Global Time and Clock Synchronization in Networks

"Milestone" papers concernng 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

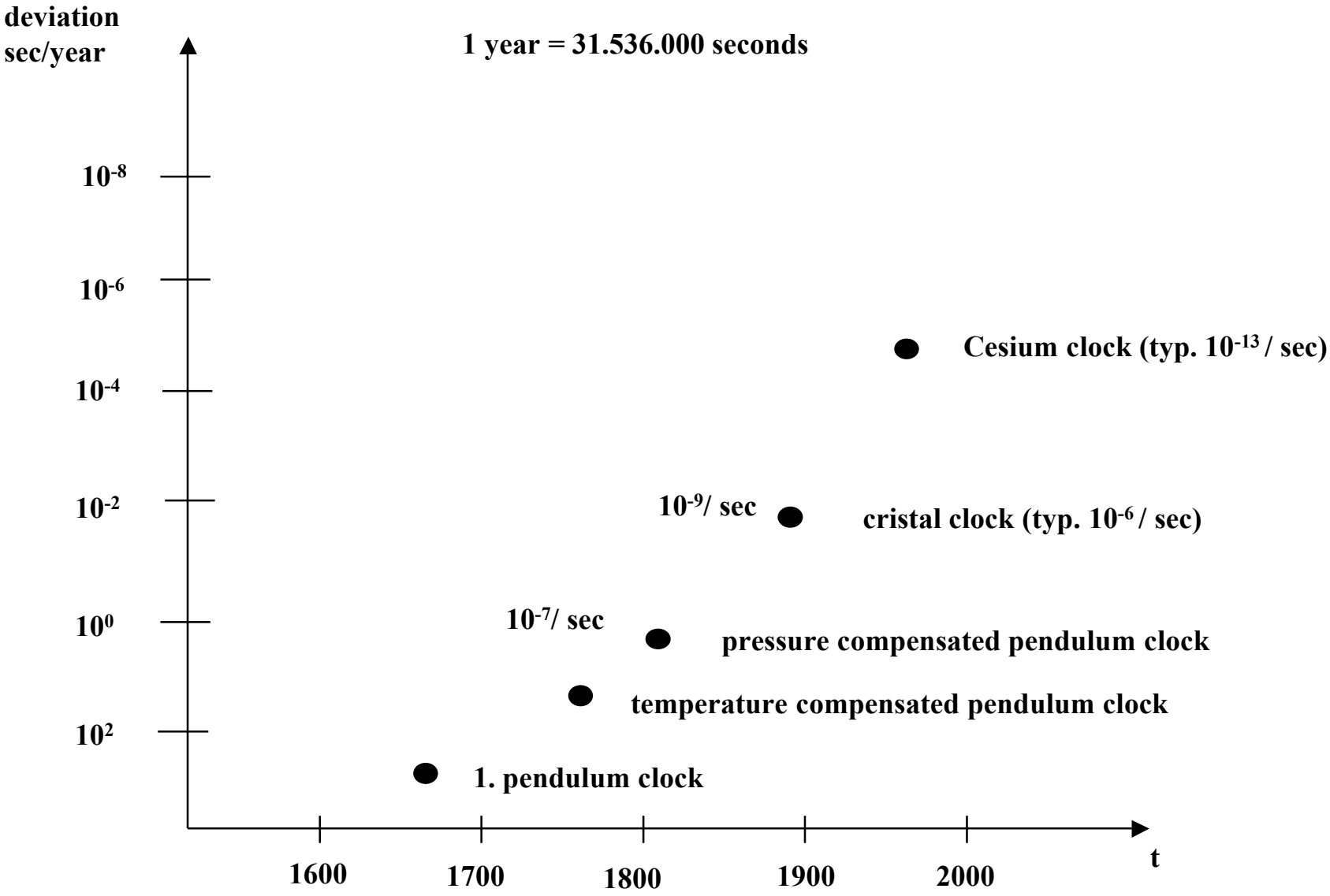
What is the function of time ?

Orientation:
Determine the position of an event in the continuum of time.

Regulation
Enforcing a time regime.

Coordination
Synchronizing cooperative tasks.

Advances in Time Measuremet



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)		**basiert auf der gleichförmigen Bewegung von Himmelskörpern**
Physikalische Zeit (PT)		**basiert auf periodischen physikalischen Prozessen (Schwingungen)**
Sonnendurchlauf:		**Höchster Punkt der Sonne während eines Tages**
Sonnentag:		**Intervall zwischen zwei aufeinanderfolgenden Sonnendurchläufen**
Sonnensekunde:		**Der 1 / 86400 Teil eines Sonnentages**
Mittlere Sonnensekunde:		**Sonnensekunde gemittelt über eine große Anzahl von Sonnentagen Erster Zeitstandard (1820) basiert auf mittlerer Sonnensekunde**
Zeitzonen:		**Gebiete für die dieselbe Zeit festgelegt ist. 1884 wird die Welt in 24 Zeitzonen aufgeteilt. Die Zeitzonen 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 Sonnentags, d.h. auf der Erdrotation**
ET	**(AT, 1955)**	**Ephimeridenzeit (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 aufeinaderfolgenden 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 Zeitzählung, 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 occurence 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

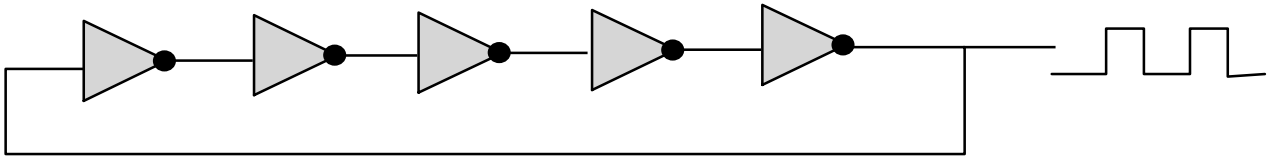
Physical clock: Time reference that is based on periodic physical processes like oszilations of a pendulum, a cristall or an atom to measure the progression of time.

Characterization of physical clocks:

Frequency/Rate:	**f**	**Number of periodic events per time frame. (oszllations, clock pulses)** **Metric: Hz, KHz, MHz, GHz, . . .**
Period	**p=1/f**	**Time interval between two consecutive periodic events.** **Metric: sec, ms, *µ*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 garnularity determines the lower bound of the temporal distance beween two events that are not recognized as simultaneous events.** **Metric: sec, ms, *µ*s, ns . . .**

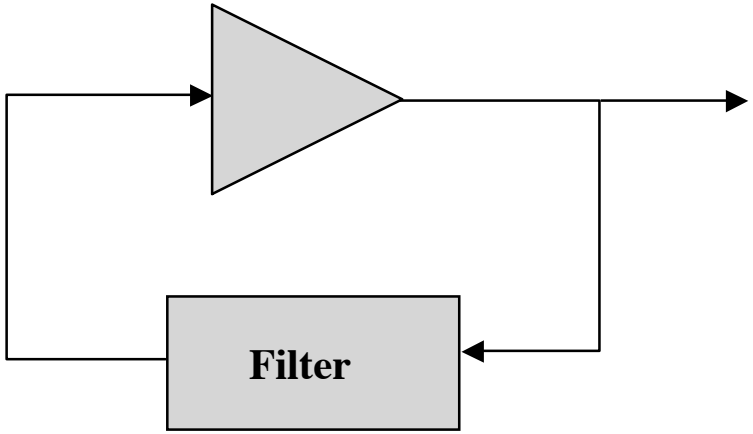
Physical Clocks

Simple oszillator



amplifier

Frequency controlled oszillator



alternatives, e.g.:

Terminology

stability	**:**	**deviation of the specified frequency**
accuracy α	**:**	**deviation wrt. a standard time**
granularity *g* (resolution)	**:**	**smallest measurable time difference**
precision π	**:**	**long term difference between two clocks**
convergence	**:**	**difference of two clocks immediately after a synchronization**
offset δ	**:**	**time difference between two clocks**
skew	**:**	**frequency difference between two (clock) oszillators**
drift ρ	**:**	**slowly increasing difference to a standard time**

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 : ∀ t: C(t) = t

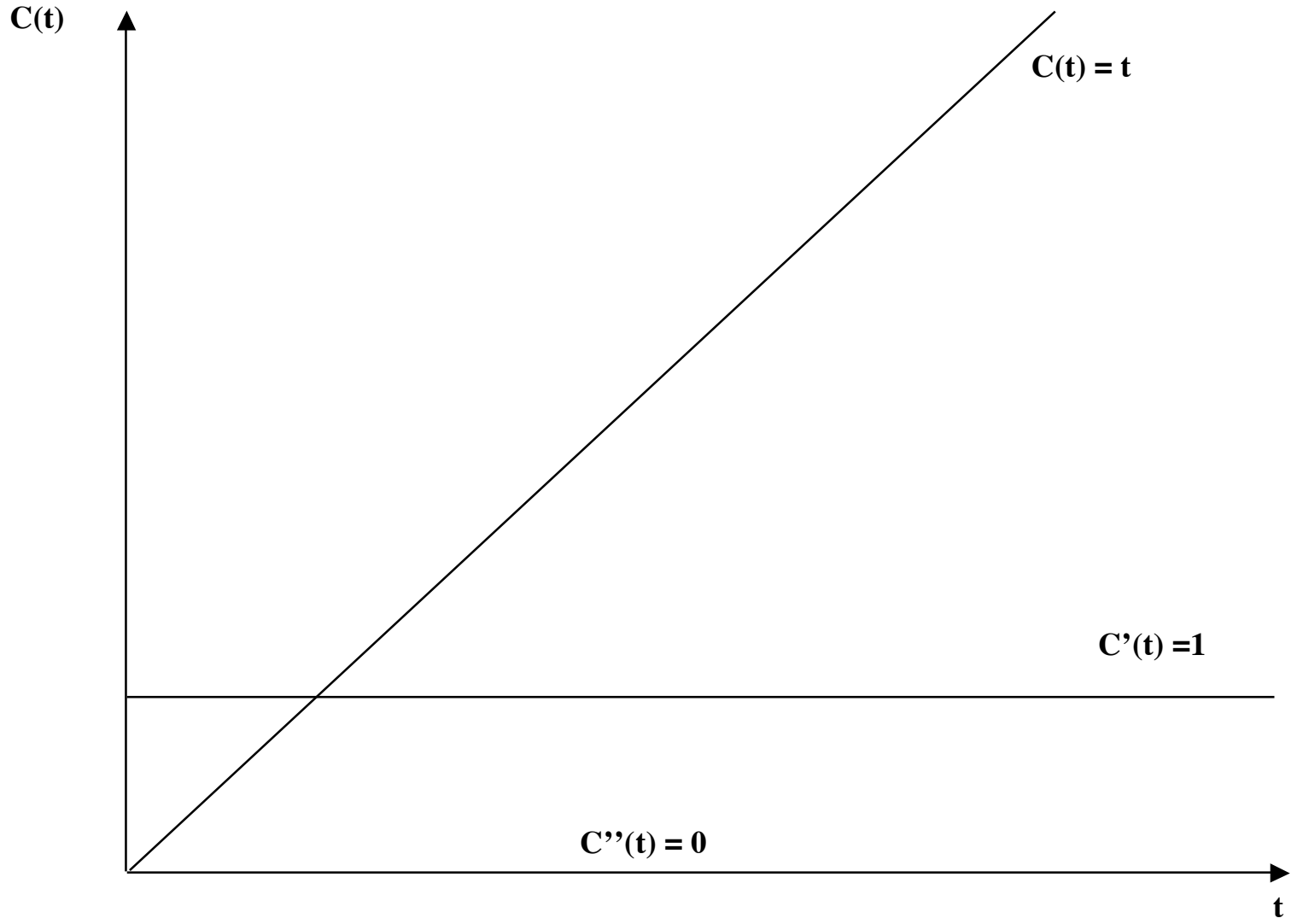
Def. 2:
a clock is correct at time t_0 , if $C(t_0) = t_0$ is satisfied.

