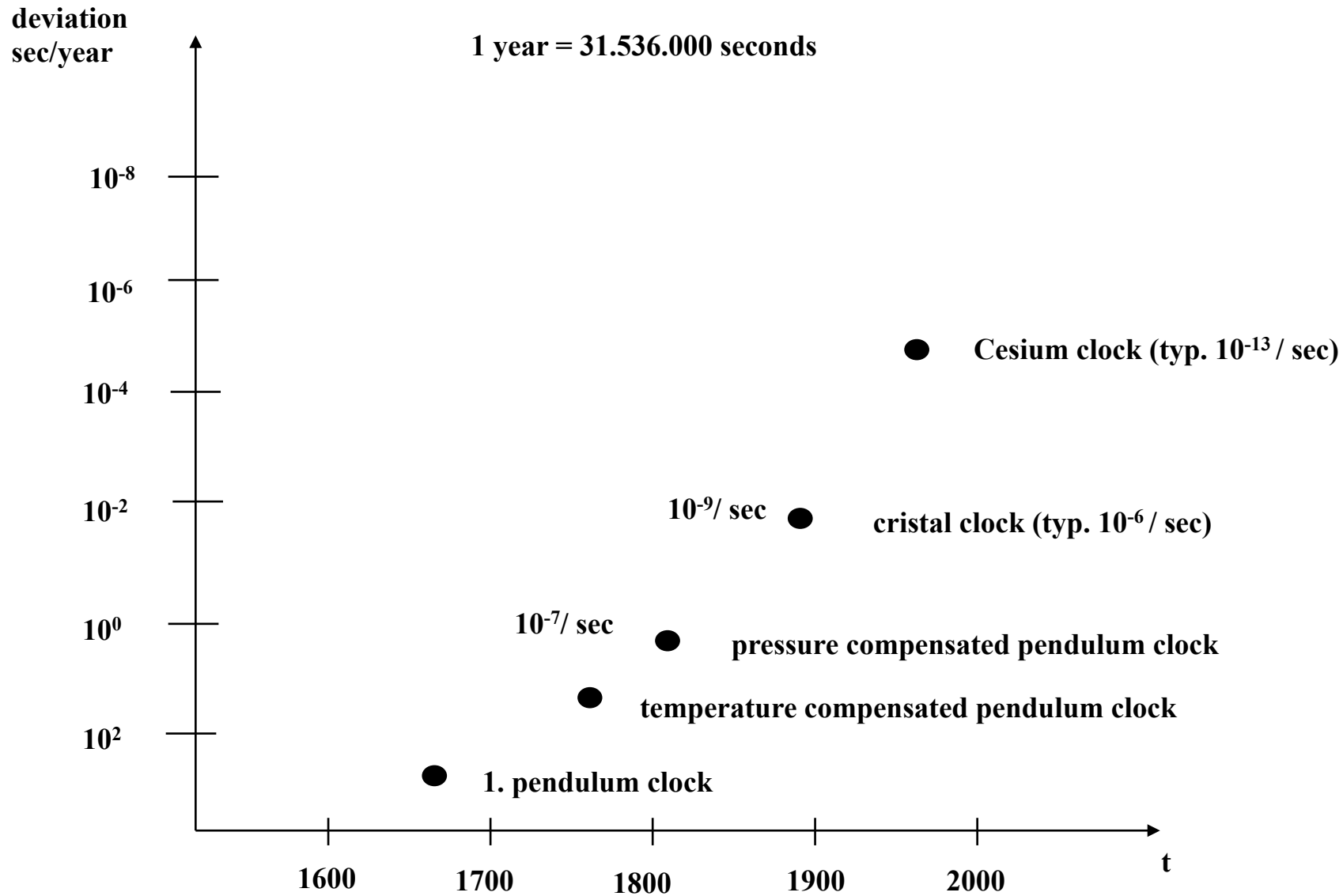
**Enforcing a time regime.**

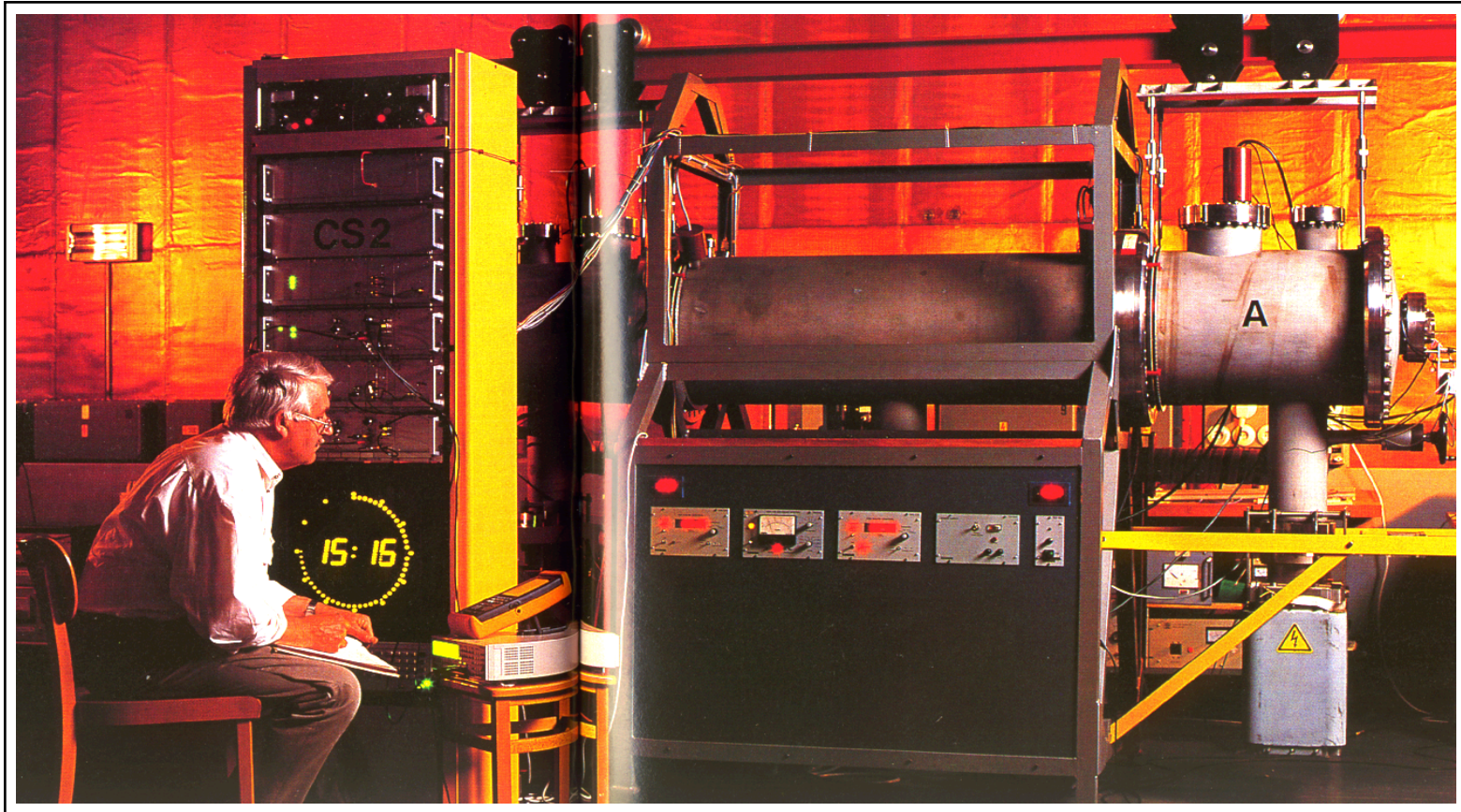
## **Coordination**

**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

■	<b>Astronomische Zeit (AT)</b>	basiert auf der gleichförmigen Bewegung von Himmelskörpern
	<b>Physikalische Zeit (PT)</b>	basiert auf periodischen physikalischen Prozessen (Schwingungen)
	<b>Sonnendurchlauf:</b>	Höchster Punkt der Sonne während eines Tages
	<b>Sonntag:</b>	Intervall zwischen zwei aufeinanderfolgenden Sonnendurchläufen
	<b>Sonnensekunde:</b>	Der 1 / 86400 Teil eines Sonntages
	<b>Mittlere Sonnensekunde:</b>	Sonnensekunde gemittelt über eine große Anzahl von Sonnentagen
	<b>Zeitzone:</b>	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:** Given two events  $e_1$  and  $e_2$  at time  $t_{e1}$  and  $t_{e2}$  respectively on an internal time scale. If  $t_{e1} \leq t_{e2}$  holds internally, this relation must also hold on an external time scale.

The following additional properties can be derived:

- 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.

**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 . .

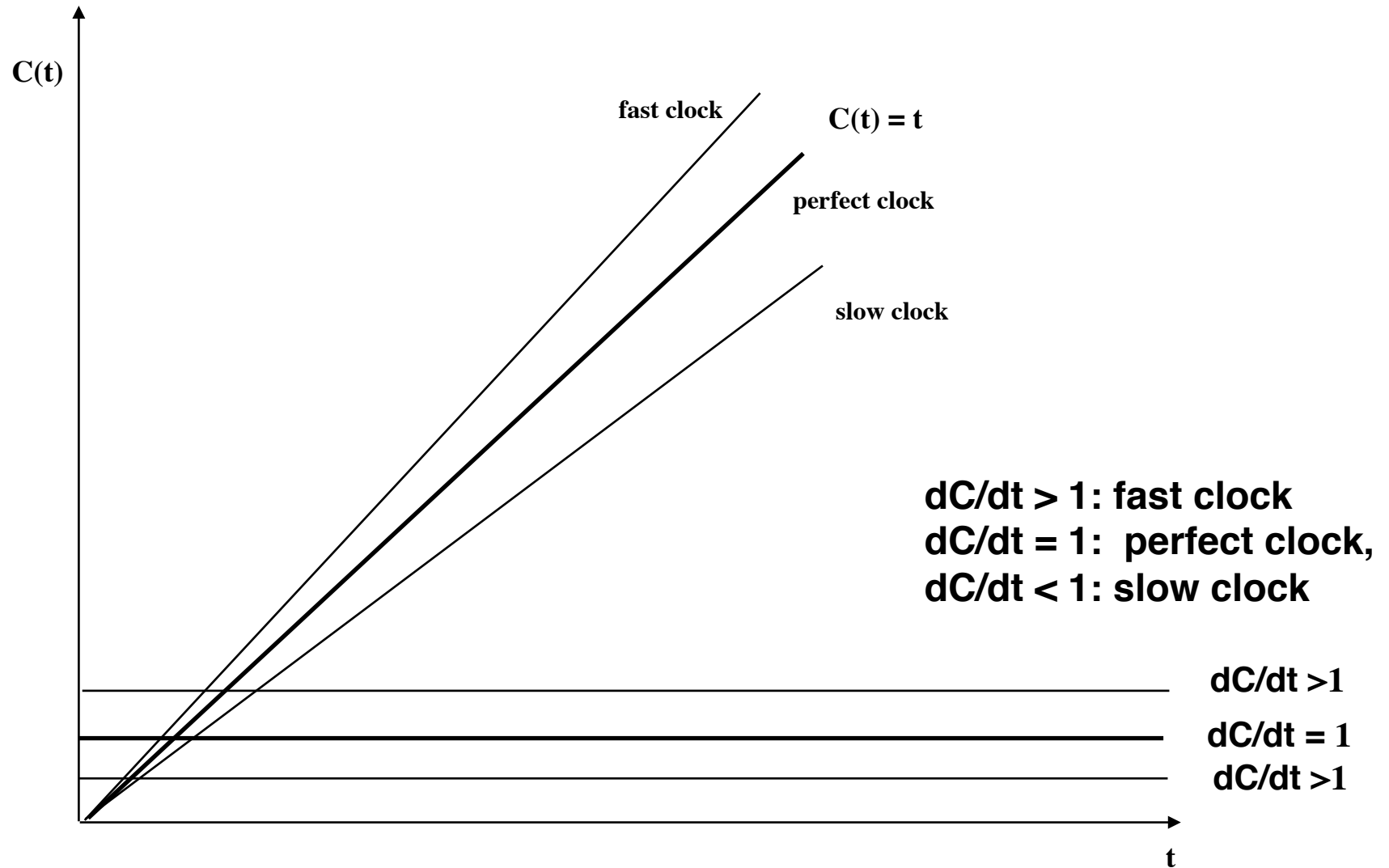
**Granularity:  $g$**  time interval between two clock pulses (corresponds to a period).  
Can only be determined by a clock of higher granularity. The granularity determines the lower bound of the temporal distance between two events that are not recognized as simultaneous events.