Def. 3:
a clock rate is perfect at t_0 , if at t_0 : $C'(t) = 1$ ($dC(t) / dt = 1$)

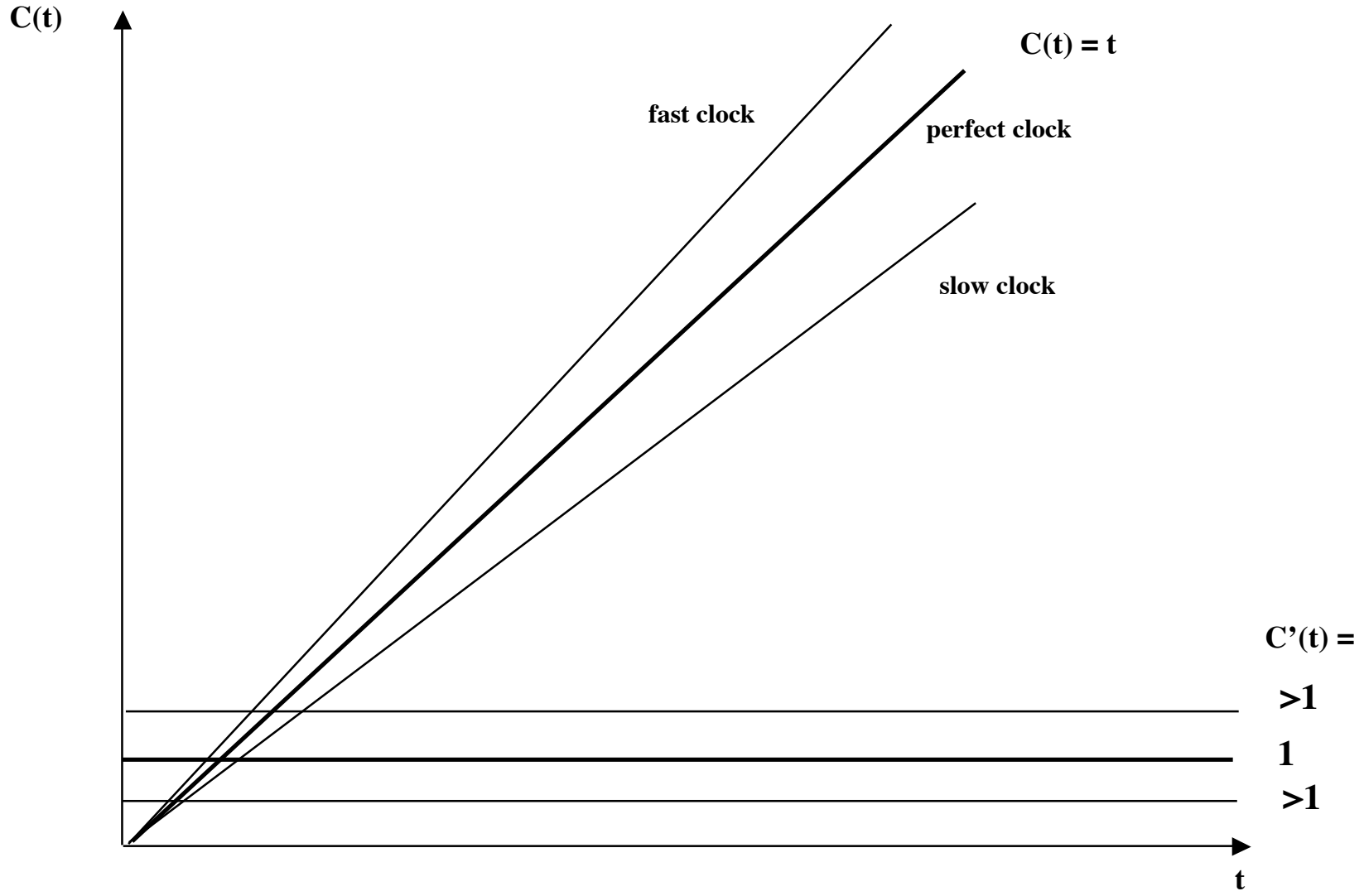
C'(t) denotes the (rate) stability of a clock.
We consider 3 cases:
dC/dt > 1: *fast clock*
dC/dt = 1: *perfect clock*,
dC/dt < 1: *slow clock*

Drift(rate) r : deviation of the clock/time
The drift is assumed to be bounded by:
$1-\rho \leq dC/dt \leq 1+\rho$

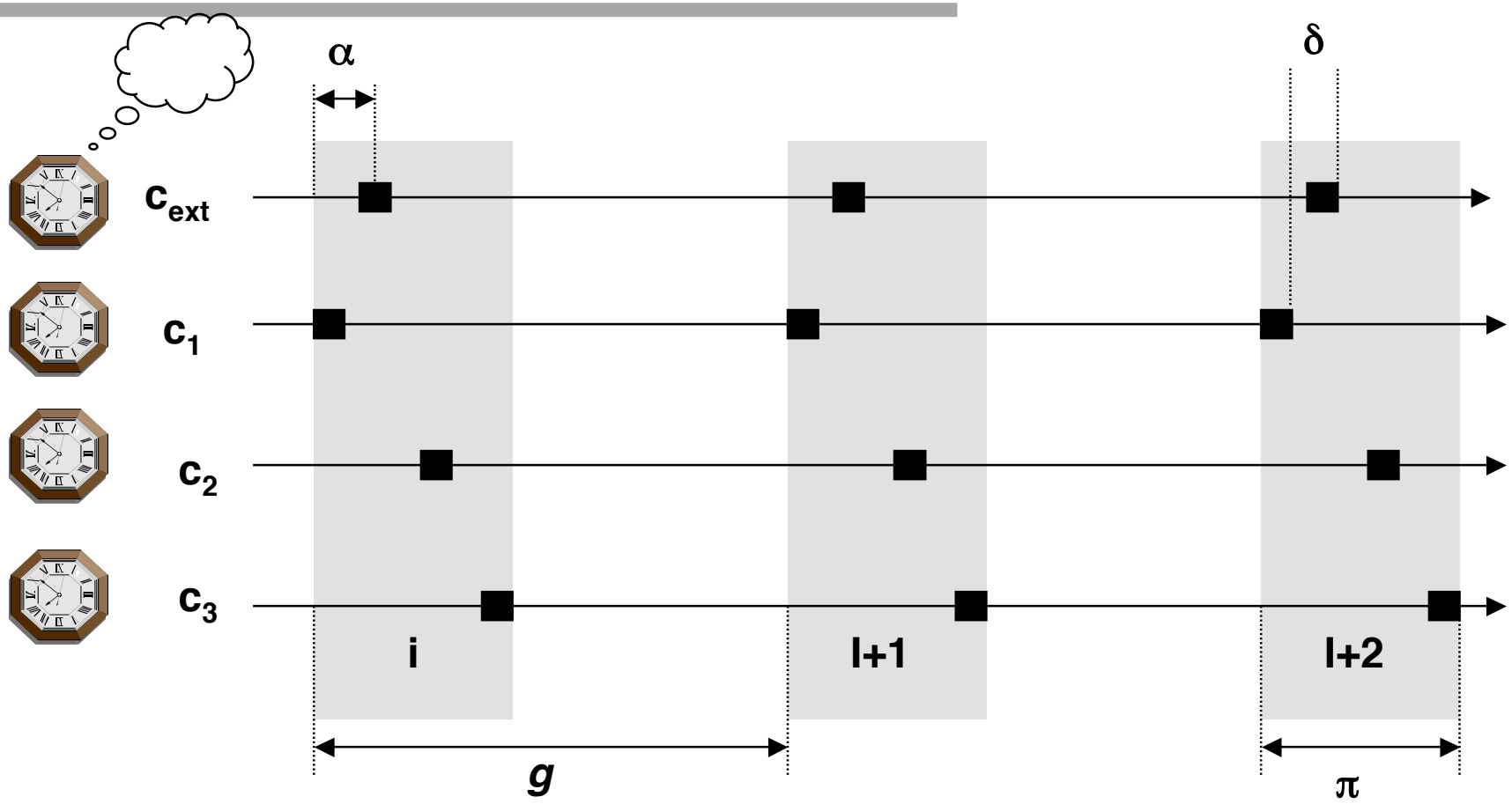
The perfect Clock



Clock with constant Drift Rate

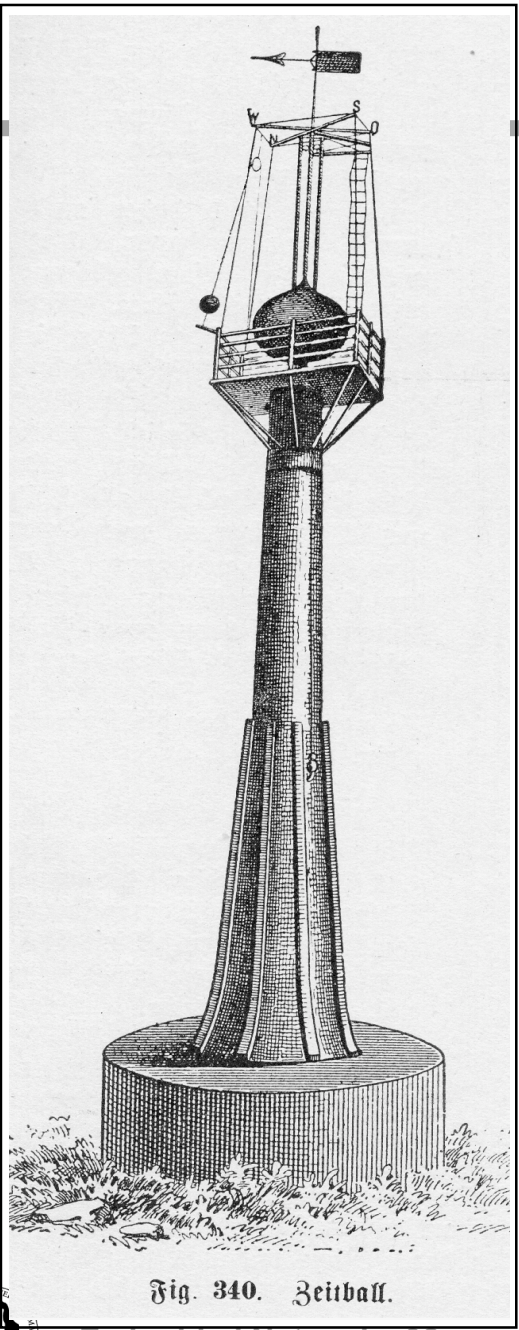


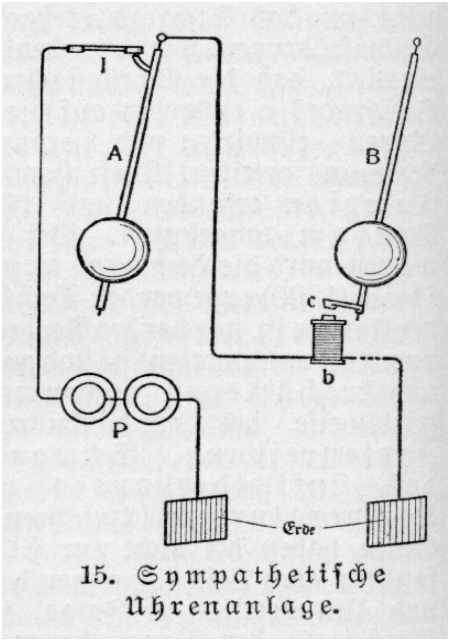
Relation between the main distributed clock parameters