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



# Clock with constant Drift Rate

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



# Terminology

---

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



# Clock Synchronization

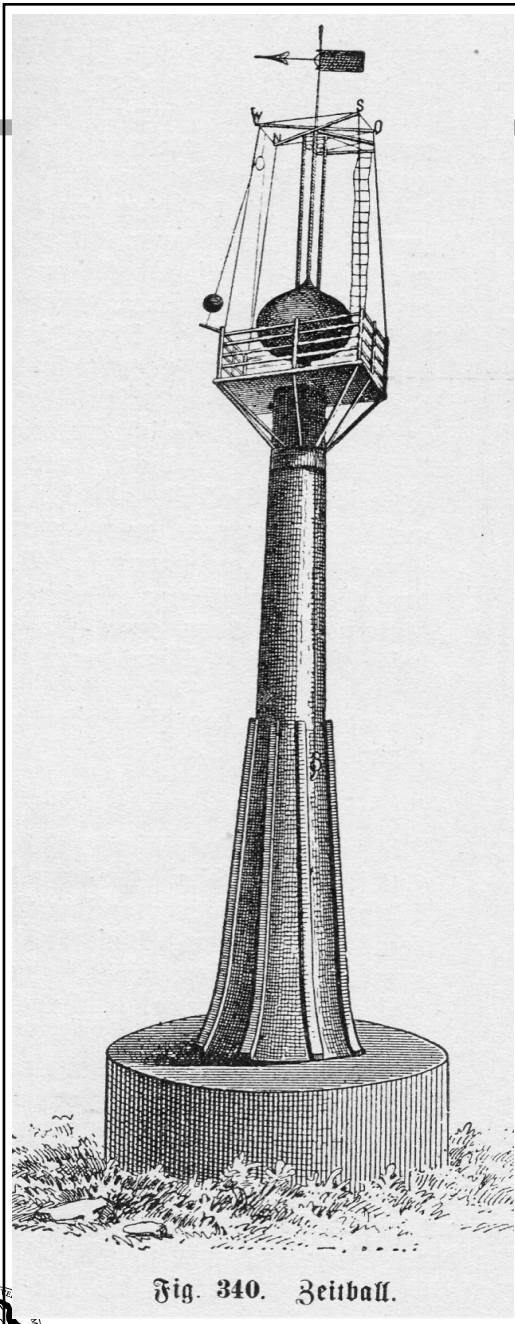
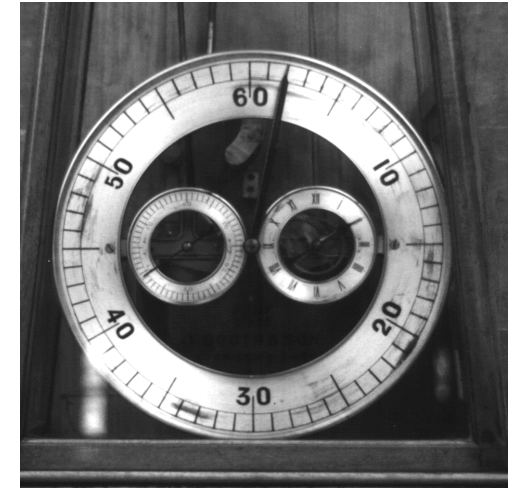
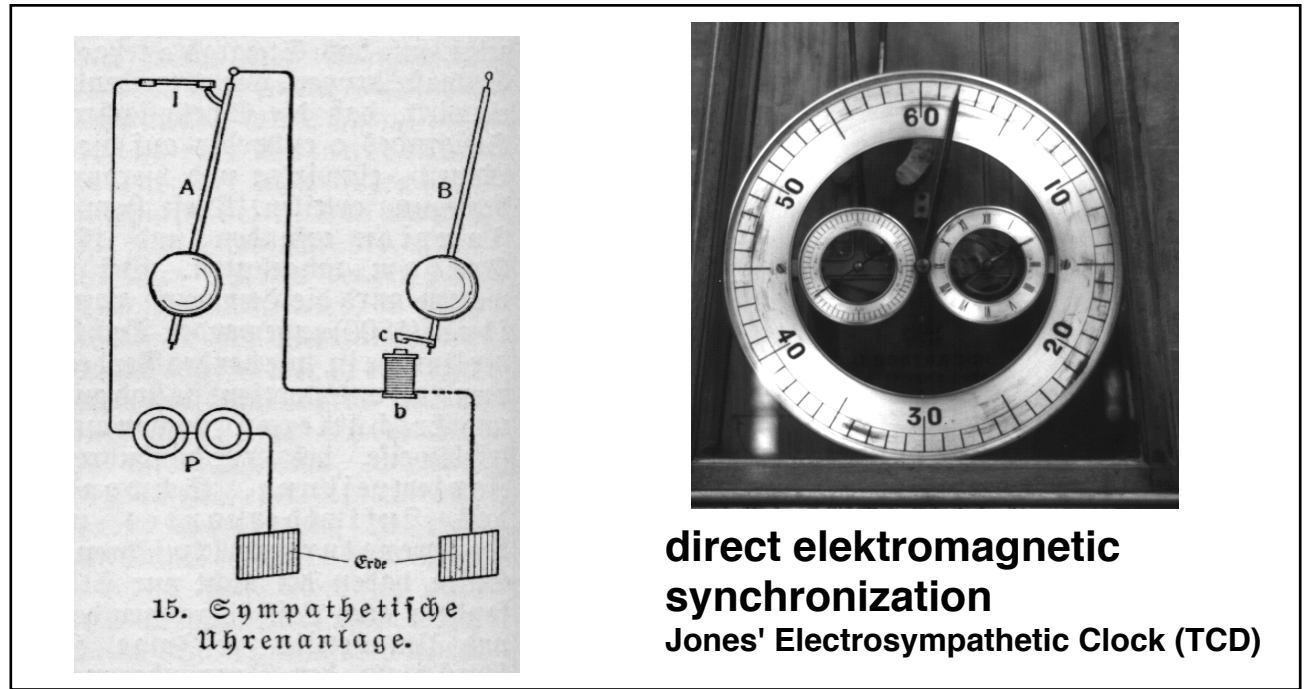
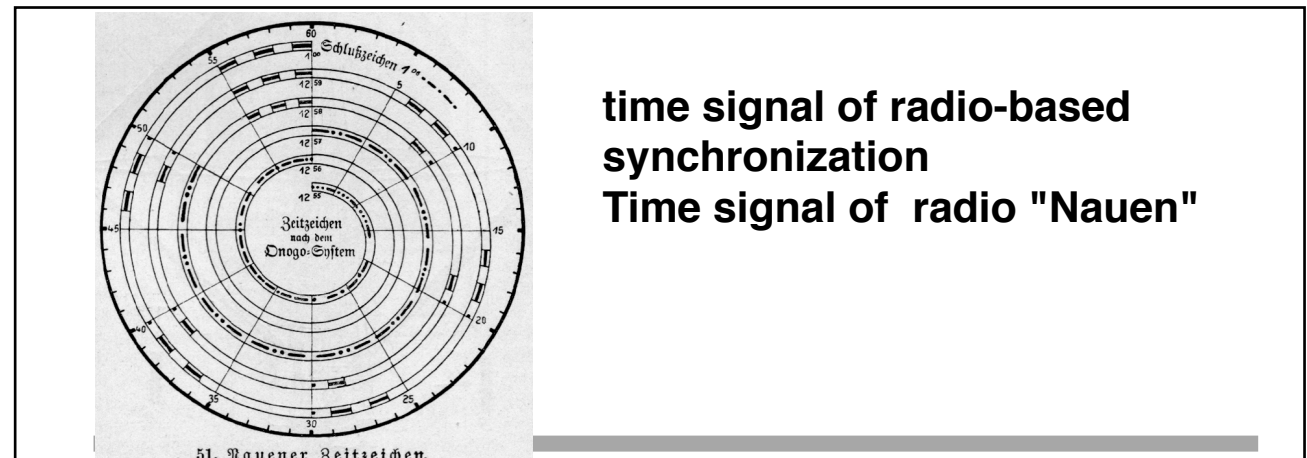


Fig. 340. Zeitball.

Embedded Networks 12



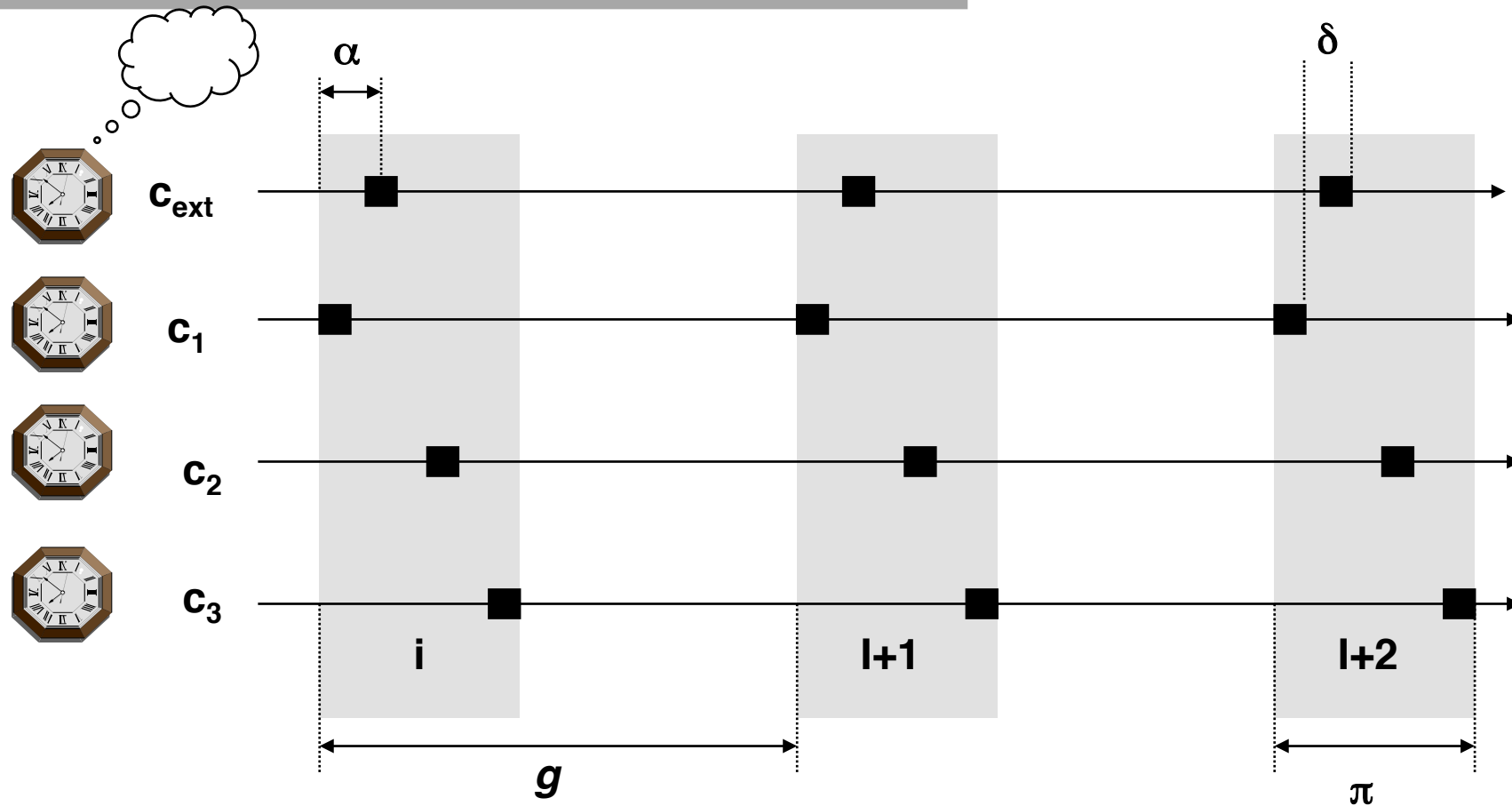
direct elektromagnetic synchronization  
Jones' Electro sympathetic Clock (TCD)



time signal of radio-based synchronization  
Time signal of radio "Nauen"



# Relation between the main distributed clock parameters



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



# 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  $\rho$ :

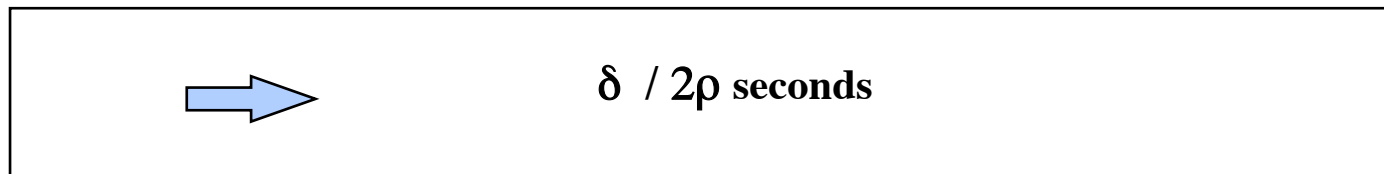
$$1 - \rho \leq dC/dt \leq 1 + \rho$$

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

difference of two un-synchronized clocks at  $\Delta t$  (which were in sync. at  $t$ ):

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

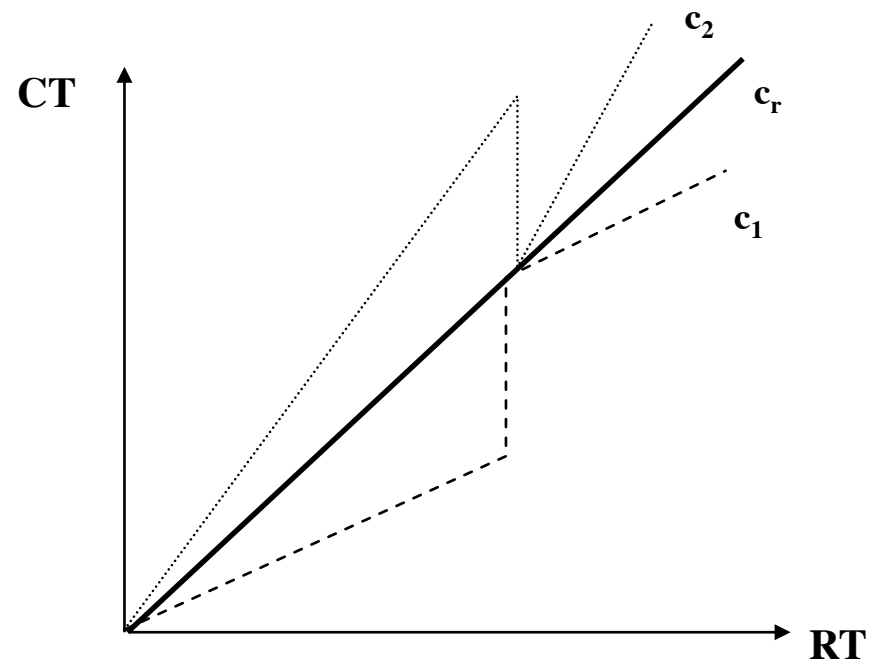
re-synchronization interval if two clocks should not deviate more than  $\delta$  seconds after