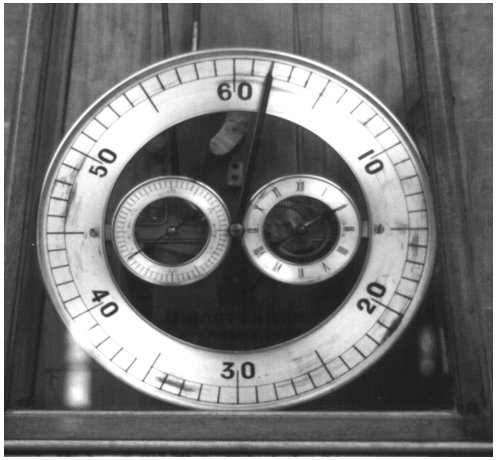


α : accuracy
π : precision
g : granularity
δ: offset

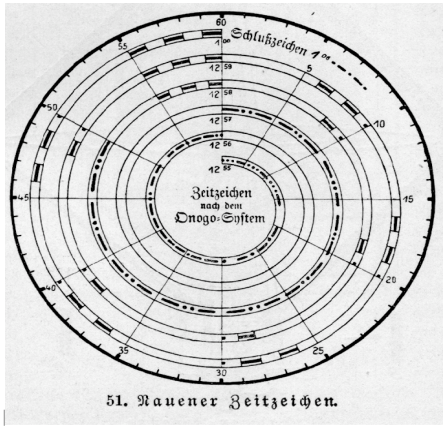
Clock Synchronization







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 ρ:

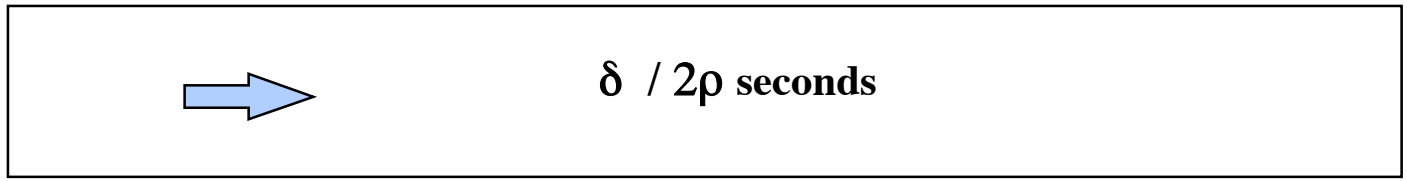
$$1-\rho \leq \frac{dC}{dt} \leq 1+\rho$$

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

difference of two un-synchronized clocks Δt:

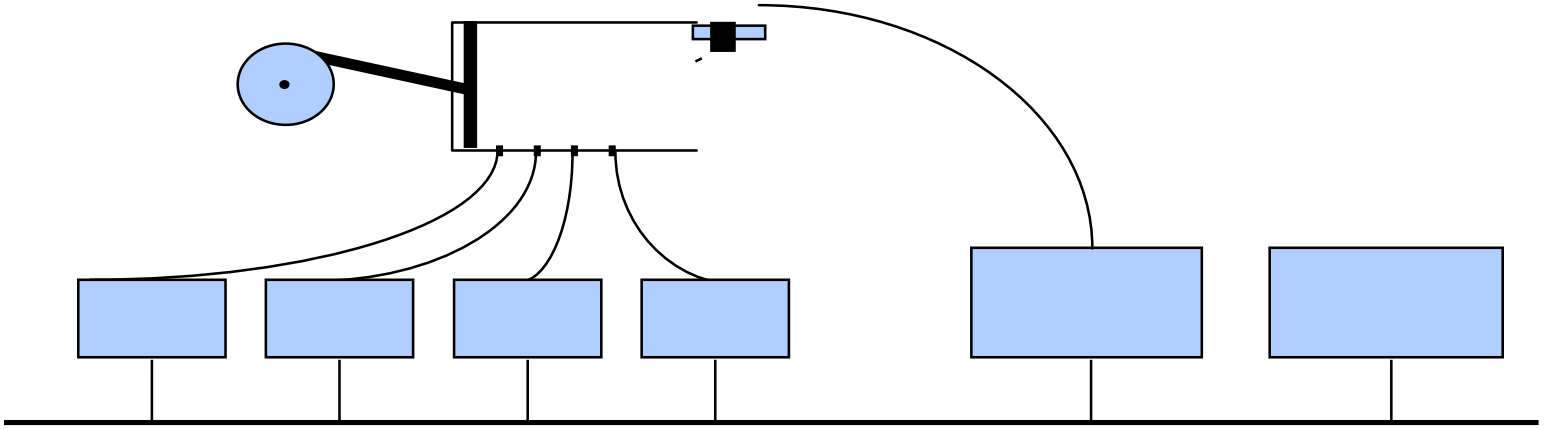
$$\leq 2\rho\Delta t$$

re-synchronization interval if two clocks should not deviate more than δ seconds after



When to re-synchronize ?

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



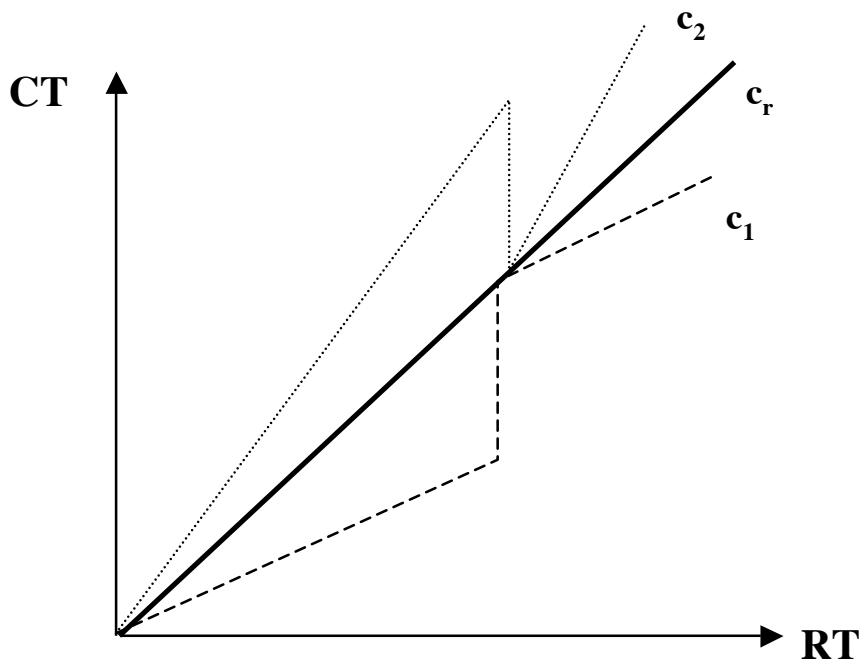
10 Mhz clock; Period = 100ns
after 10 sec., the worst-case difference between clocks can be calculated as:

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

200 times the clock resolution!!

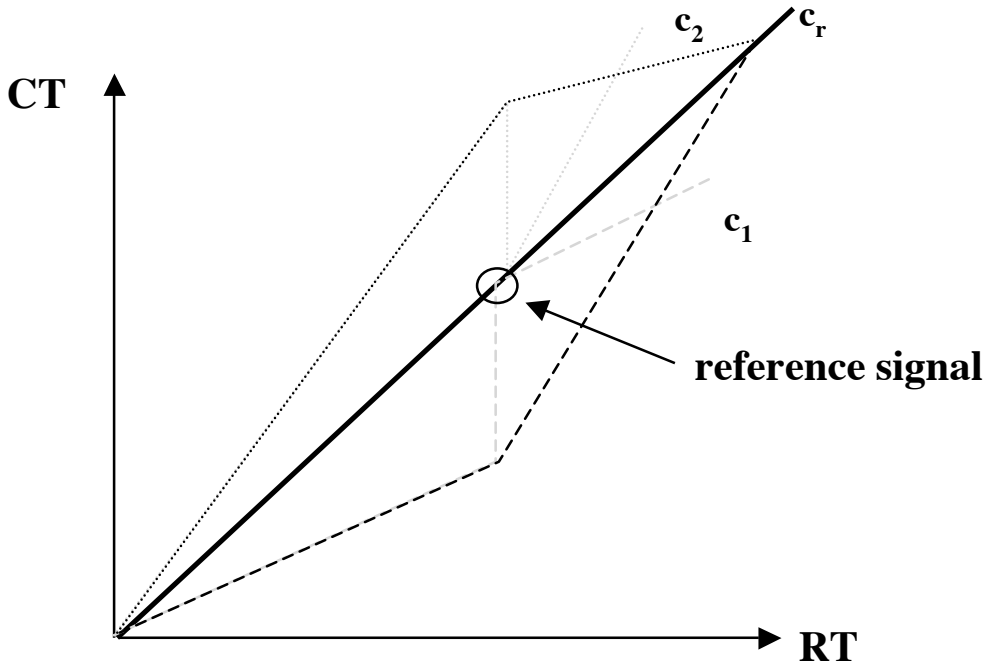
How to synchronize ?