# How to synchronize ?

---

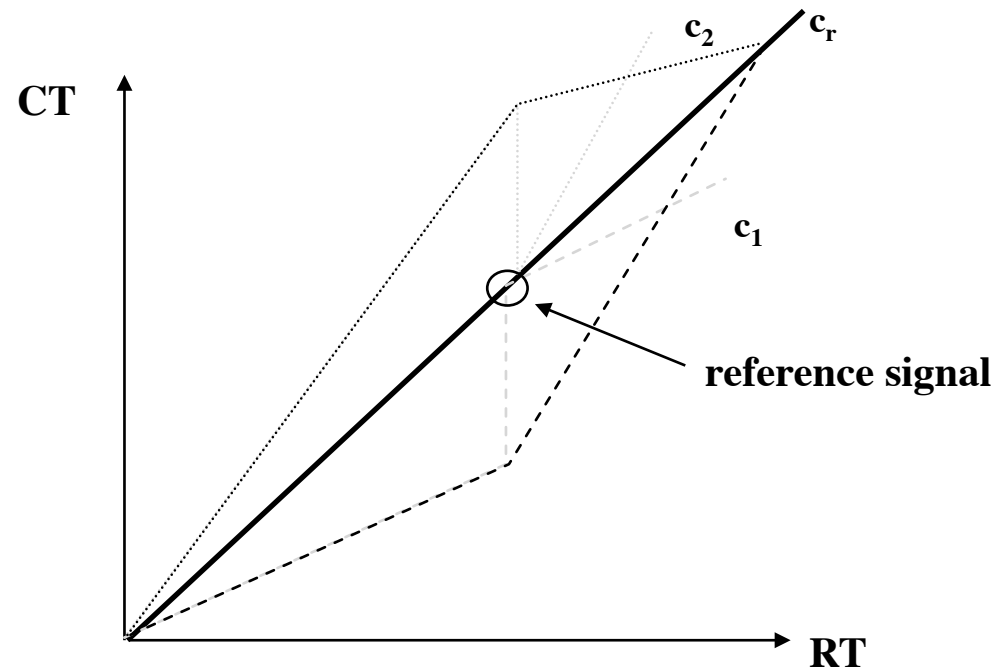
## Adapting clock values



# How to synchronize ?

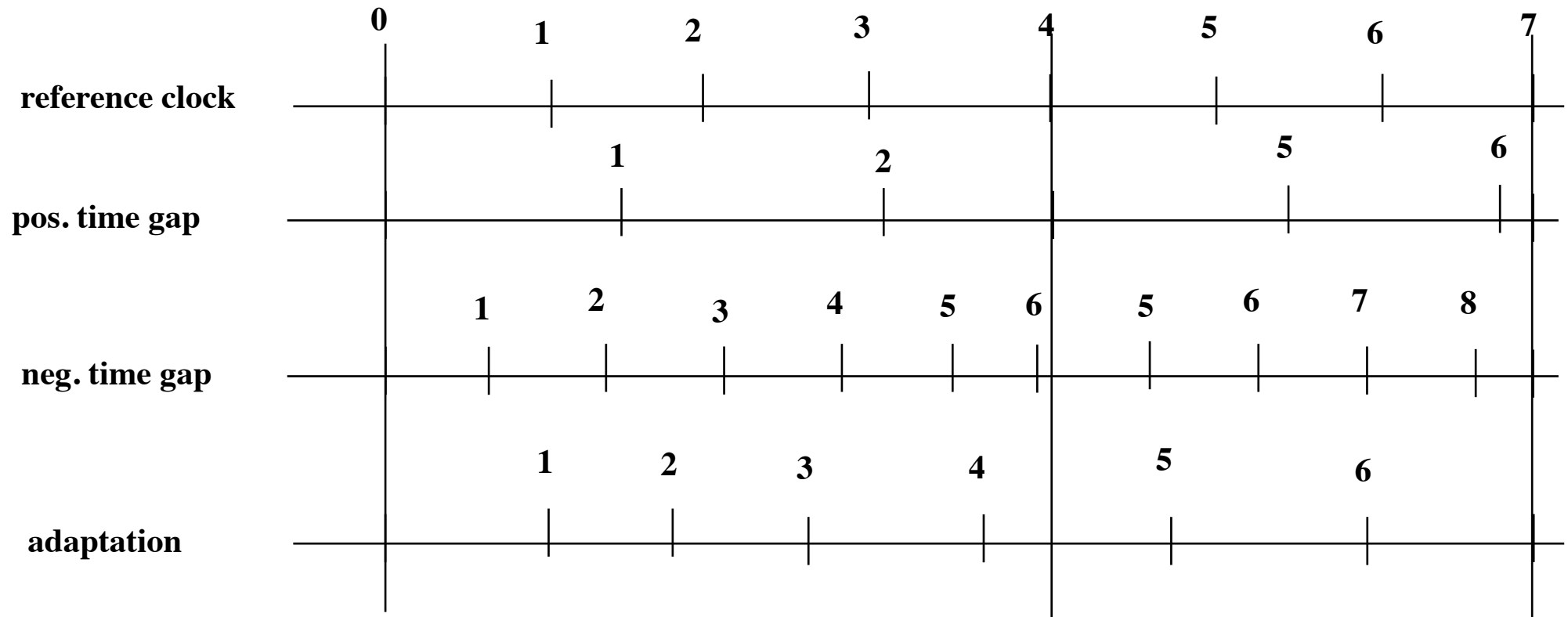
---

## Adapting clock rates



# How to synchronize ?

- counter correction
- drift correction



---

# Clock synchronization via messages in communication networks

- **Time server (external time sync.)**
- **Master- Slave synchronisation**
- **Decentralized and co-operative sync.**



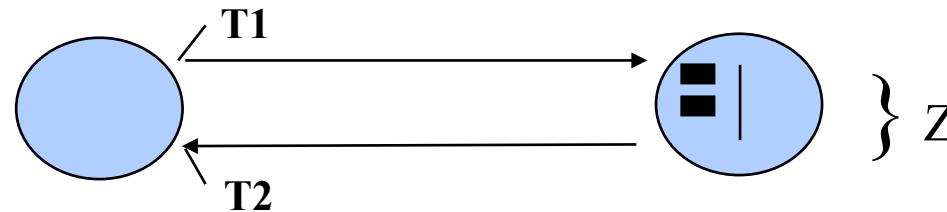
# Server-based clock synchronization

---

## Cristian's Algorithm:

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

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



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

Problems:

- Monotony properties of time
- Measurements are based on roundtrips ( $T2-T1$ ). A roundtrip may vary depending on network conditions.
  - > 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.



# Parameter for the calculation of time

---

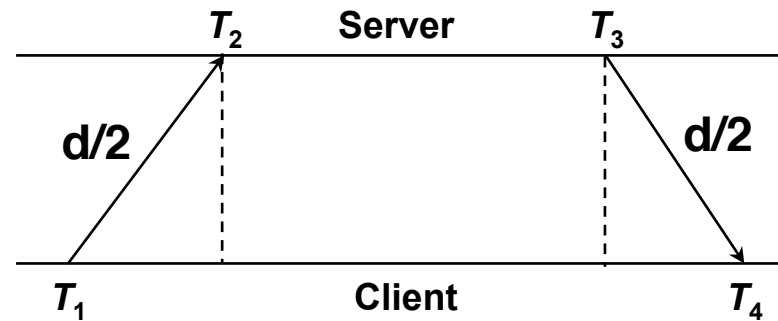
- ➔ **message delays in the communication network  
(steadyness + tightness)**
- ➔ **Time until the message is sent  
Delays in the Reply Queue**
- ➔ **Delay when reading the local clock  
Delay in the Request Queue**
- ➔ **Physical parameters of the server clock:  
Drift, Offset**



# Calculating offset and delay

---

Delay  $d$



$$\begin{aligned}d &= (T_4 - T_1) - (T_3 - T_2) \\ &= T_4 - T_1 - T_3 + T_2 \\ &= -T_1 + T_2 + T_4 - T_3 \\ &= T_2 - T_1 + T_4 - T_3\end{aligned}$$

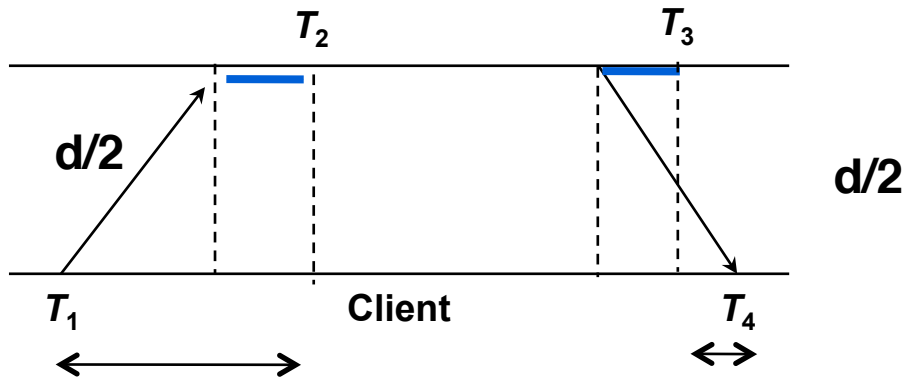




# Calculating offset and delay

## Offset $o$

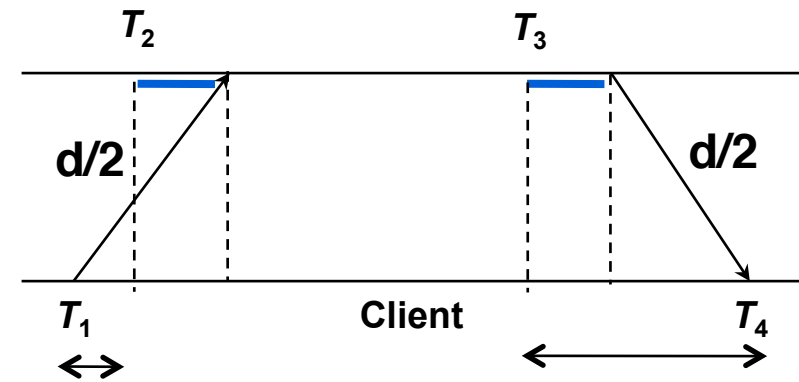
Case 1: Offset positive



$$o_R = T_2 - T_1 - d/2$$

$$o_T = T_3 - T_4 + d/2$$

Case 2: Offset negative



$$o_R = T_2 - T_1 - d/2$$

$$o_T = T_3 - T_4 + d/2$$

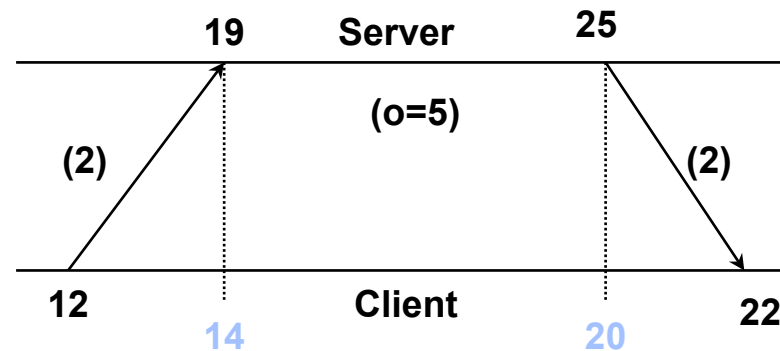
$$\begin{aligned} o &= [(T_2 - T_1) - d/2 + (T_3 - T_4) + d/2] / 2 \\ o &= [(T_2 - T_1) + (T_3 - T_4)] / 2 \\ &= [(T_2 - T_1) + (-T_4 + T_3)] / 2 \\ &= [(T_2 - T_1) - (T_4 - T_3)] / 2 \end{aligned}$$



# Calculating offset and delay

---

Offsets  $o$   
Delay  $d$



$$d = (22 - 12) - (25 - 19)$$
$$d = 4$$

$$o = [(19 - 12) + (25 - 22)] / 2$$
$$o = 5$$

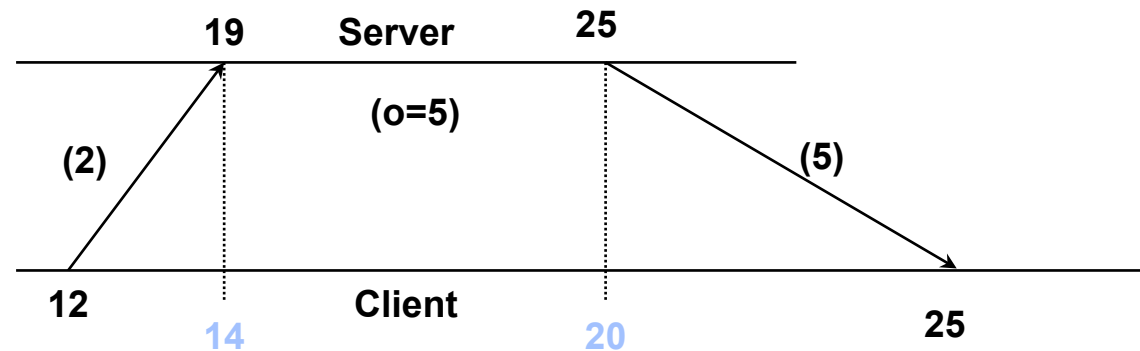


# Calculating offset and delay

---

Offsets:  $o$

Delay:  $d$



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

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



# IEEE1588 basic sync cycle (Precision Time Protocol PTP)

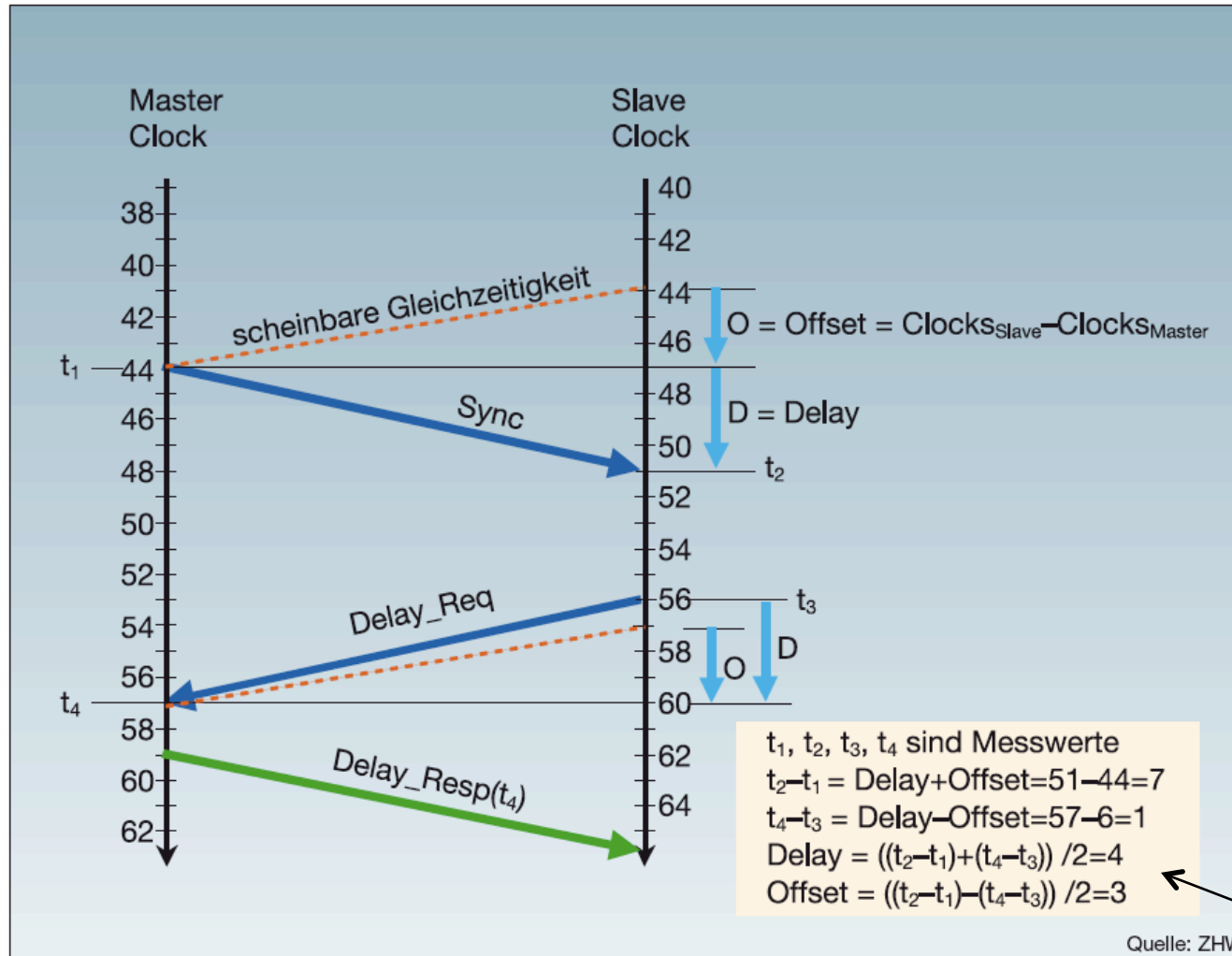
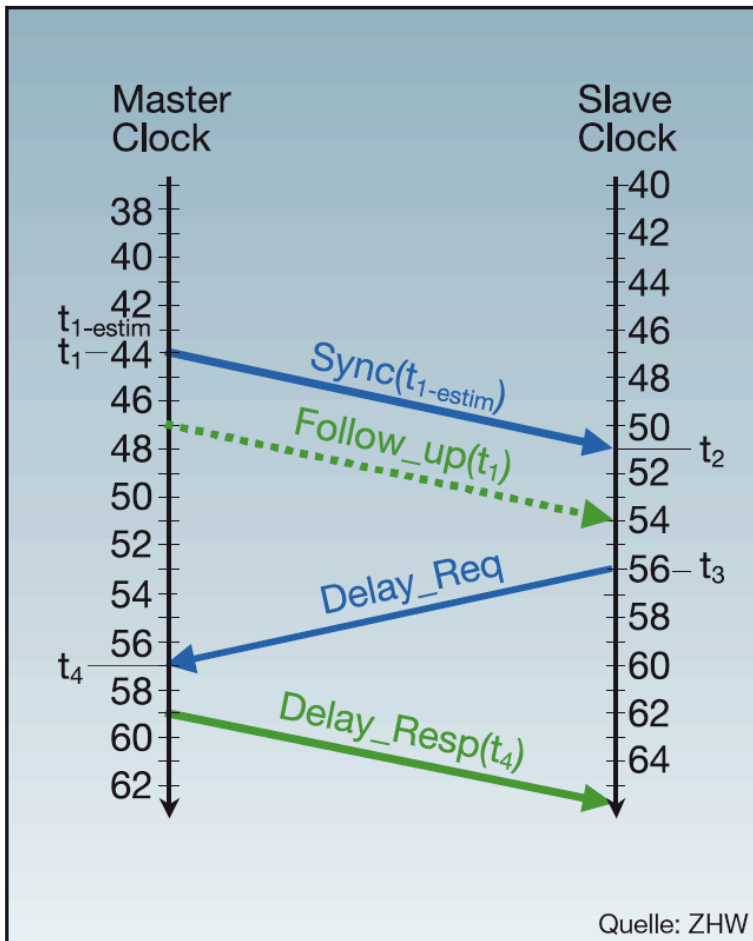