Adapting clock values



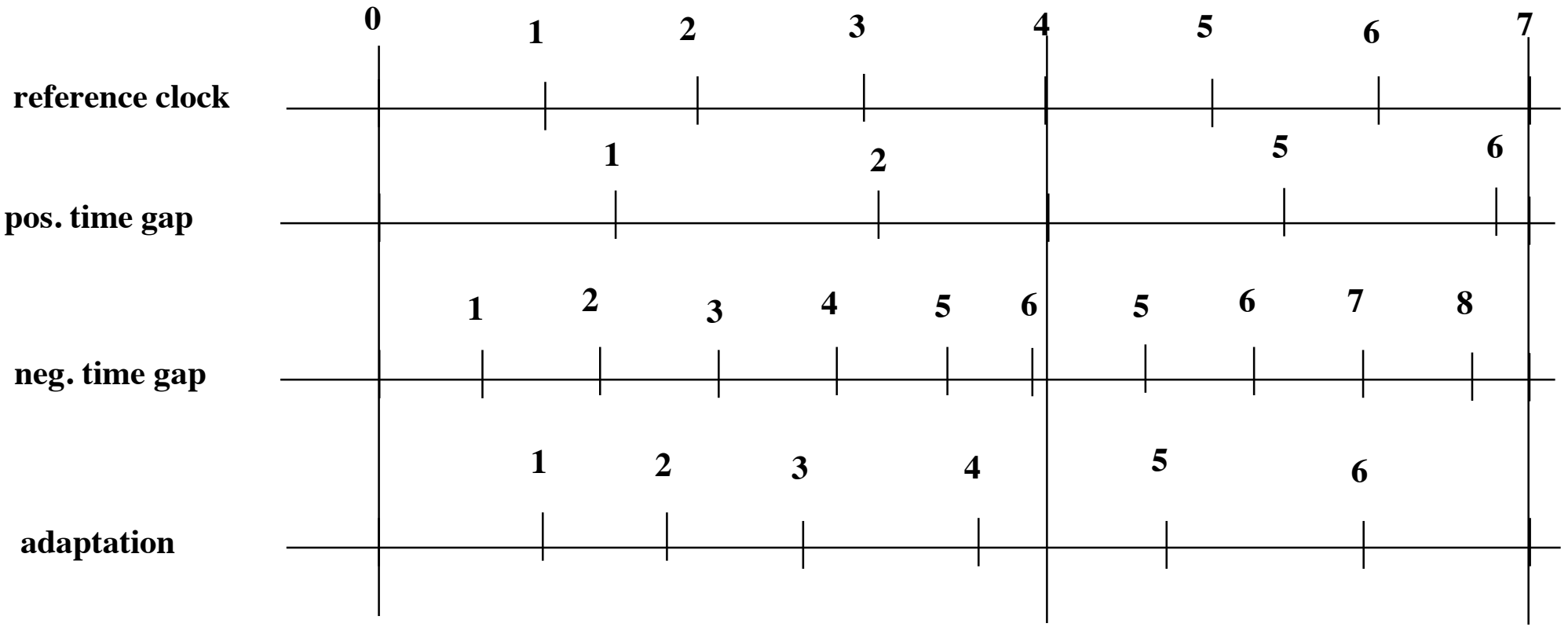
How to synchronize ?

Adapting clock rates

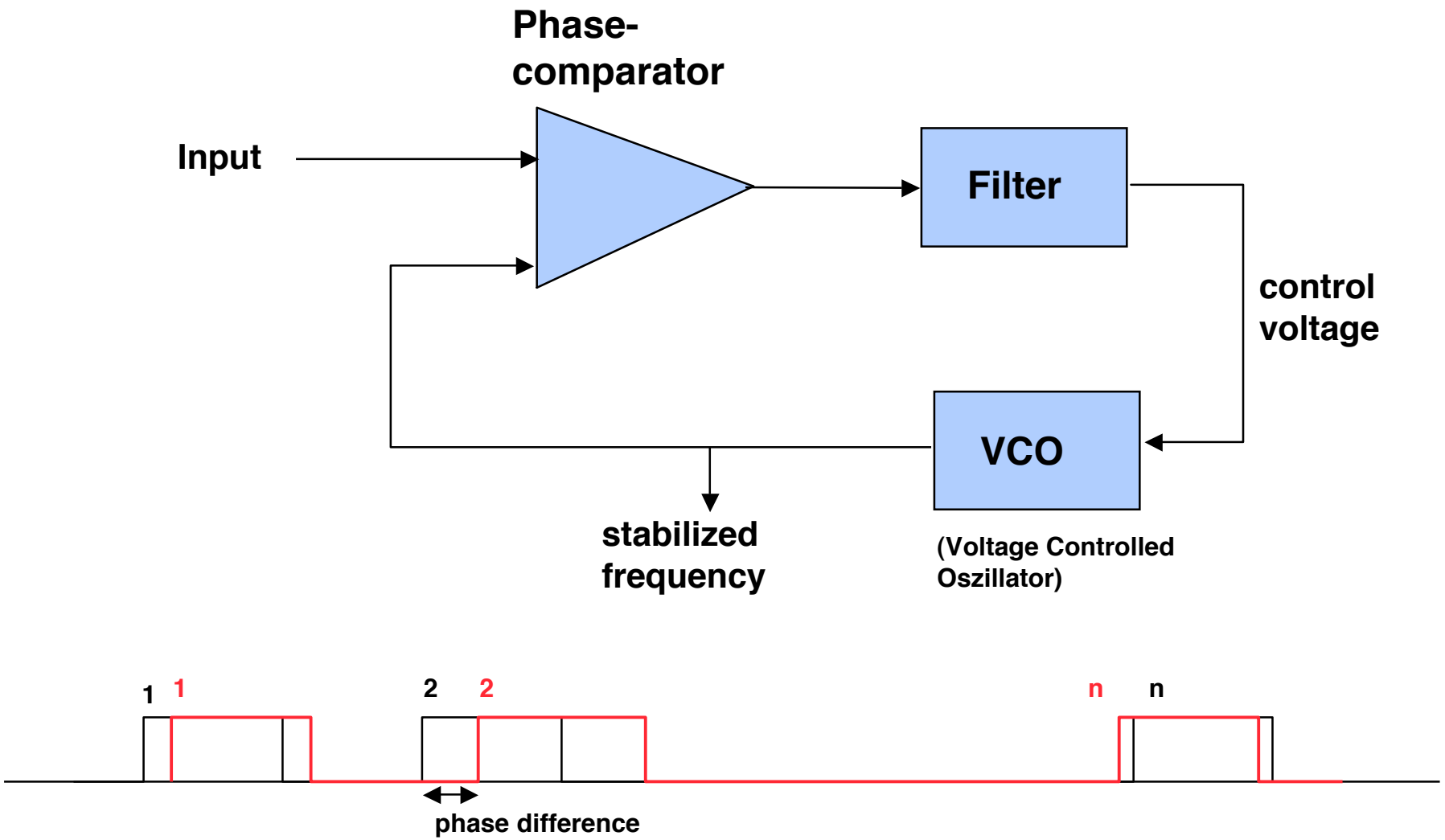


How to synchronize ?

- counter correction
- drift correction



Adapting the clock rate: PLL (Phase Locked Loop)



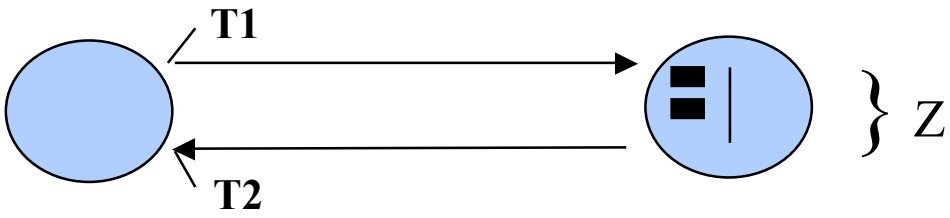
Clock synchronization via messages in communication networks

- 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 δ**

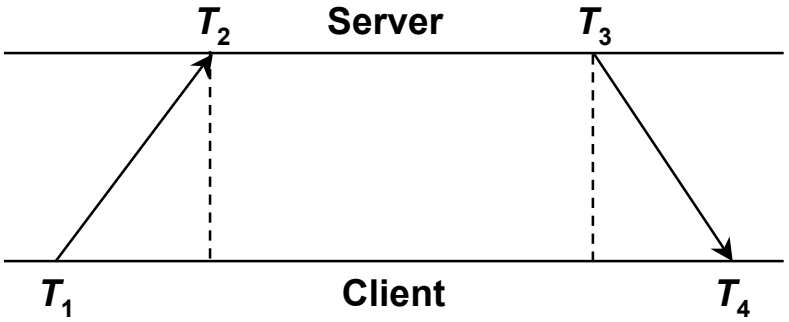


(a) send request for a time stamp ***Z*** to the time server
(b) calculate a correction factor ***K*** to correct the local clock