figure from:  
Hans Weibel, "Uhren mit IEEE  
1588 synchronisieren" Bulletin  
SEV/VSE 17/04

vgl. Folie 25





## Timestamping a sync message:

1. Hardware takes timestamp and immediately appends it to the message
2. Hardware (or Software) takes timestamp of the sending time and sends it in a separate message (follow up).

**Effect: Receiver obtains the exact sending time**

figure from:  
 Hans Weibel, "Uhren mit IEEE  
 1588 synchronisieren" Bulletin  
 SEV/VSE 17/04



## Handling delays in LANs with routers and switches in IEEE1588

---

**Traffic can cause delays with substantial variations in switches  
(mean delay  $25\mu\text{s}$ , Std. deviation  $82\mu\text{s}$ )**

**Transparent clocks compensating variation by measuring the delay locally  
and correcting clock value**

**Transparent clocks enable slave sync accuracy to master similar to that of  
a crossover cable between master and slave**

(Paul Skoog: Using IEEE-1588 transparent clocks to improve system time synchronization accuracy, Embedded.com)



# Handling delays in LANs with routers and switches in IEEE1588

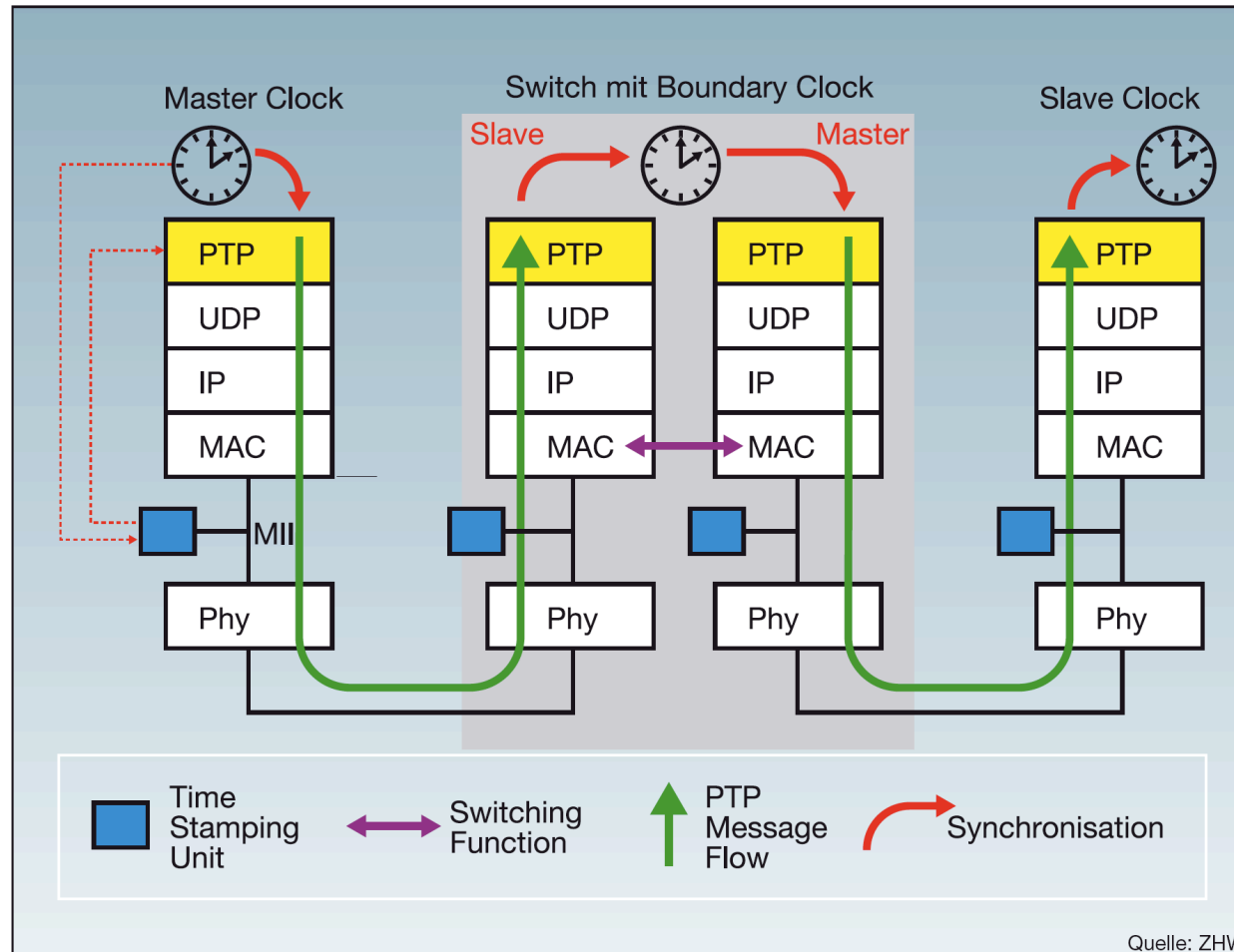
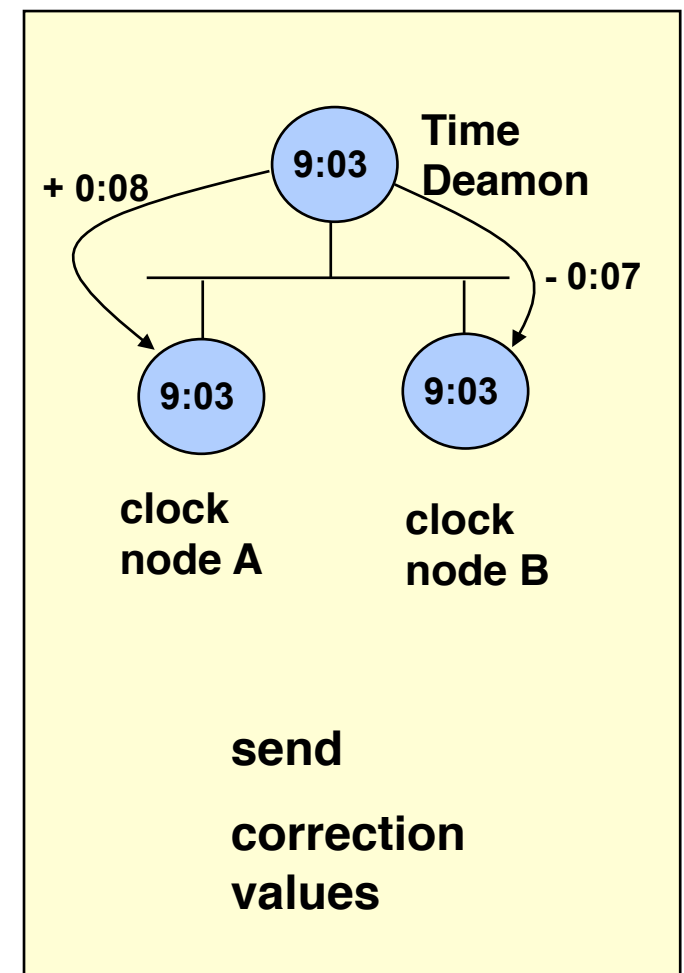
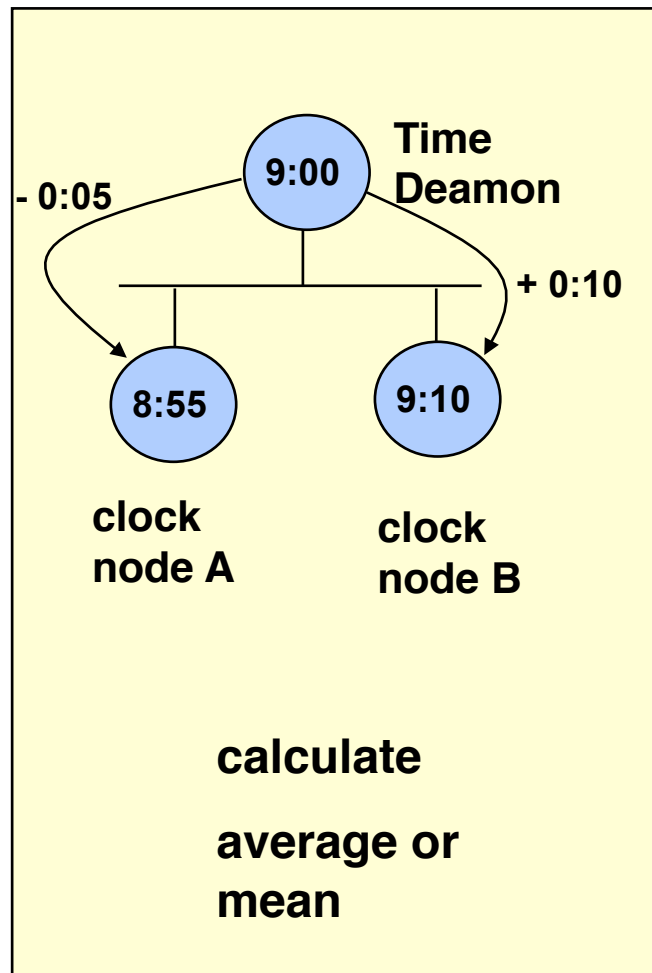
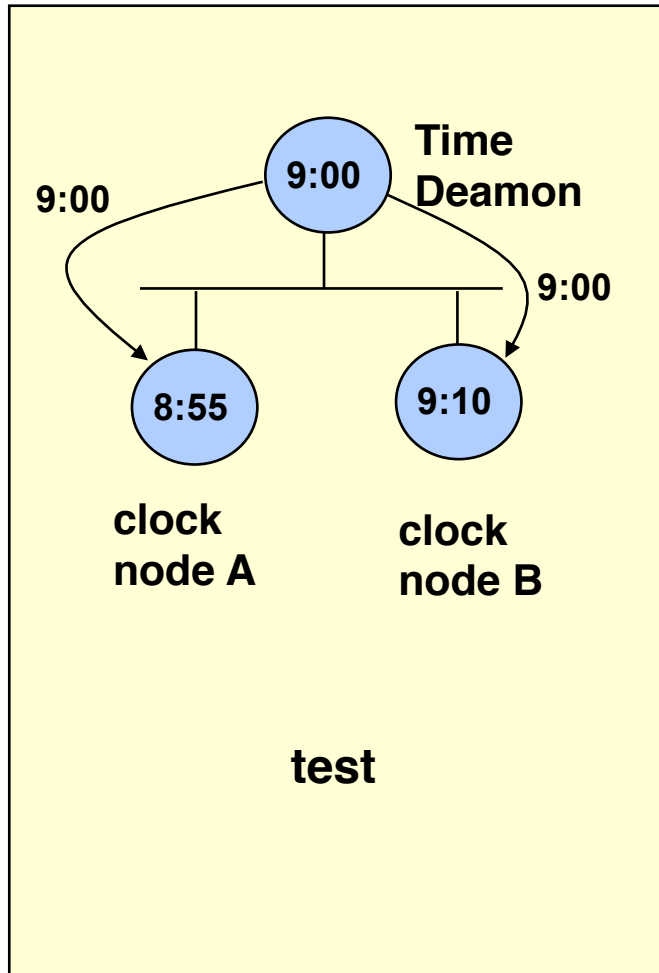


figure from:  
Hans Weibel, "Uhren mit IEEE  
1588 synchronisieren" Bulletin  
SEV/VSE 17/04



# The Berkeley Algorithm

R. Gusella, S. Zati: "The accuracy of the clock synchronization achieved by TEMPO in Berkeley Unix 4.3 BSD." IEEE Trans. Softw. Eng., (15)7, July 1989





# 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

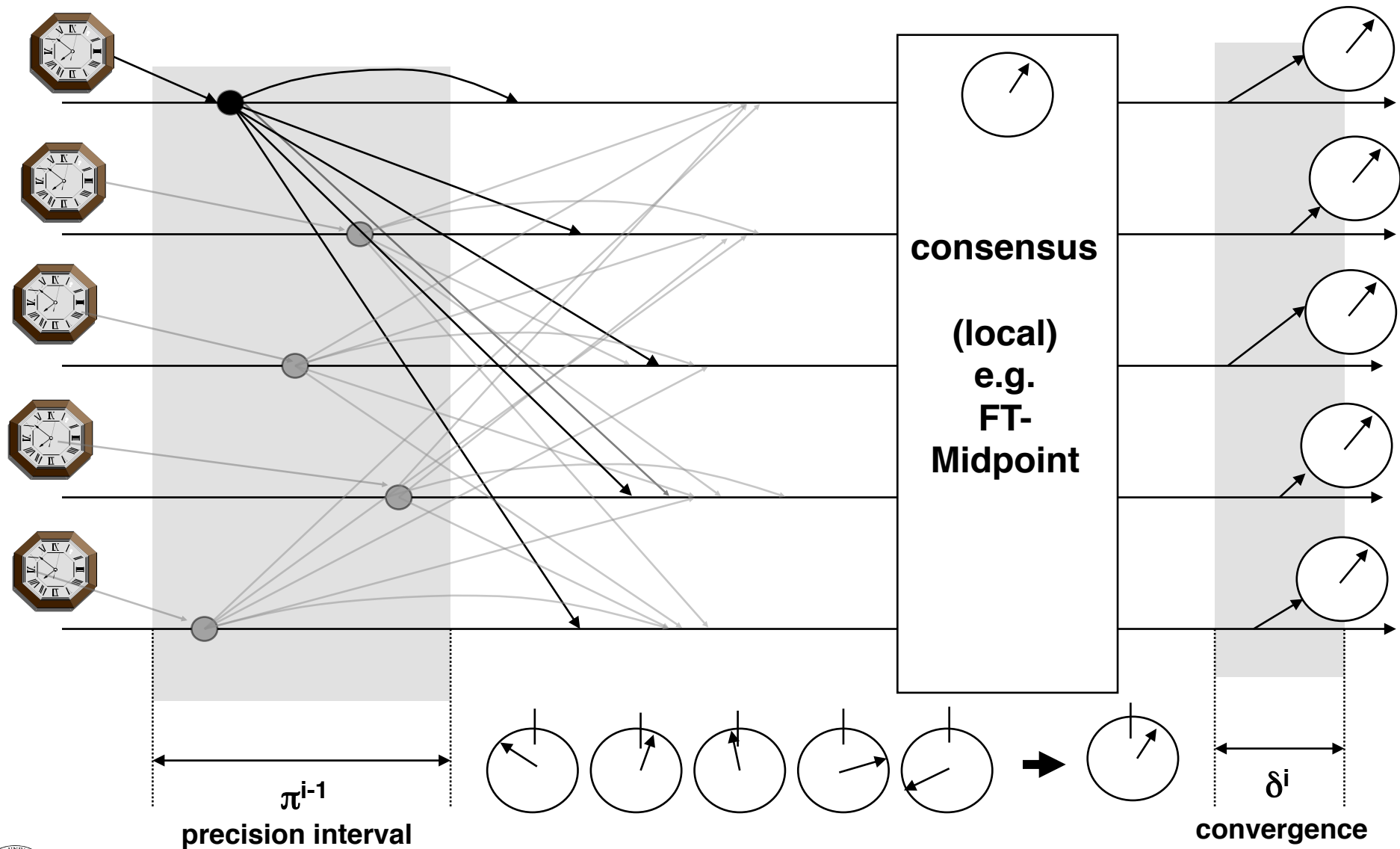
## Hybrid:

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  $k+1$  and  $n-k$ :

- **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.



# **Characteristics of Averaging:**

---

**The protocol is peer-to-peer**

**The protocol provides internal synchronization**

**An arbitrary node initiates the protocol**

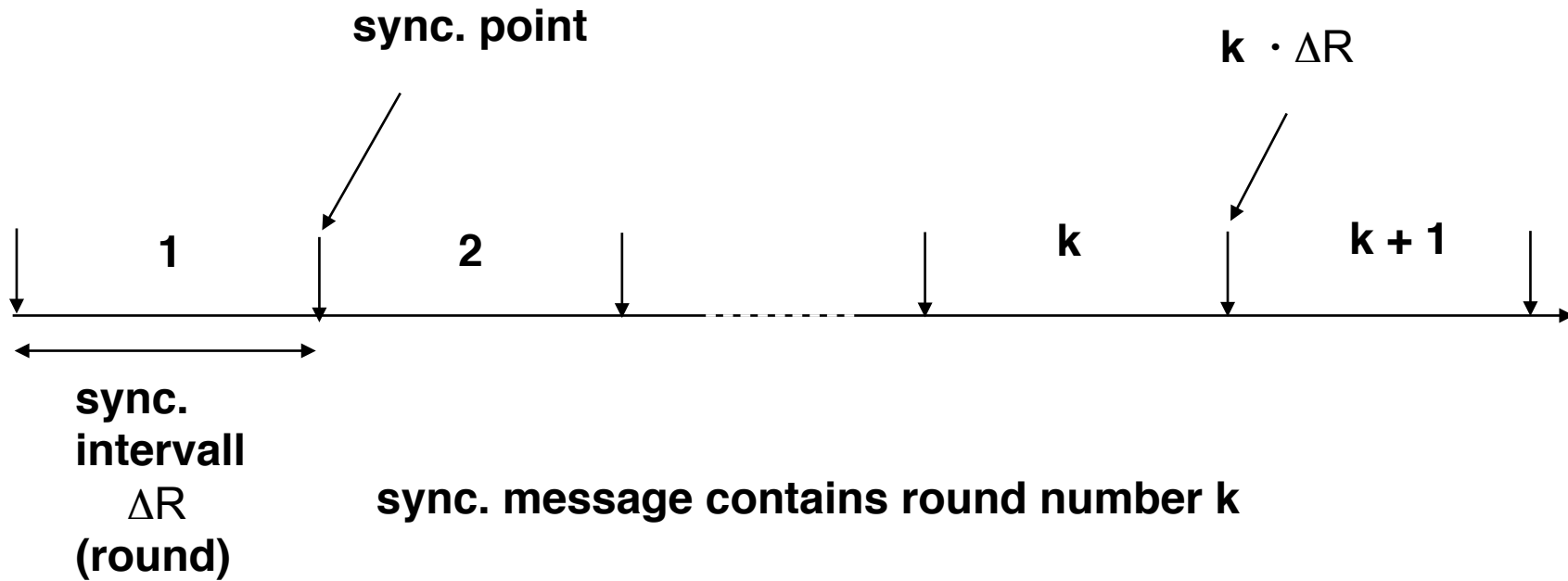
**The protocol may be initiated at an arbitrary point in time**

**Offset is calculated to adjust time or rate**

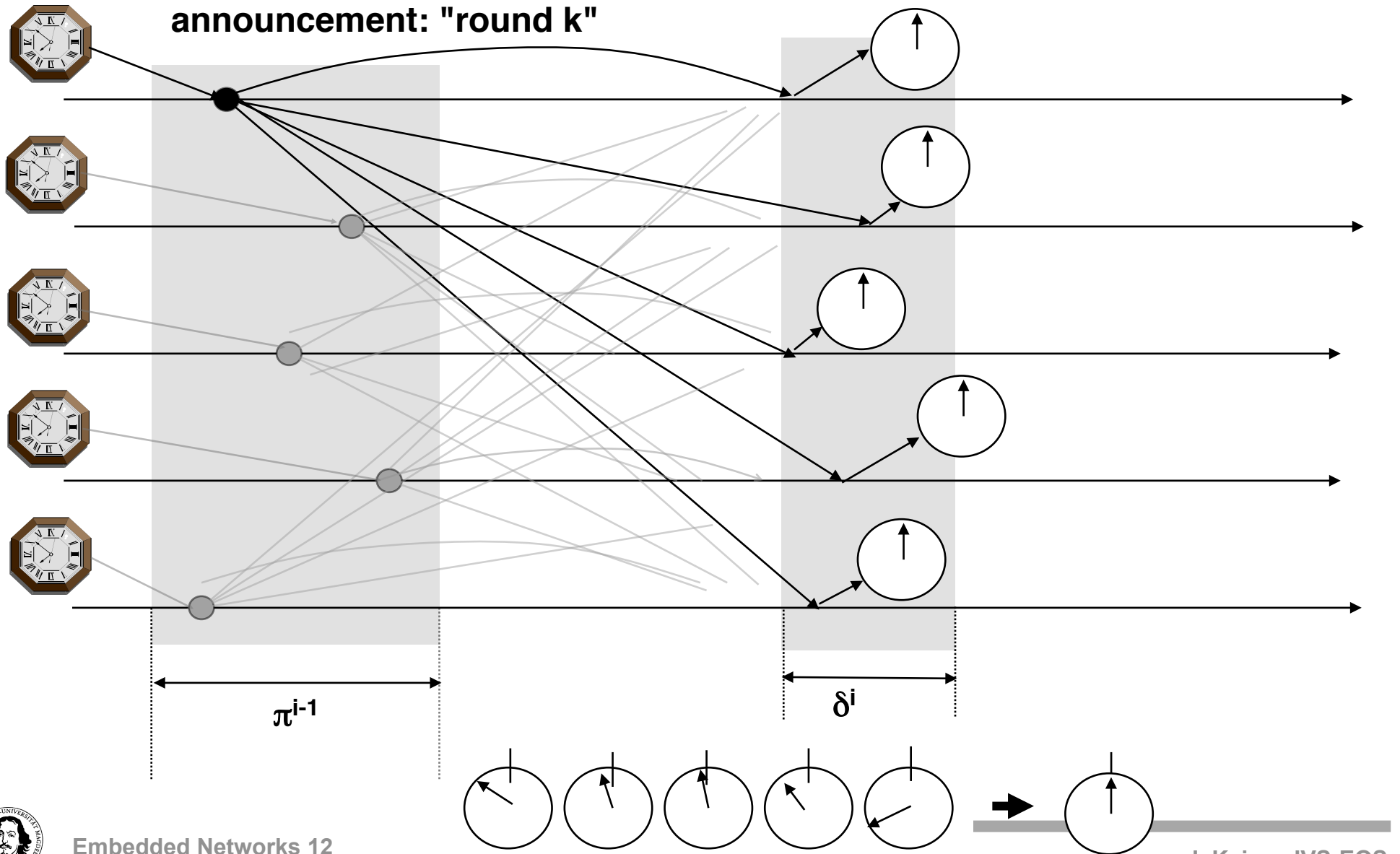
**Synchronization to external time is possible**



# Non-averaging synchronization



# Non-averaging synchronization



# **Characteristics of non-averaging synchronization**

**The protocol is peer-to-peer**

**The protocol provides internal synchronization**

**An arbitrary node initiates the protocol**

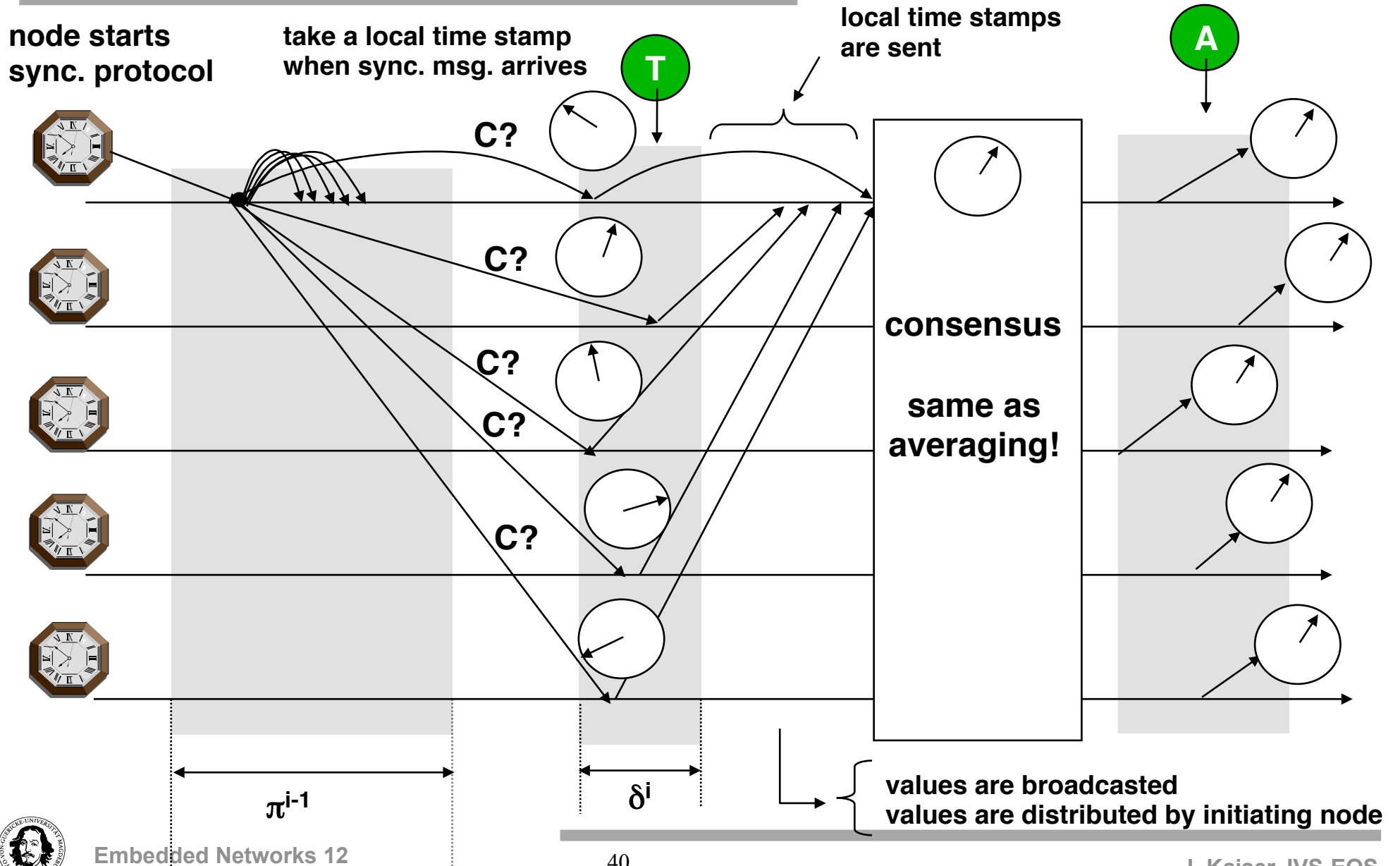
**The protocol is initiated within a defined time interval**