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

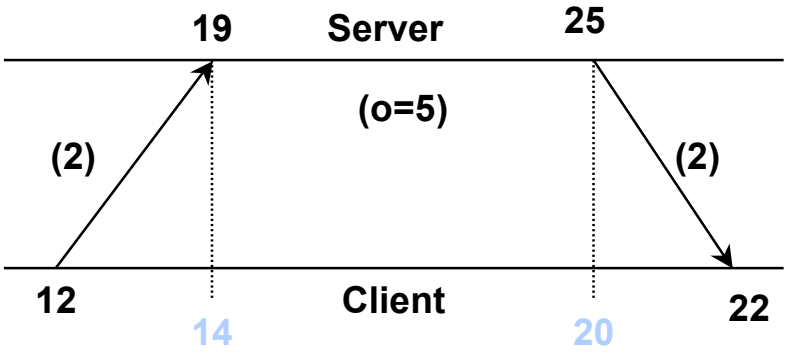


$$o = [(T_2 - T_1) + (T_3 - T_4)] / 2$$

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

Calculating offset and delay

Offsets o
Delay d

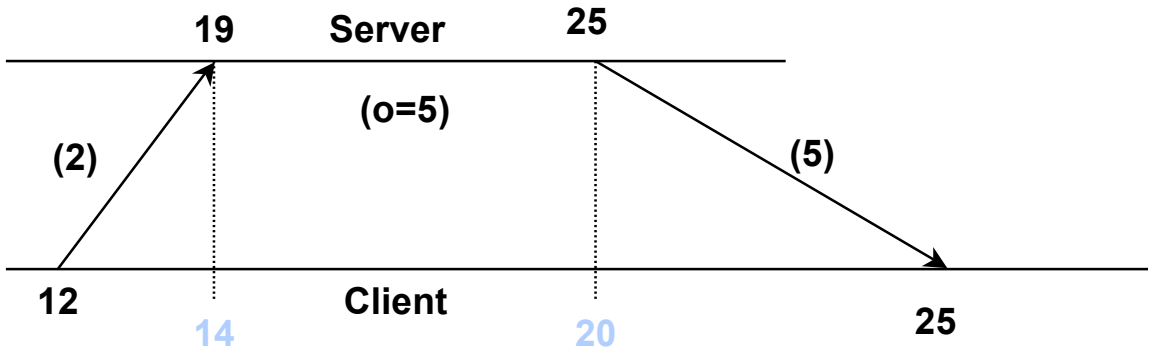


$o = [(19 - 12) + (25 - 22)] / 2$
$o = 5$

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

Calculating offset and delay

Offsets: o
Delay: d



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

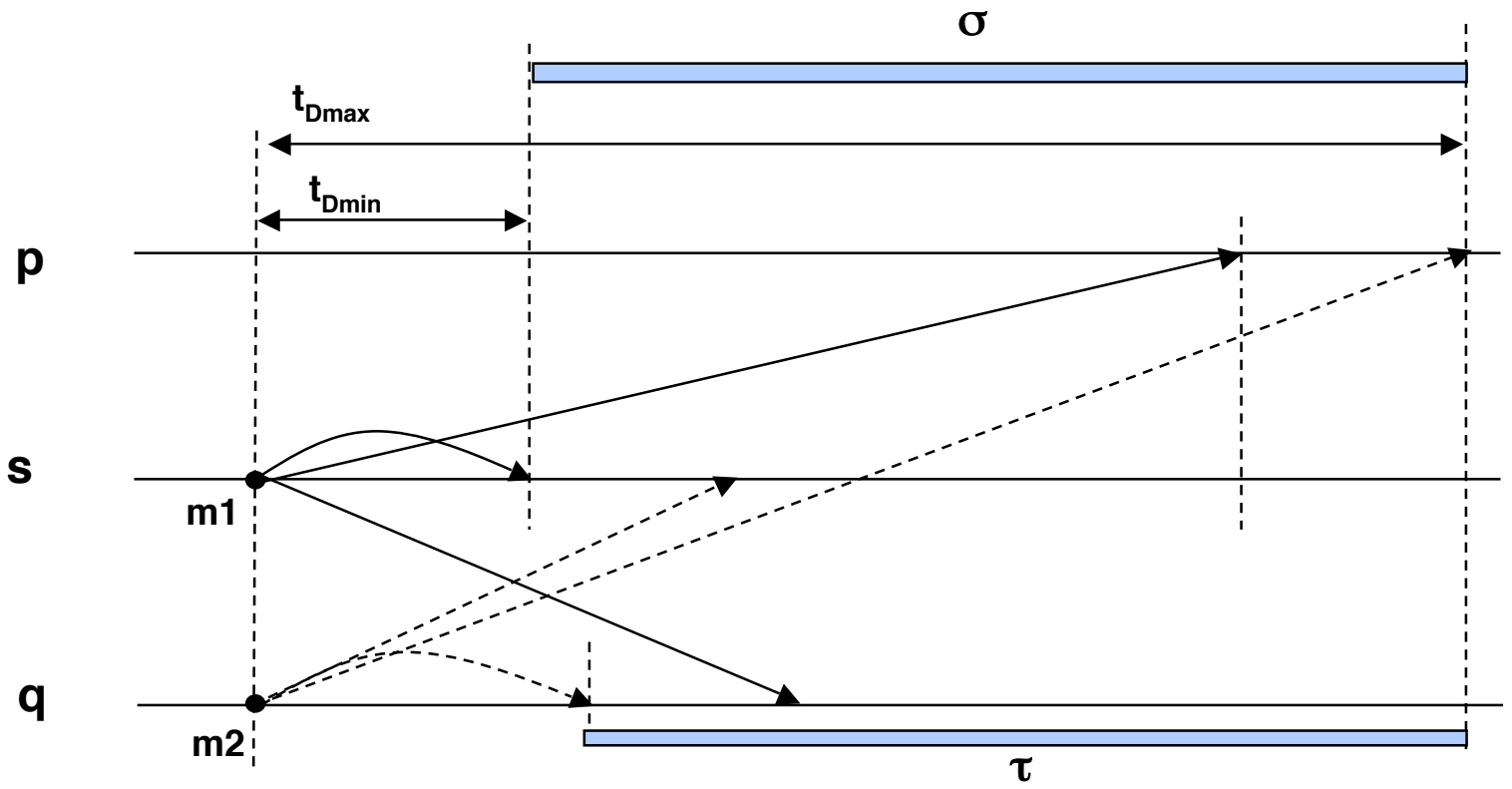
$d = (25 - 12) - (25 - 19)$
$d = 7$

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

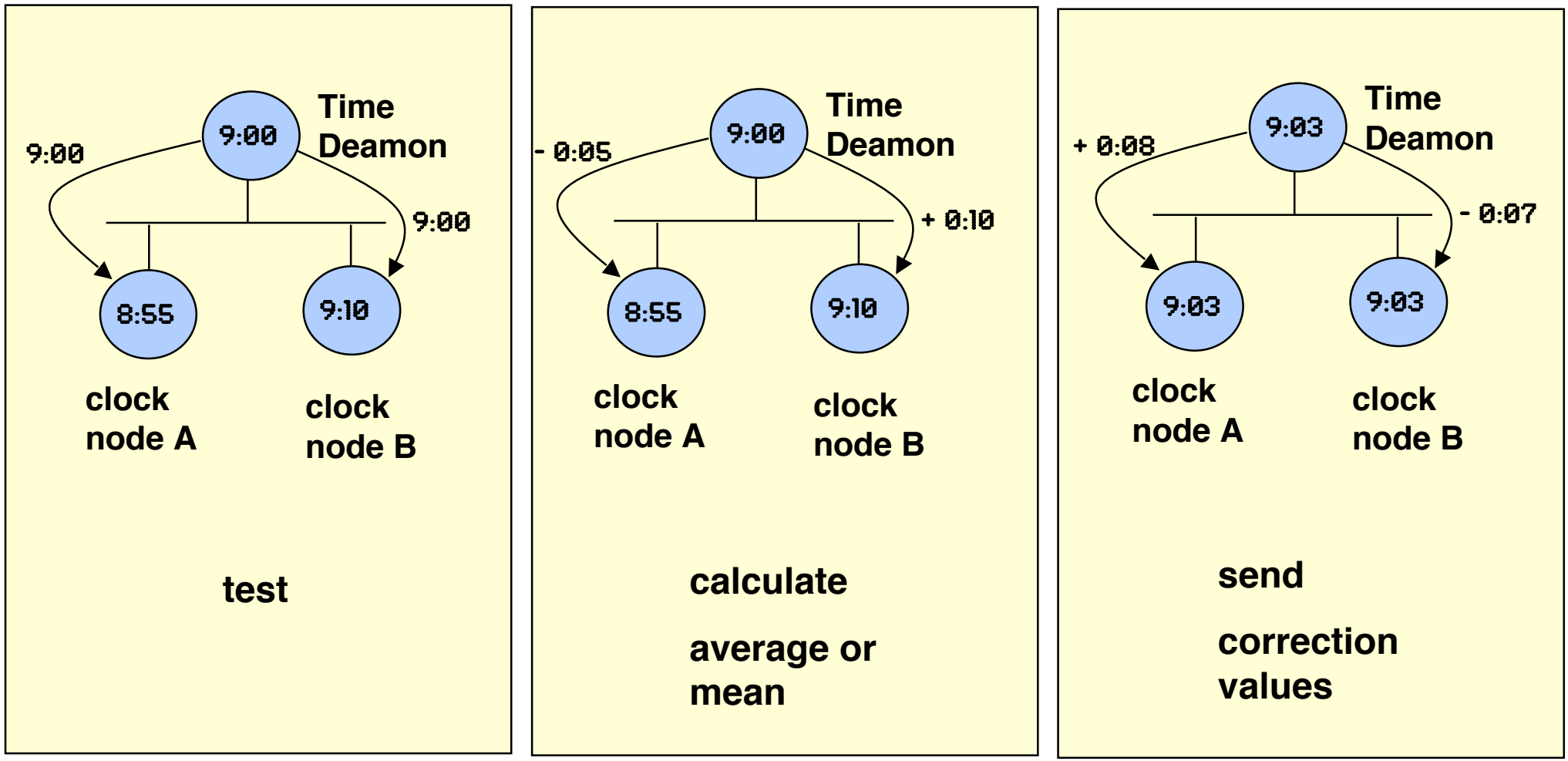
to remember:

Steadyness σ : measures the maximal difference of delivery times of DIFFERENT mesages.
Tightness τ : measures the difference of transmission times of ONE message to DIFFERENT nodes.



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. Meliar-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

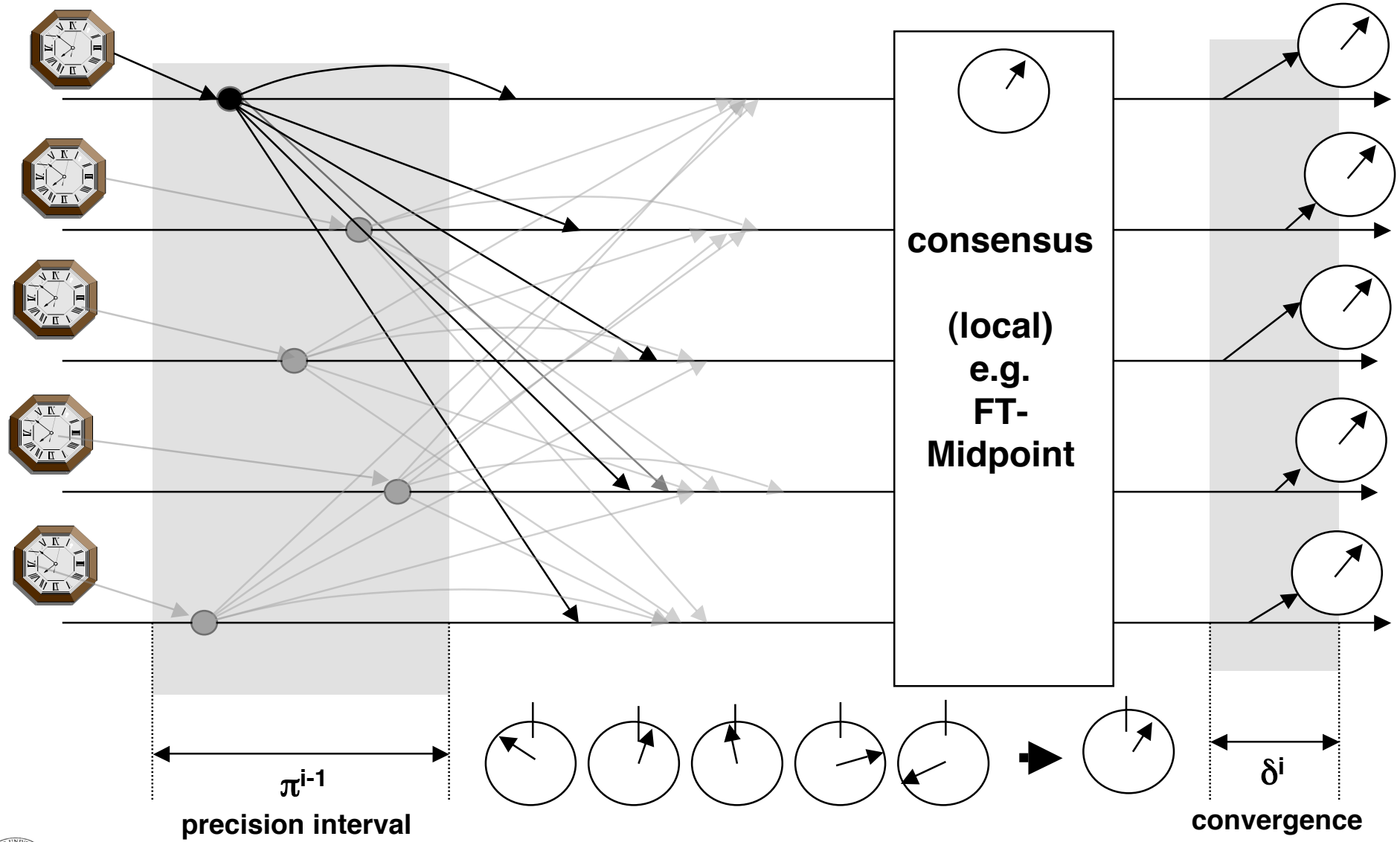
T. Srikanth, S. Toueg:“Optimal Clock Synchronization“, Journal of the ACM,(34)3, 627-645, 1987

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:

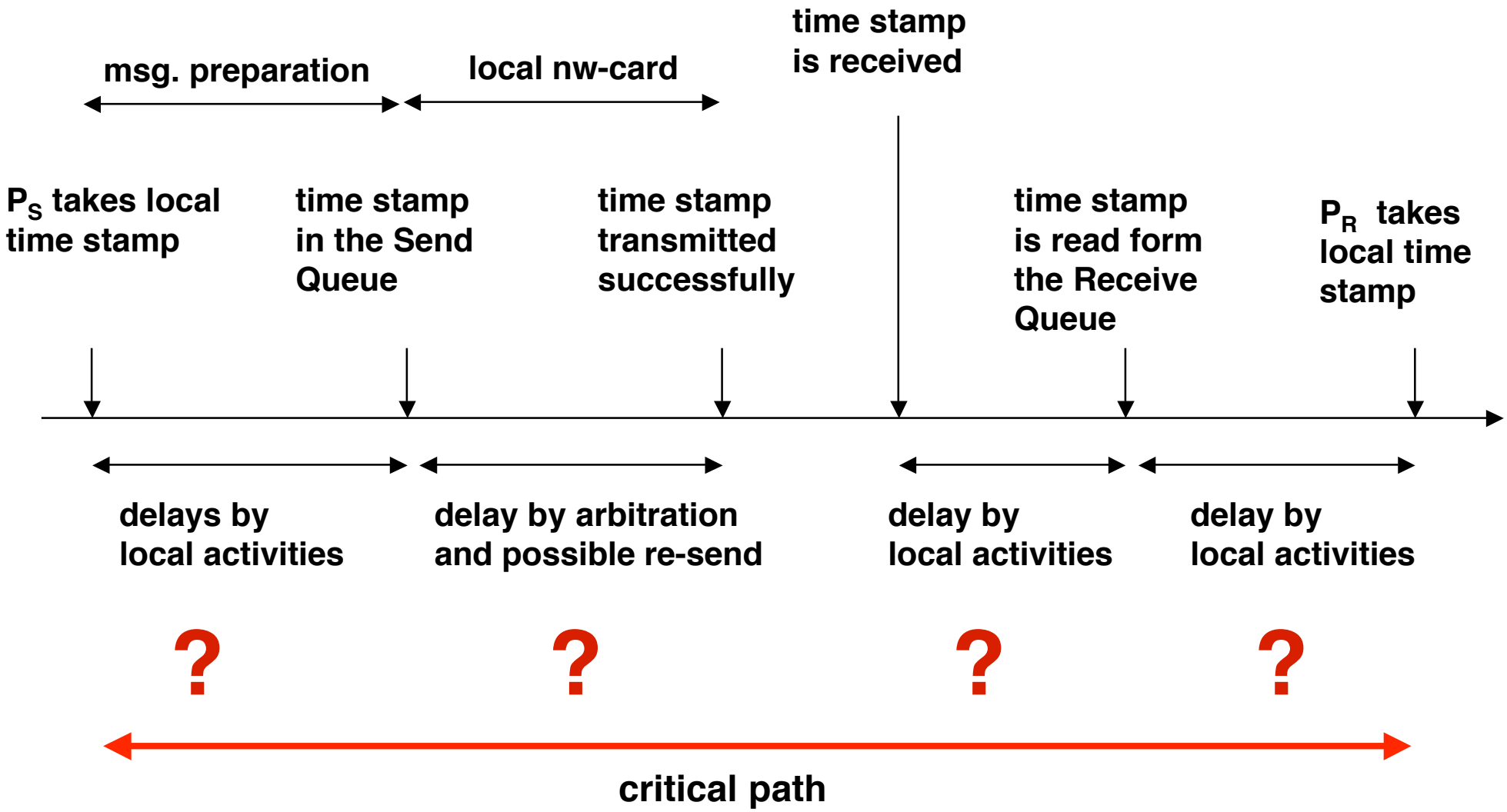
The *k* largest and the *k* smallest values will not be considered.
The average is computed from the values k+1 und n-k:

- Midpoint: (max_value + min_value) / 2
- this is NOT the MEAN value of the sorted list of values!!!!

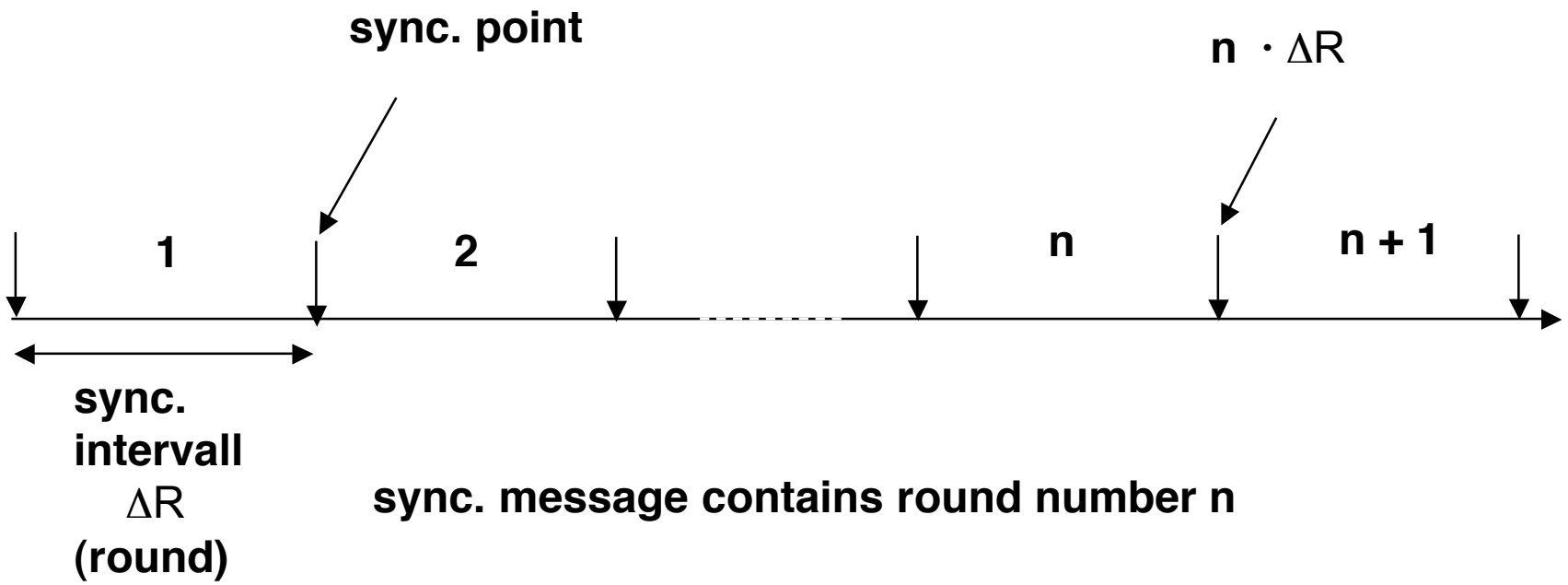
– Fault-tolerant Average:

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

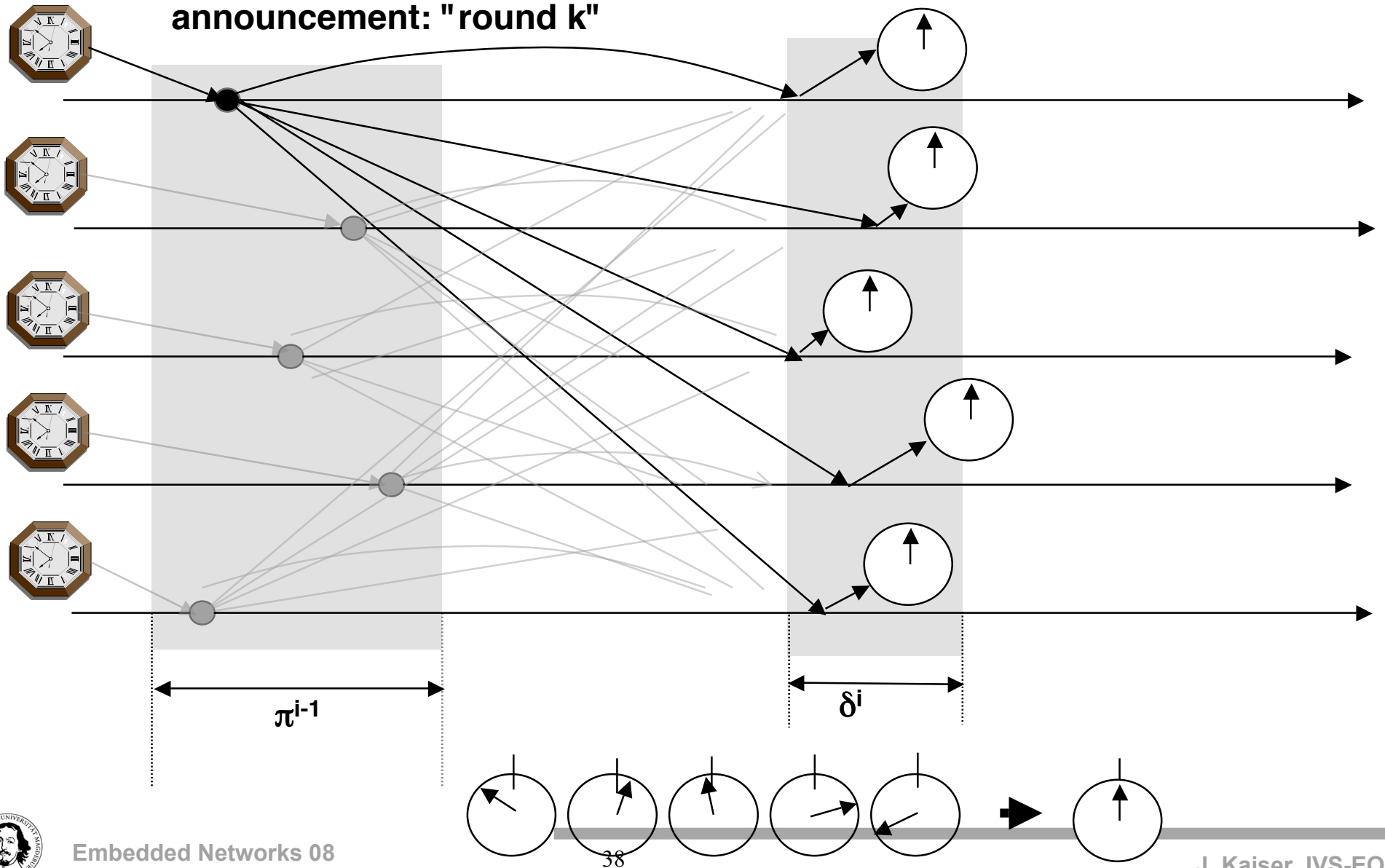
Practical Considerations:



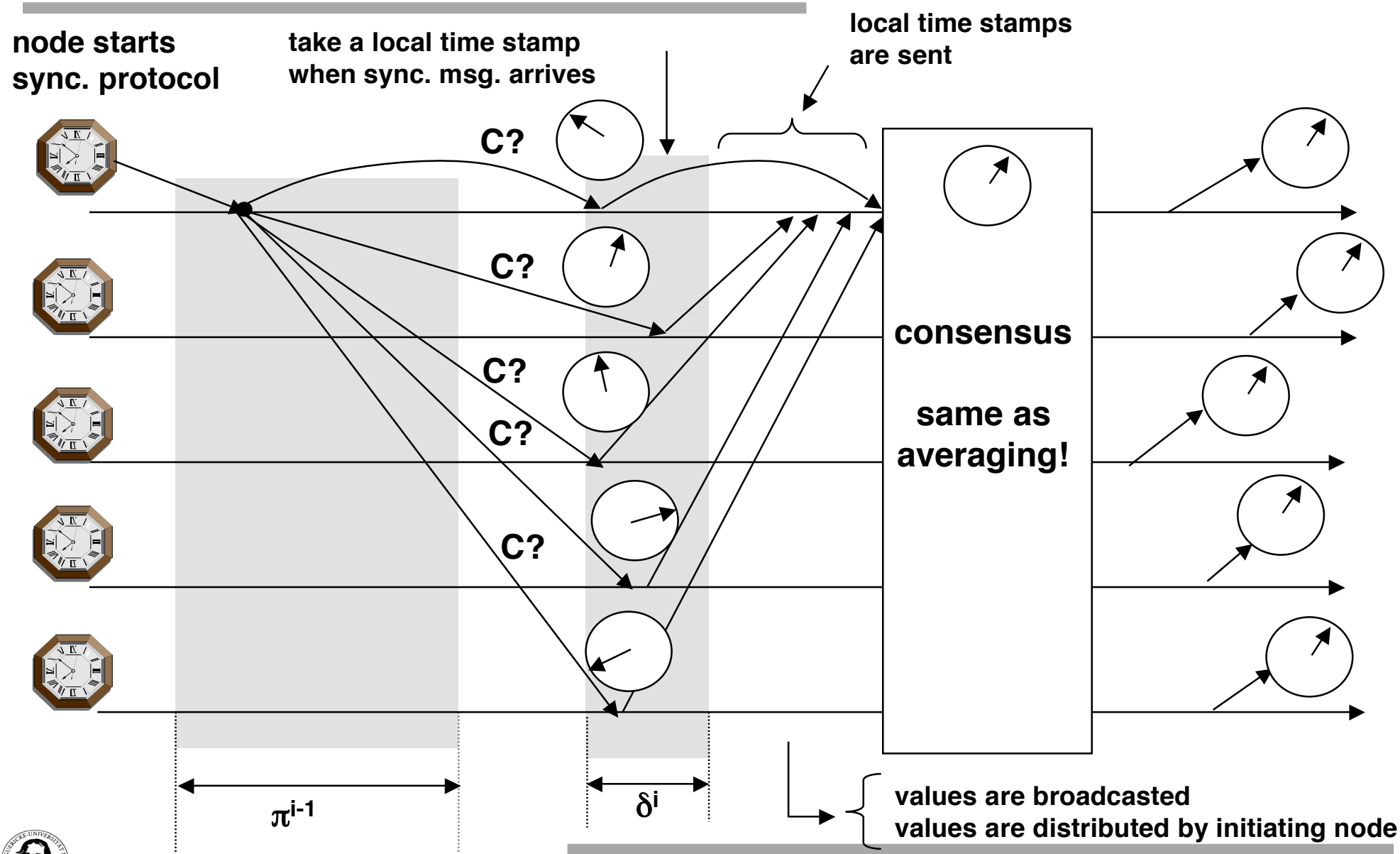
Non-averaging synchronization



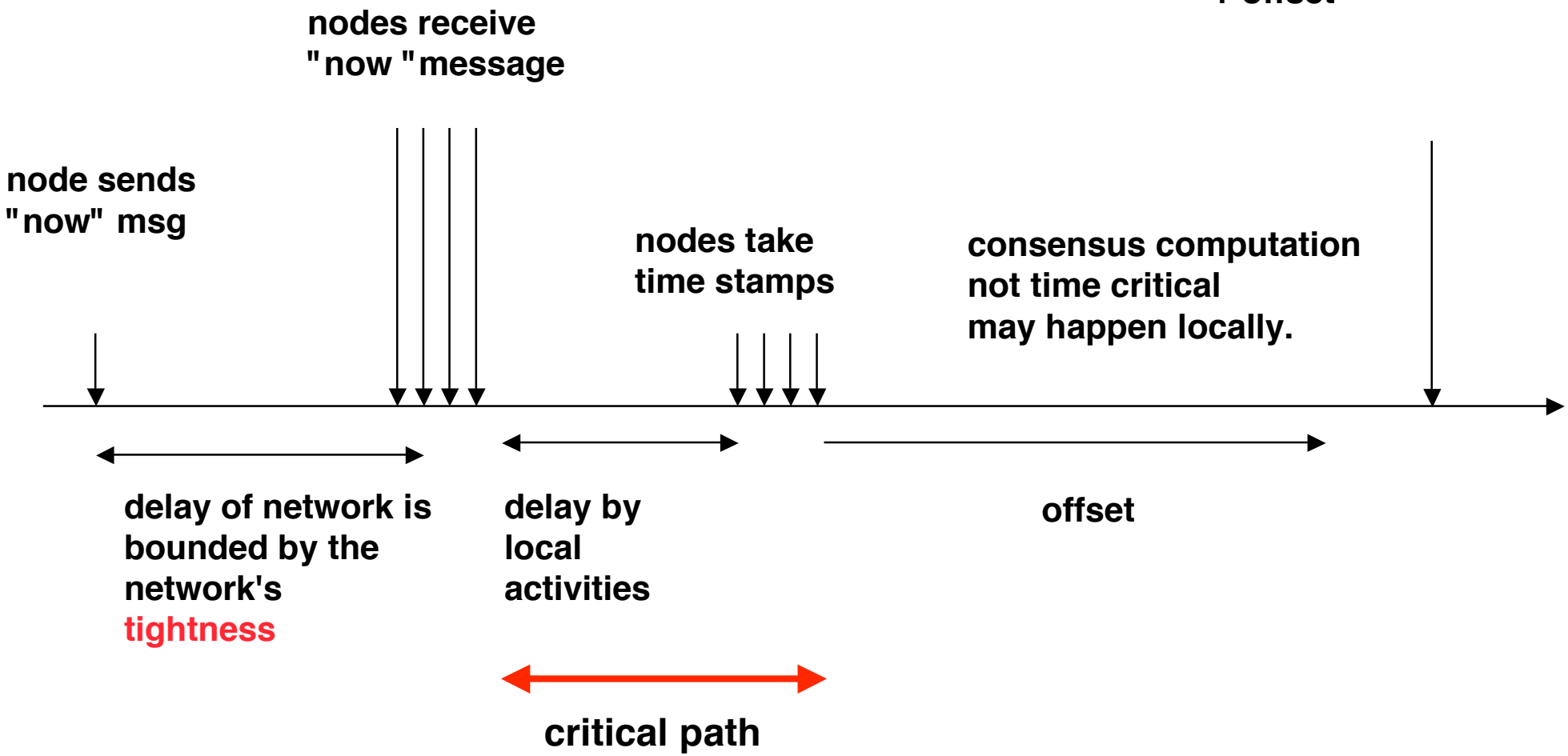
non avaraging synchronization



Hybride syncronization



nodes set new time:
t= corrected time stamp
+ offset



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:

$$C_j = \begin{Bmatrix} t_{1,1} & \cdots & t_{1,j} & \cdots \\ \cdots & t_{2,2} & t_{2,j} & \cdots \\ \cdots & \cdots & t_{3,j} & \cdots \\ \cdots & \cdots & t_{j,j} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & t_{n,j} & \cdots \; t_{n,n} \end{Bmatrix}$$

*

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:

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

$O_{i,j}$ (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

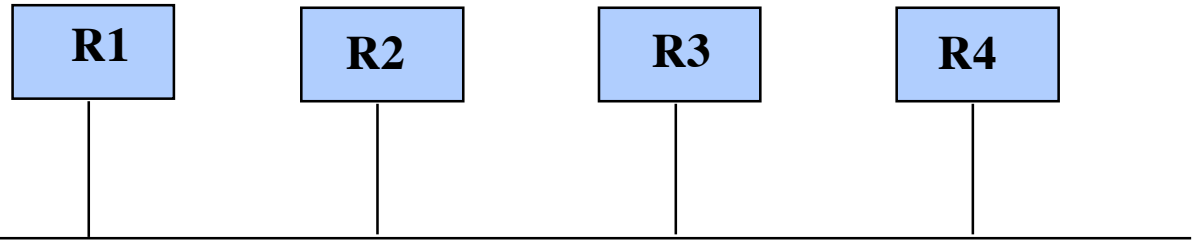
$$K^t = \begin{pmatrix} 0 & k_{1,2} & \cdots & k_{1,j} & \cdots & k_{1,n} \\ k_{2,1} & 0 & \cdots & k_{2,j} & \cdots & k_{2,n} \\ \vdots & \vdots & & k_{3,j} & & \vdots \\ \vdots & \vdots & 0 & & & \\ \vdots & \vdots & \cdots & 0 & \cdots & \vdots \\ \vdots & \vdots & & & & \vdots \\ & & & & 0 & \\ k_{n,1} & k_{n,2} & \cdots & k_{n,j} & \cdots & 0 \end{pmatrix}$$

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

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

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	.	.	.
69	60	.	.
95	.	87	.
20	.	.	16

j=1

32	37	.	.
.	60	.	.
.	94	87	.
.	19	.	16

j=2

32	.	38	.
.	60	67	.
.	.	87	.
.	.	21	16

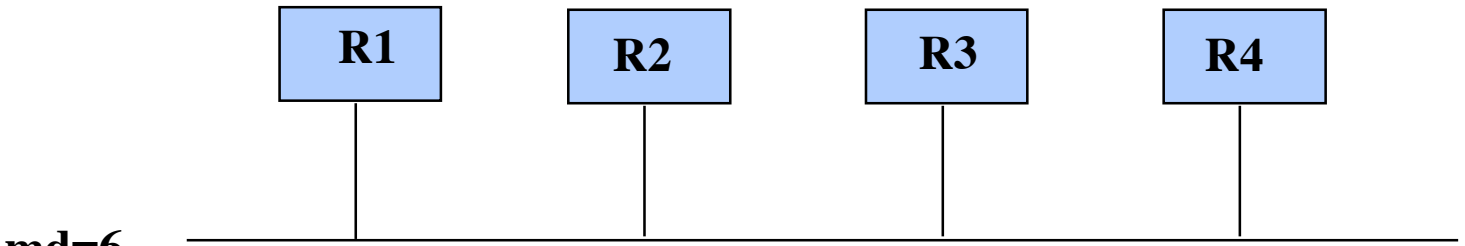
j=3

32	.	.	41
.	60	.	66
.	.	87	98
.	.	.	16

j=4

Co-operating clock synchronization

Example:



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

time stamp contains local receive time

$C_j^r =$

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

j=1 j=2 j=3 j=4

Co-operating clock synchronization

$C_j^r =$

32	.	.	.
69	60	.	.
95	.	87	.
20	.	.	16

32	38	.	.
.	60	.	.
.	94	87	.
.	19	.	16

32	.	38	.
.	60	67	.
.	.	87	.
.	.	21	16

32	.	.	41
.	60	.	71
.	.	87	98
.	.	.	16

j = 1

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

j = 2

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

j = 3

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

j = 4

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

$K^t =$

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

Co-operating clock synchronization

Example:

$C_j =$

32	.	.	.
69	60	.	.
95	.	87	.
20	.	.	16

32	37	.	.
.	60	.	.
.	94	87	.
.	19	.	16

32	.	38	.
.	60	67	.
.	.	87	.
.	.	21	16

32	.	.	41
.	60	.	71
.	.	87	98
.	.	.	16

$K^t =$

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

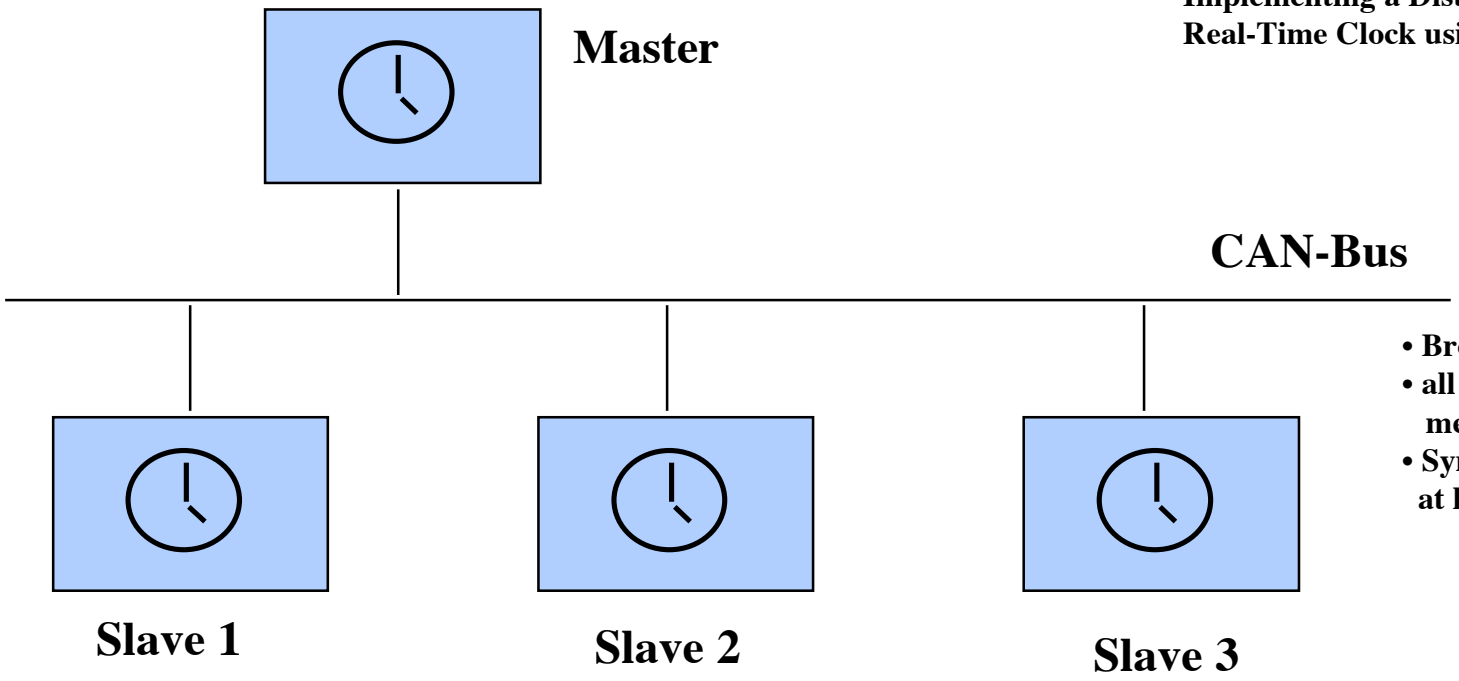
$j = 1$	$j = 2$	$j = 3$	$j = 4$	$\frac{\sum_{i=1}^{n} k_{i,j}}{n}$
0	-1	0	3	
3	0	1	5	
2	1	0	5	
-2	-3	-1	0	

= 3/3 = 1	= -3/3 = -1	= 0/3 = 0	= 13/3 = 4	correction values

clock will slow down	clock will speed up	clock runs correct	clock will slow down

Clock synchronization on the CAN-Bus*

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



- Broadcast
- all participants receive the message at the same time
- Synch.- messages can be sent at high priority

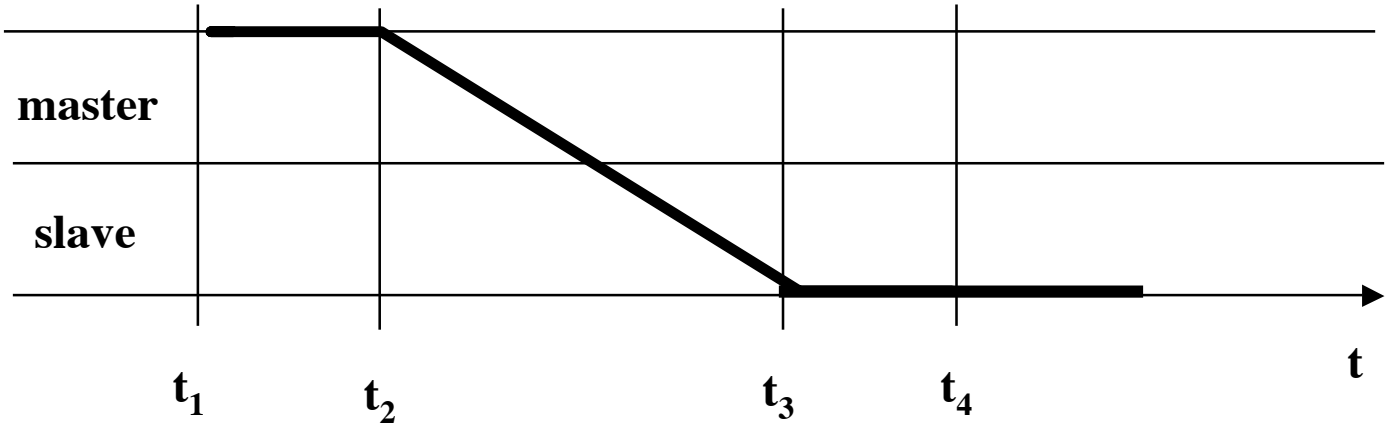
System:

- 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

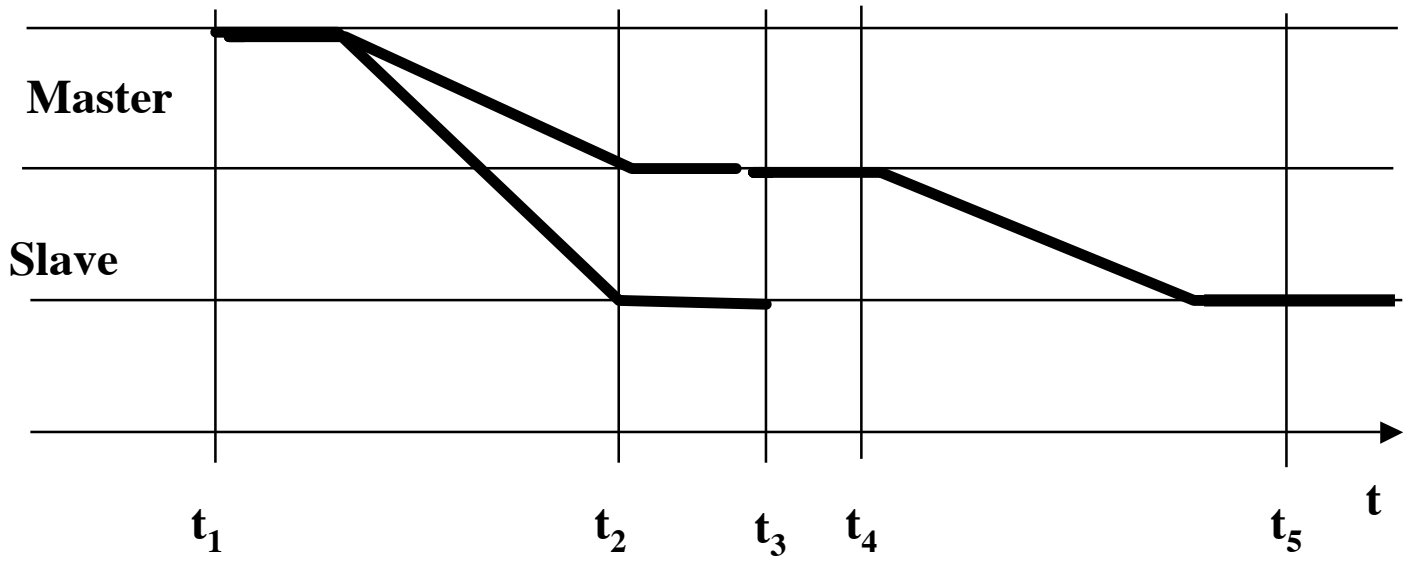
- **master sends a time stamp**
- **slaves receive the message**
- **slaves adjust local clocks**



Clock synchronization on the CAN-Bus*

improved protocol

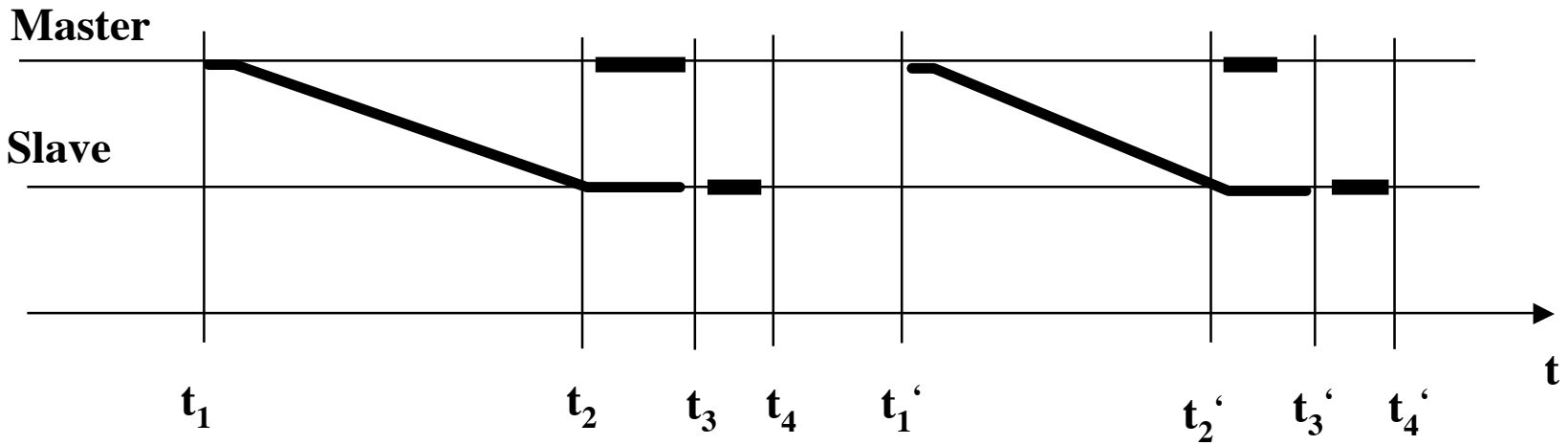
- some node i sends the message "now" (t_1)
- master and slaves read their local clocks when receiving the message at (t_2)
- master sends its time stamp (t_3, t_4)
- slaves adjust their clocks according to the master time stamp (t_5)



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 μsec
- synchronization with external time
  - e.g. via GPS receiver
- no additional cost
  - impelmented on embedded controller

- Application areas:
  - co-operative data acquisition
  - co-operative control