**No offset is calculated, clock is set to a predefined (future) time point**

**Synchronization with external time difficult!**

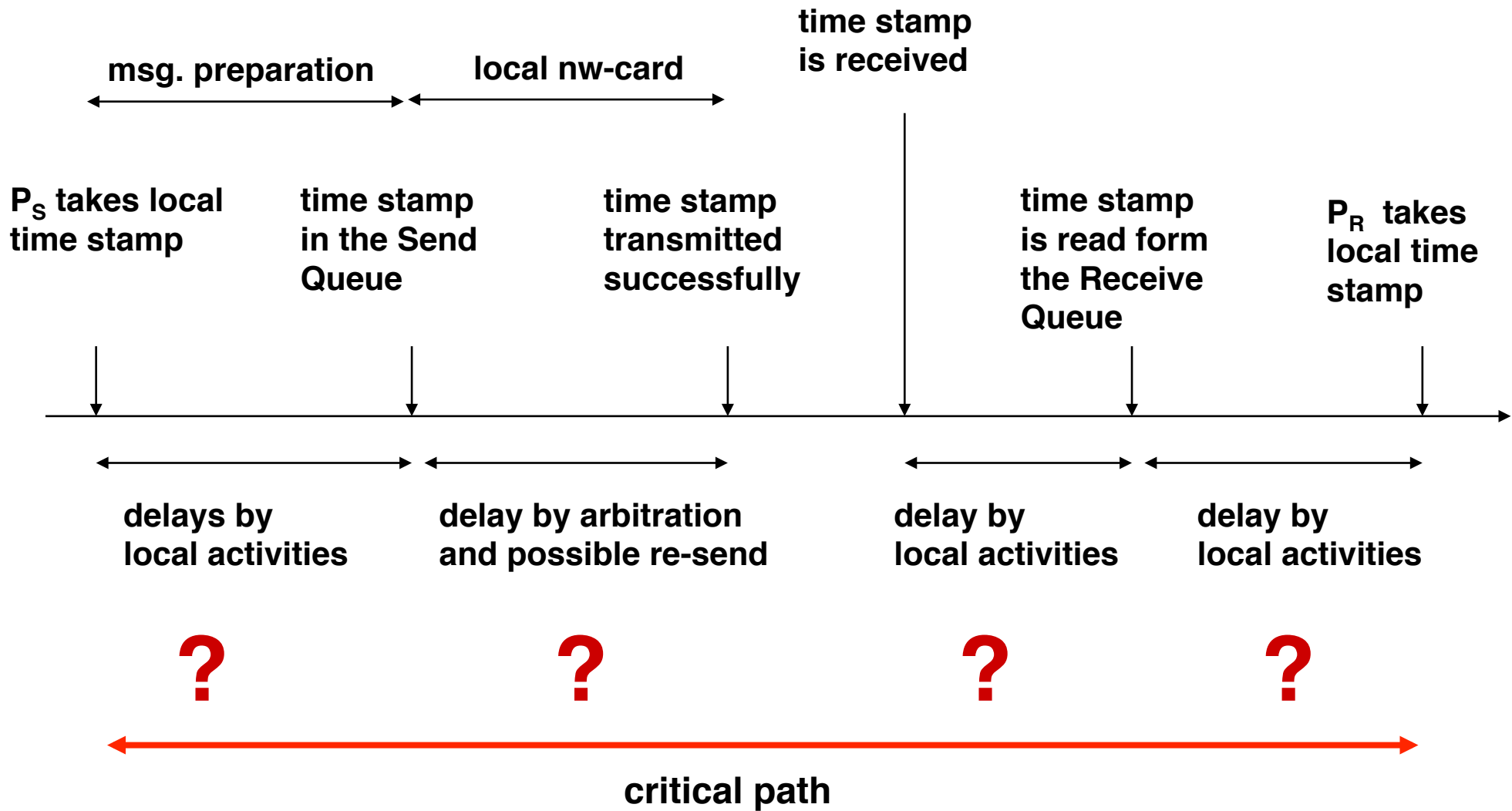


# Hybrid synchronization



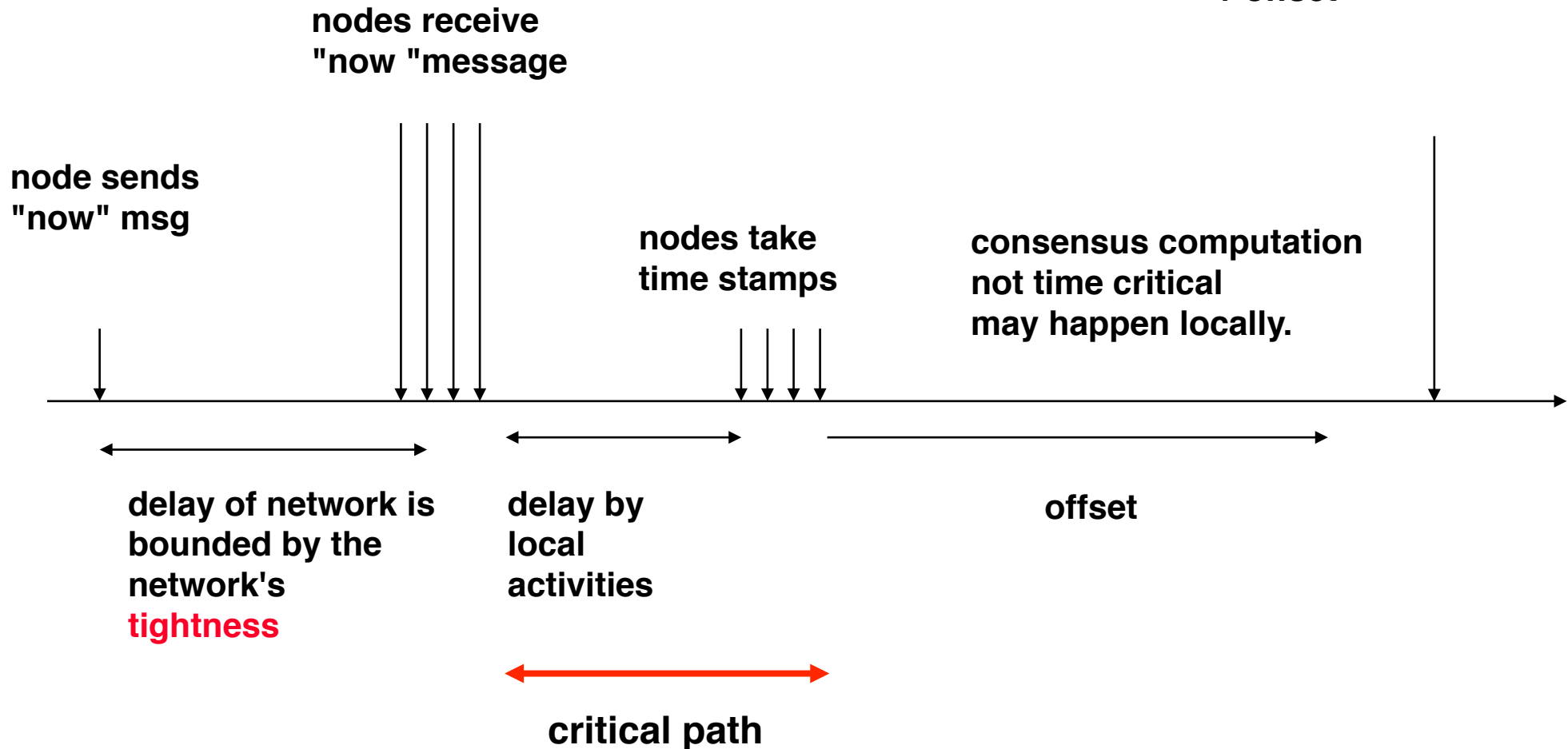


# Practical Considerations:



# Practical Considerations:

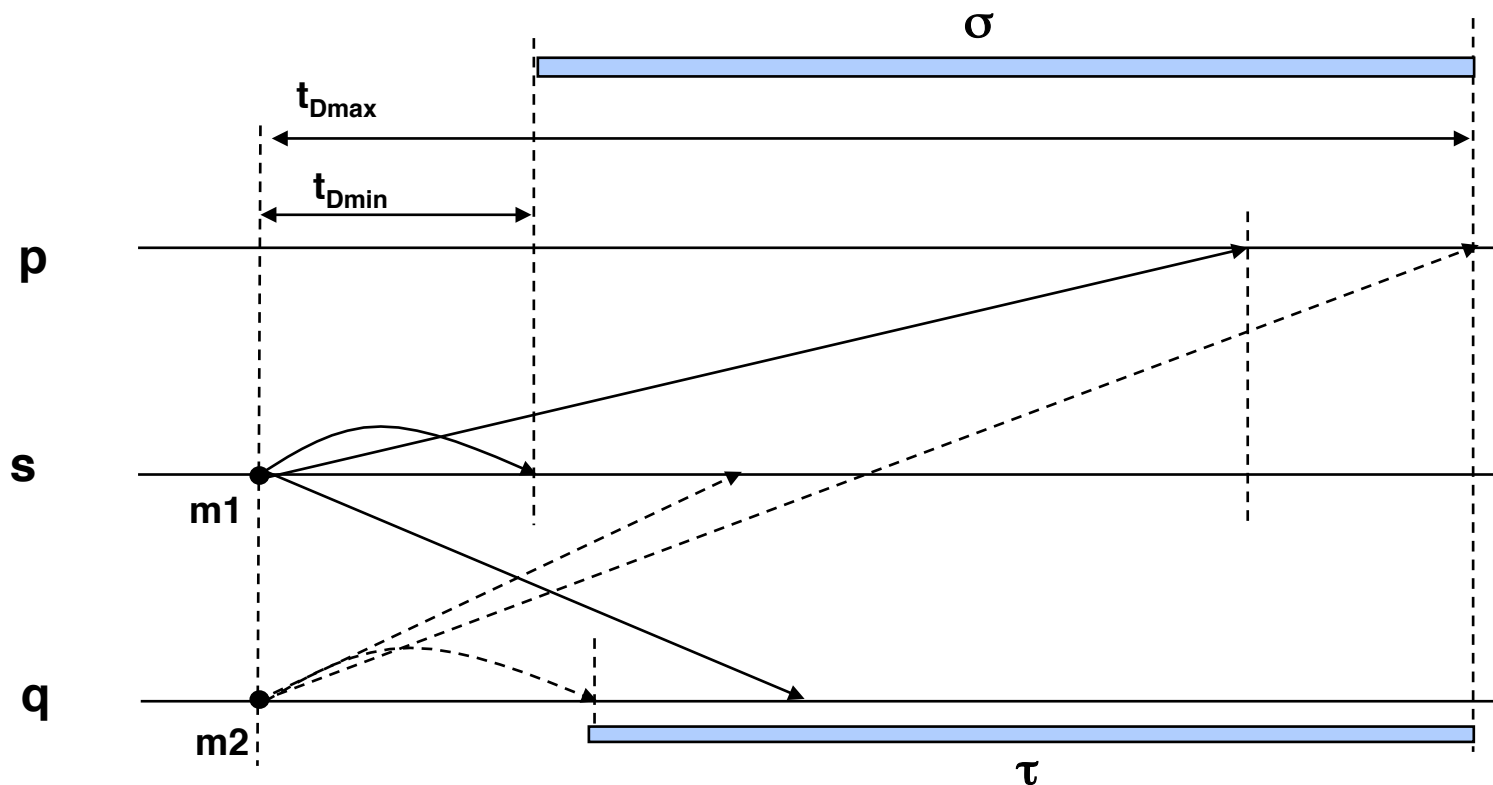
nodes set new time:  
 $t = \text{corrected time stamp} + \text{offset}$



# Remember: Steadiness and Tightness

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

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



# **Characteristics of hybrid synchronization**

**The protocol is peer-to-peer**

**The protocol provides internal synchronization**

**An arbitrary node may initiate the protocol**

**The protocol is initiated within a defined time interval**

**An offset is calculated and distributed**

**The critical path is reduced**

**Synchronization to external time is possible**



## Example: 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:

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

\*

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

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

$t_{1,j} - t_{1,1} - md \cdot N$

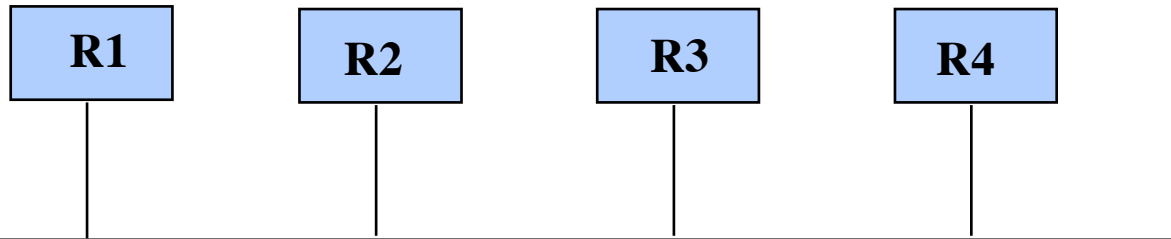
For every element  $k_{i,j}$  :

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



# Co-operating clock synchronization

Example:



**md=6**

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

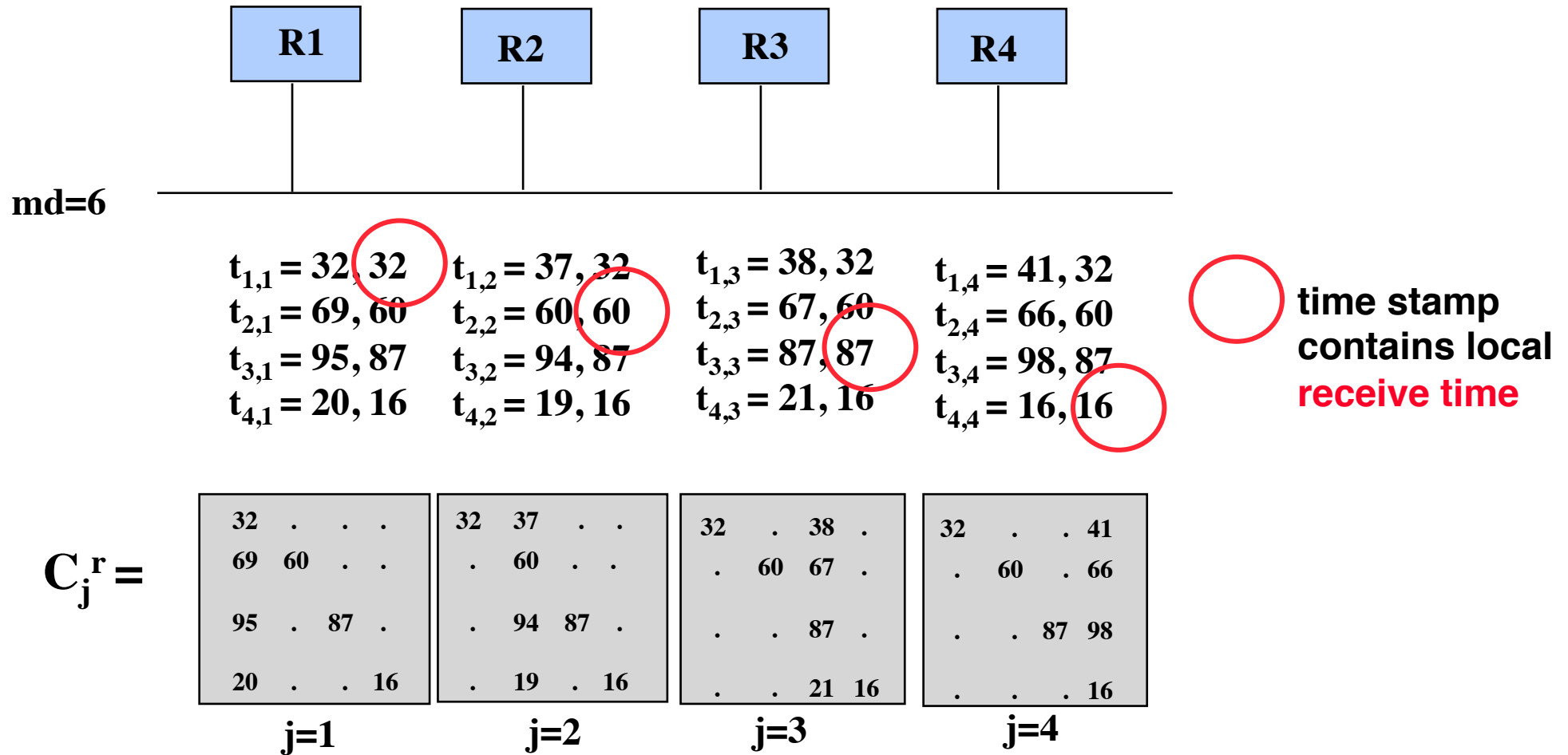
$C_j^r =$

32 . . .	32 37 . .	32 . 38 .	32 . . 41
69 60 . .	. 60 . .	. 60 67 .	. 60 . 66
95 . 87 .	. 94 87 .	. . 87 .	. . 87 98
20 . . 16	. 19 . 16	. . 21 16	. . . 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 37 . .	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

$j = 1$

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

$j = 2$

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

$j = 3$

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

$j = 4$

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

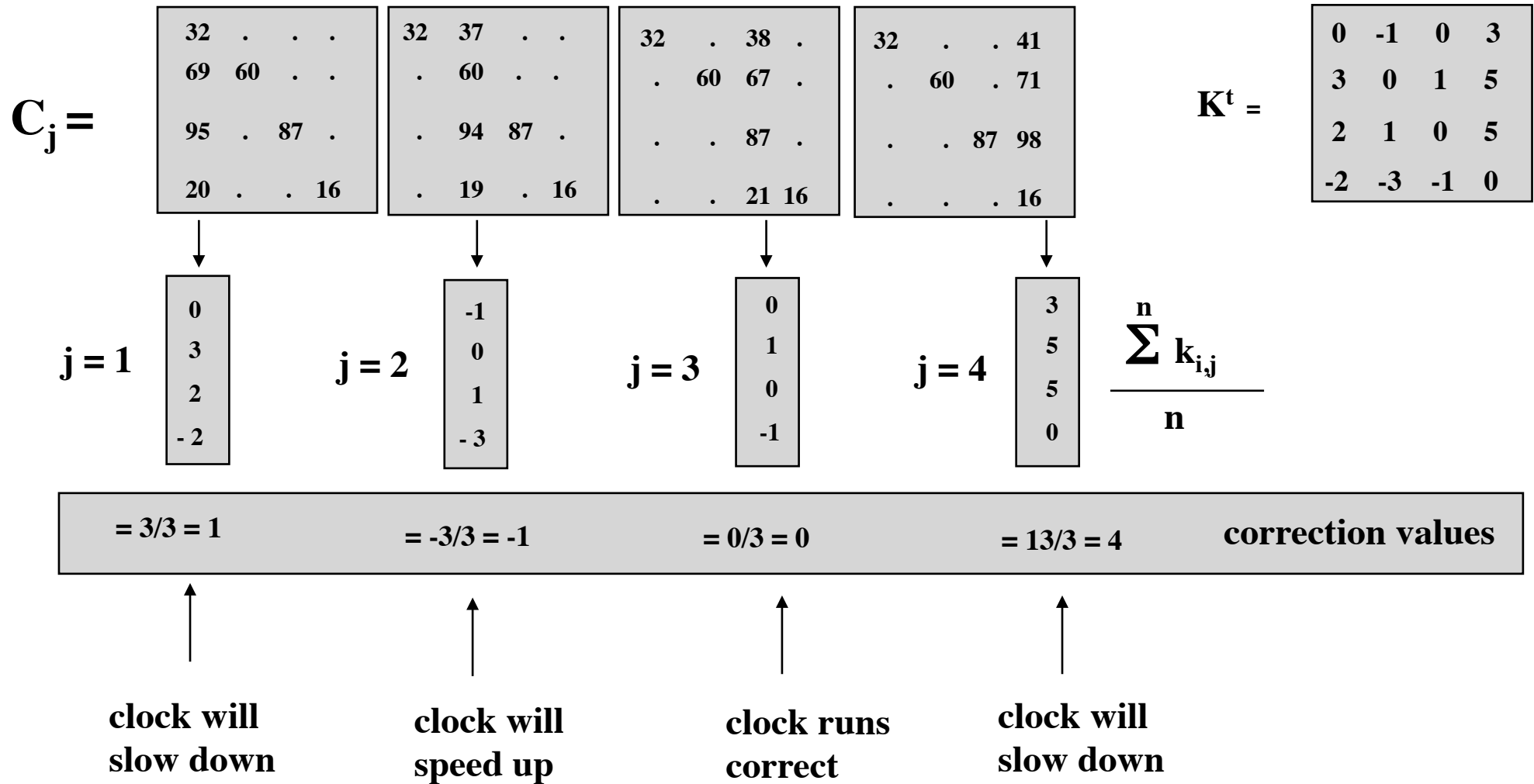
$K^t =$

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



# Co-operating clock synchronization

Example:



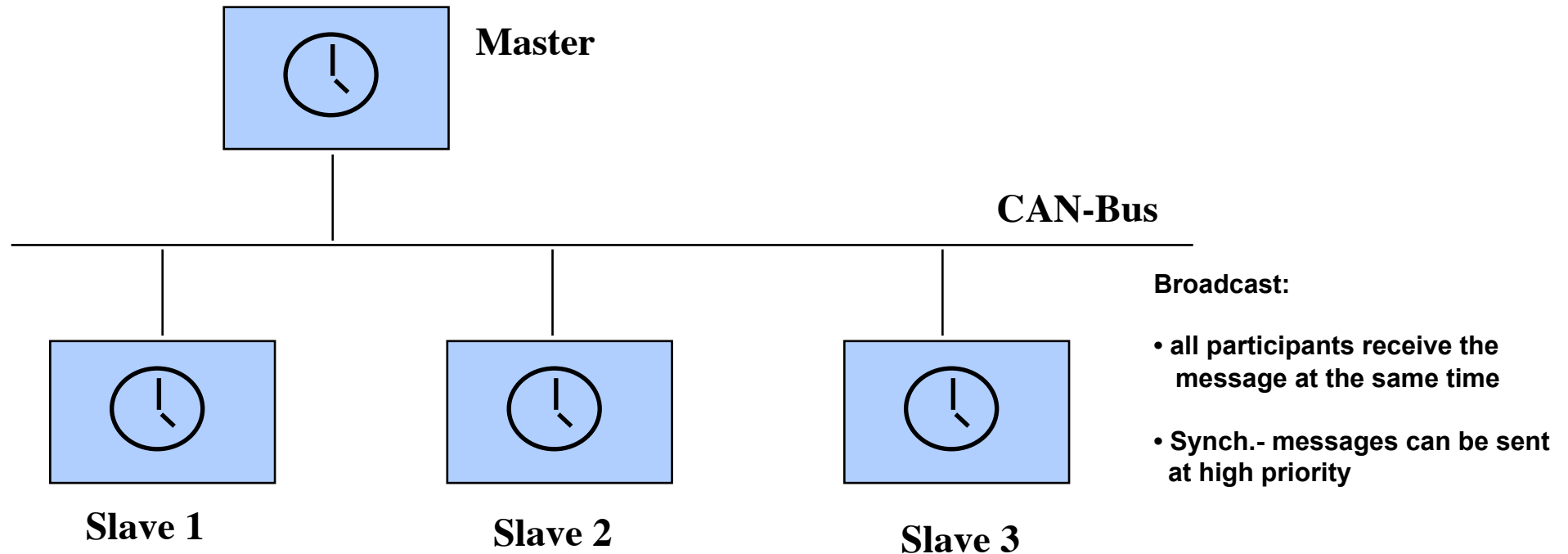
---

# Exploiting Broadcast Networks



# Clock synchronization on the CAN-Bus

M. Gergeleit and H. Streich. Implementing a distributed high-resolution real-time clock using the CAN-bus. In 1st International CAN Conference, 1994.



## System:

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

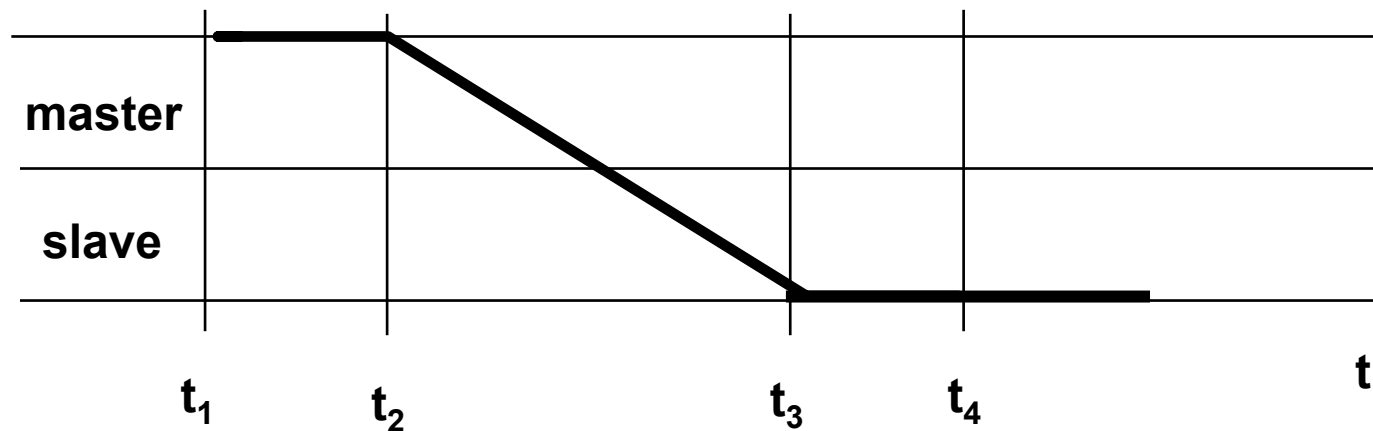


# Clock synchronization on the CAN-Bus

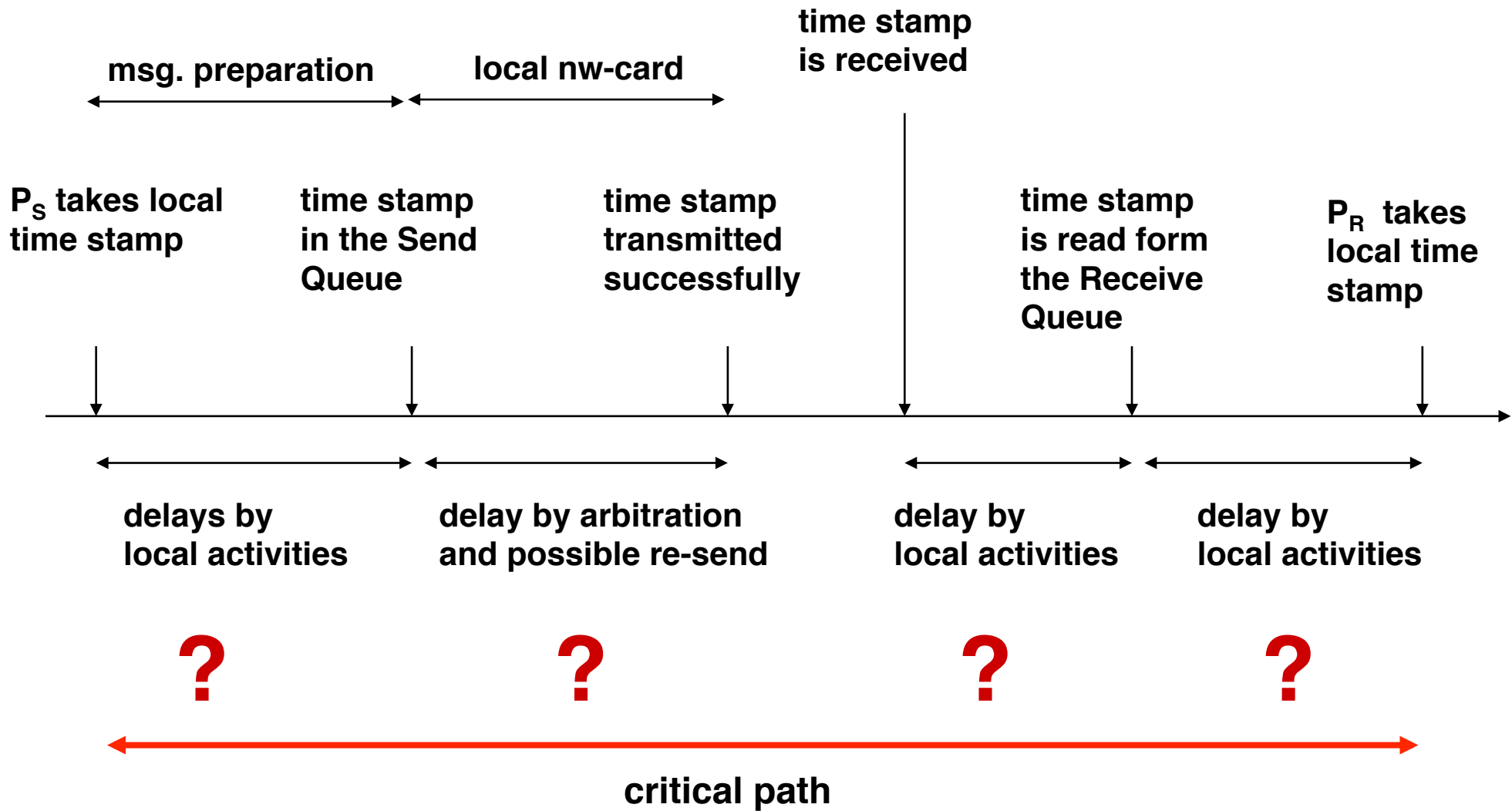
---

## Simple protocol

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



# Practical Considerations:

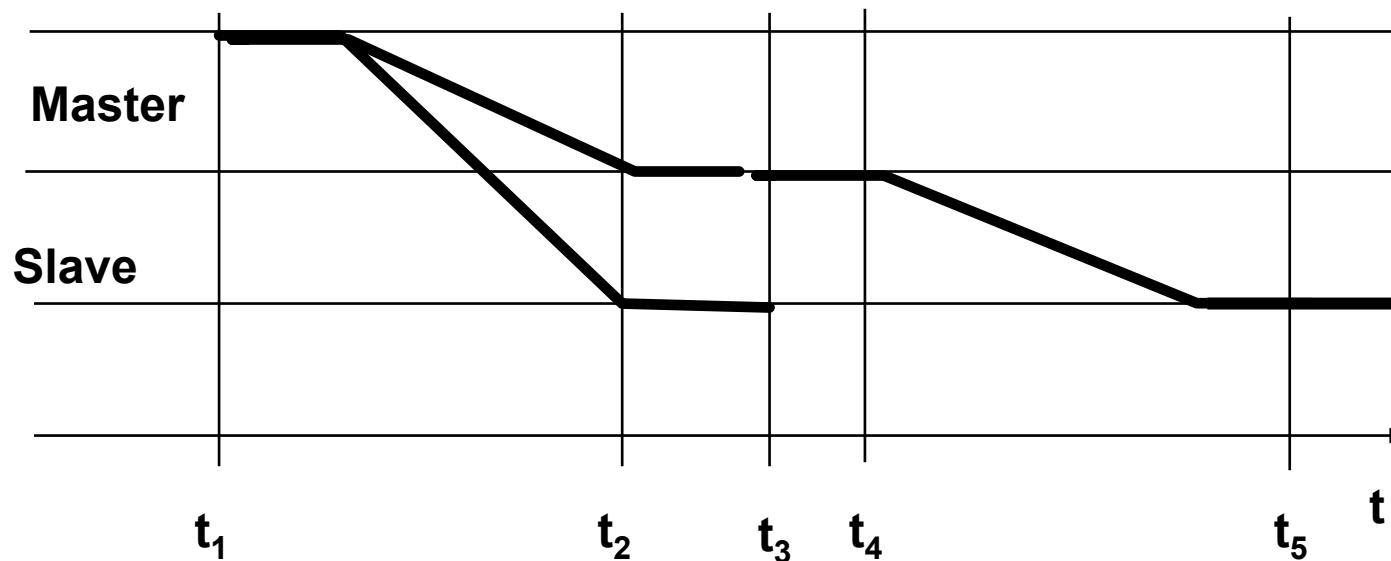


# Clock synchronization on the CAN-Bus

---

## 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 ( $t_5$ )

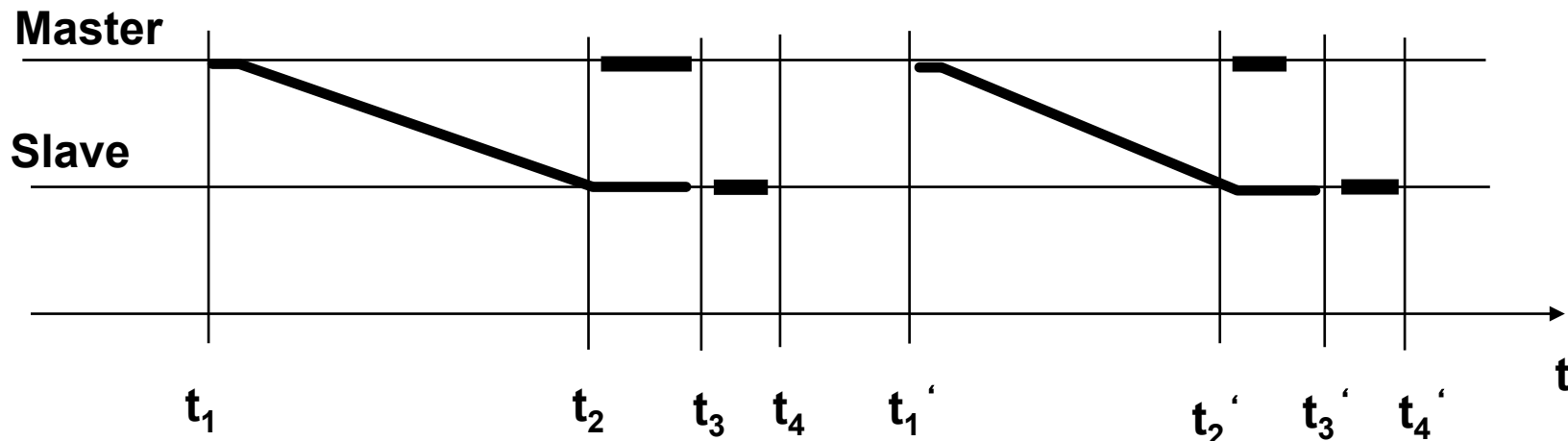


# Clock synchronization on the CAN-Bus\*

---

## Optimized protocol

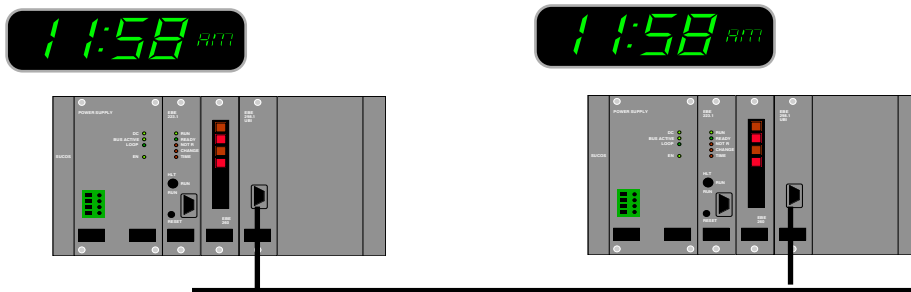
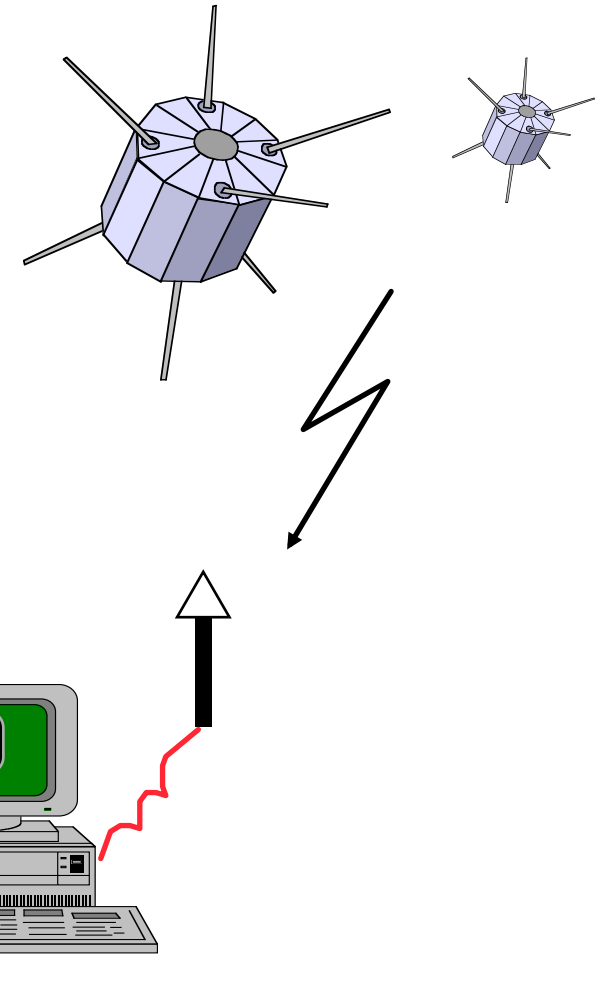
- master sends message "now" ( $t_1$ )
- this message contains the time stamp of the last synchronization round
- master reads its clock on the receive interrupt of the message ( $t_3$ )
- slaves read their clocks on the receive interrupt of the message ( $t_2$ )
- slaves adjust their clocks according 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 is easy because of Master-Slave concept
- no additional cost
  - implemented on embedded controller
- Application areas:
  - co-operative data acquisition
  - co-operative control



# **Characteristics of Gergeleit/Streich**

**The protocol is master-slave**

**The protocol provides internal and external synchronization**

**The master node initiates the protocol periodically**

**The critical path is reduced drastically**

**Optimized solution uses traffic to minimum**



# Clock Synchronization for Wireless Networks

- ➔ **Wireless networks inherently broadcast in a single hop environment**
- ➔ **Energy efficiency is a major concern**  
With the energy to transport 1 Bit over 100m (3 Joule) a modern Microcontroller can execute 3 Million instructions.
- ➔ **Restricted performance of nodes**  
AVR, MSP430: max. 16 MIPS @16 Mhz,
- ➔ **Restricted bandwidth of network**  
10-100 kps



# Sync. on Wireless Networks

---

**Assumption: Network provides tightness, i.e. all receivers receive the sync. msg. at approx. the same time.**

M. Mock, R. Frings, E. Nett, S. Trikaliotis:  
"Continuous Clock Synchronization in  
Wireless Real-Time Applications", Proc.  
19<sup>th</sup> IEEE Symp. on Reliable Distributed  
Systems (SRDS-00), Nürnberg, Germany,  
Oct. 2000

**Basic idea follows Gergeleit/Streich's CAN sync:**

- ➔ Synchronization is organized in rounds**
- ➔ Some master broadcasts the master time stamp of the last round**
- ➔ Each node including the master records its reception time stamp**
- ➔ The nodes correct their clocks using their local TS from the last round**

**The sync protocol improves the IEEE 802.11 time synchronization protocol.**



# **More about synchronization**

---

## **Clock correction:**

**Instead of correcting the clock, a table of correction values is maintained. Every local TS is corrected according to the appropriate value**

## **Sync-by-demand (post facto):**

**No continuous synchronization is performed. When clocks need to synchronize, the sync- procedure is triggered. Example: Reference broadcast.**

## **Latency measurement:**

**If sync is only needed between a source and a destination to estimate the age of an information, e.g. a sensor value, only latency measurement is applied.**



# Reference Broadcast Synchronization (RBS)

---

**Assumption: Network provides tightness, i.e. all receivers receive the msg at approx. the same time.**

J. Elson, L. Girod, D. Estrin: "Fine-Grained Network Time Synchronization using Reference Broadcasts", Proc. 5<sup>th</sup> Symp. on OS Design and Impl. (OSDI 2002)

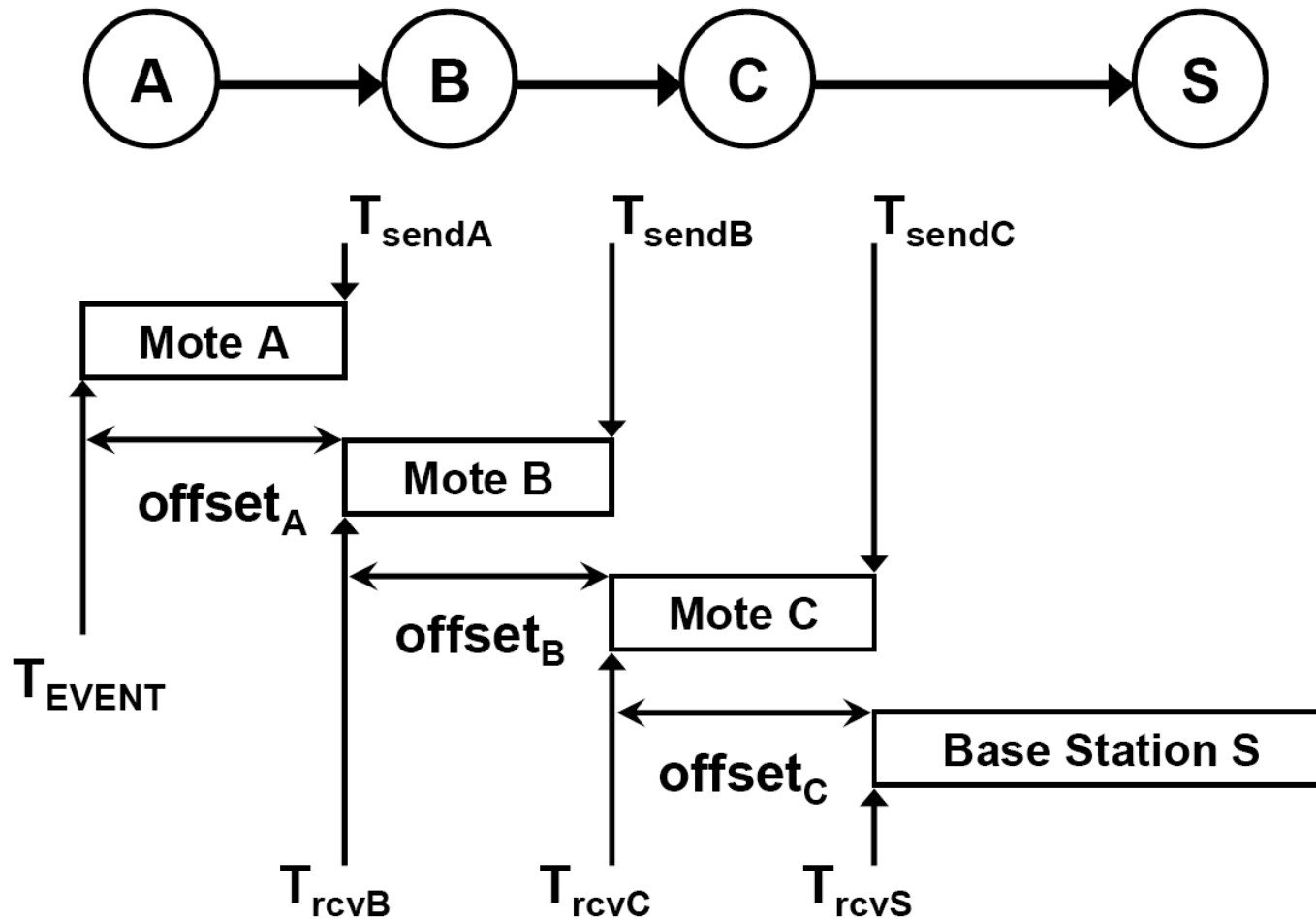
**Basic idea:**

- ➔ **Some node broadcasts a reference packet to 2 nodes**
- ➔ **Each node records its reception time stamp**
- ➔ **The nodes exchange the recorded time stamps**
- ➔ **Clock offsets can be computed with some averaging algorithm**

**Can be generalized to more than 2 nodes, however, traffic grows  $O(n^2)$ .**



# Measuring the age of information



Gyula Simon, Miklós Maróti, Ákos Lédecz, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, Ken Frampton:  
Sensor Network-Based Countersniper System, SenSys'04, November 3–5, 2004, Baltimore, Maryland, USA



## Fragen zur Uhrensynchronisation

---

**Was ist das grundsätzlich Problem bei der Verschickung und dem Vergleich von Zeitstempeln?**

**Welche Charakteristik des Netzes ist für die Genauigkeit der Uhrensynchronisation entscheidend?**

**Warum werden im IEEE 1588 Precision Time Protocol Sync-Nachrichten verschickt? Wie wird das gemacht?**

**Wie werden im IEEE 1588 Precision Time Protocol die Laufzeiten durch die Switches/Router ausgeglichen? Was ist die implizite Annahme dabei?**

**Warum ein Hybrides Sync-Protokoll?**

**Was bewirkt die Genauigkeit des CAN Sync Protokolls und was die Effizienz?**





---

In [ST87], the logical clock is corrected with value  $k \Delta R + \pi$ , where  $k$  is the round number,  $\Delta R$  is the round duration and  $\pi$  is a constant.  $\pi$  should be large enough to avoid clocks to be set backwards. In [CAS86, HSSD84], the logical clock is simply corrected with  $k \Delta R$ . In [VCR97], it is corrected with the value of one of the clock estimates contained in the clock estimation message.

[ST87] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626645, July 1987.

[CAS86] 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.

[HSSD84] J. Y. Halpern, H. R. Strong, B. B. Simons, and D. Dolev. Fault-tolerant clock synchronization. In *Proc. of 3rd International Symposium on Principles of Distributed Computing*, pages 89102, 1984.

[VCR97] 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.

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

