Time and Order
(in Real-Time Systems)

Overview

Events, computer generated and environmental

(Real) Time

The order of events, temporal and causal

Logical Clocks, 2 versions

Physical Clocks and their properties

Global (real) time in distributed systems

Topics

Can clocks (logical or physical) be used

- to derive the order of events
- to identify events
- to generate events at certain points in time ?

Which precision can be achieved

- to measure time ?
- to measure durations ?

How and how often have clocks to be synchronized?

Time in Distributed (Real-Time) Systems

Actions/events/... in distributed real-time systems

- concurrent
- on different nodes
- must have a consistent behaviour / order.

Consistent order

- temporal order
- causal order

Global Time Base

Events 1

Computer Generated Events:

- execution of statement
- sending/receiving a message
- start and end of a compilation
- creation/modification of a file
- ...

Sequence of states is determined by

- instructions, disk accesses, ...
- discrete steps

Events 2

Environmental Events:

- newton mechanics
- pipe rupture
- human interaction
- ...

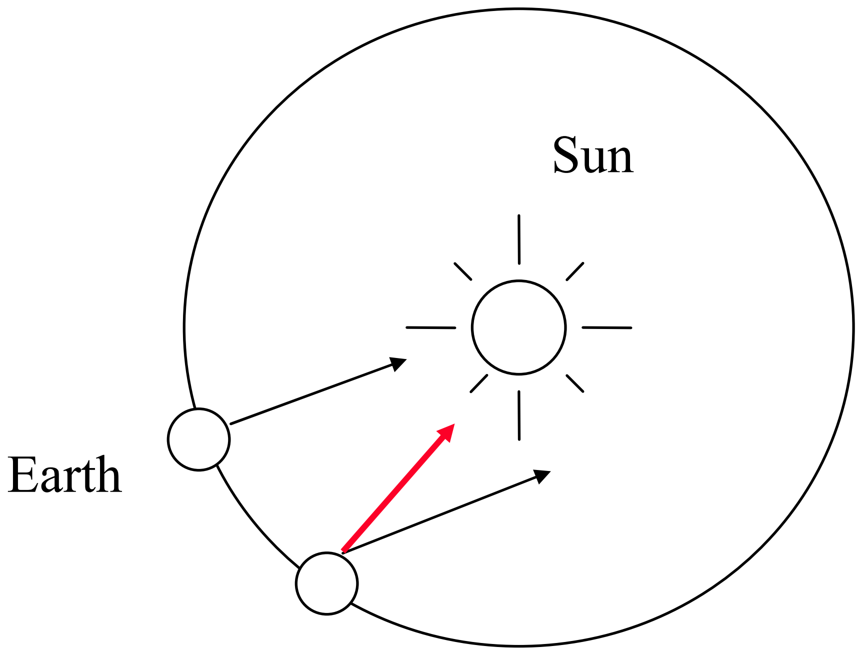
Sequence of states is determined by

- laws of physics
- physical (or real) time: „second"
- continuous

Astronomical Time

Solar Day: from noon to noon

Solar Second: Solar Day / (24 * 60 * 60)



Atomic Time

TAI ... International Atomic Time:

1 second =

- “duration of 9192631770 (9 Gigahertz)
  periods of of the radiation of a specified transition of the caesium atom 133“ (Kopetz)

Time Standard(s)

UTC .... Coordinated Universal Time (UTC)
TAI adjusted to Astronomical time

Sources:

- earth-bound radio
- Geos satellites
- GPS

Temporal vs. Causal Order of Events

Temporal Order:

- induced by (perfect) timestamp

Causal Order:

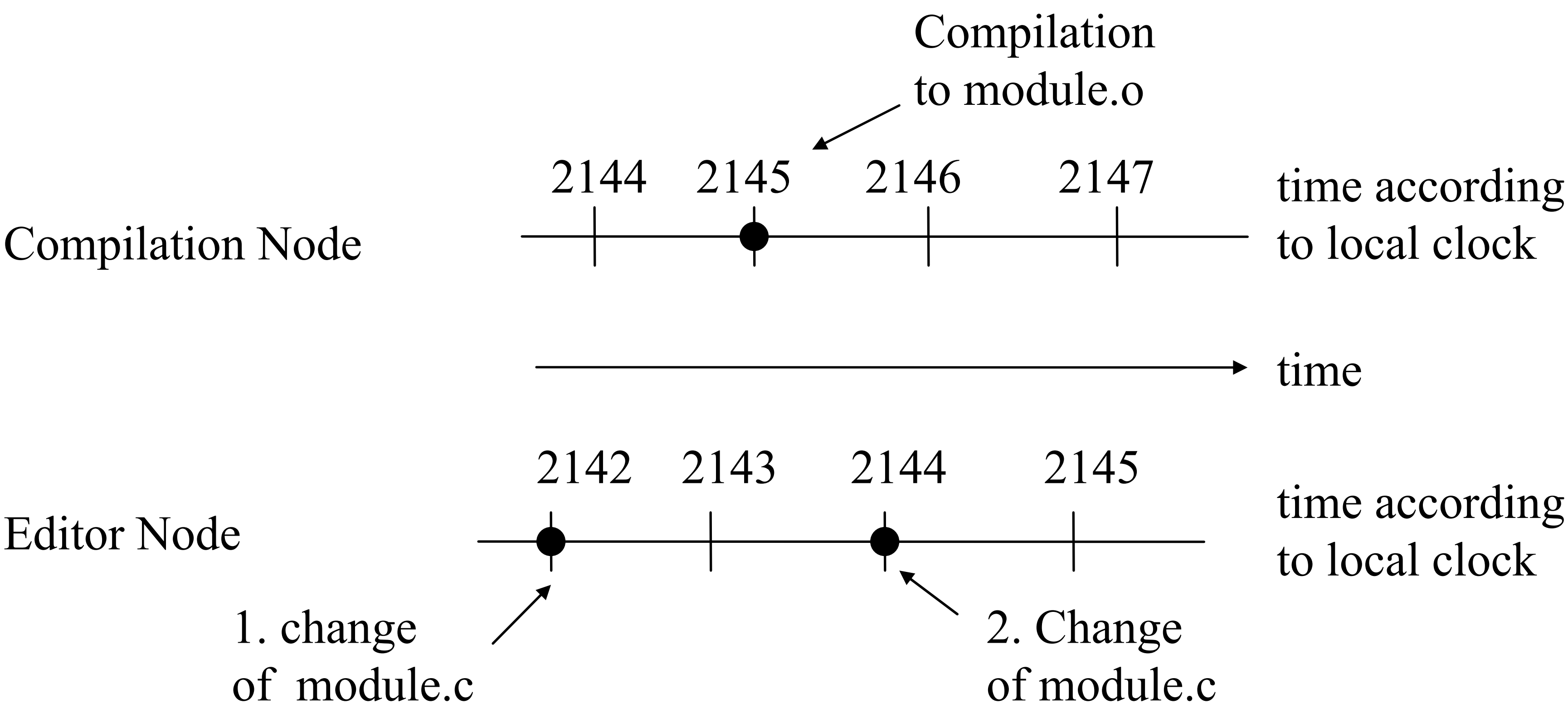
- induced by some causal dependency between events

Example

- e1: somebody enters a room
  e2: the telephone rings
- cases
  e1 occurs after e2 causal dependency possible
  e2 occurs after e1 causal dependency unlikely

Temporal order
is necessary but not sufficient to establish causal order.

Another Example



Imperfect Timestamps can be misleading in establishing causal dependency.

Causal Order (for Computer Generated Events)

Partial Order for Computer Generated Events

a —> b "a causes b" (happened before, causally dependent)

(1) If a, b events within a sequential process
then a —> b, if a is executed before b.

(2) If a is „sending of a message" by a process and
b the „reception of that message" by another process,
then a ——> b.

(3) —> is transitive.

Temporal Order

Modeling the continuum of time: infinite set of instants {T}

- {T} is *ordered*:
  if p, q any 2 instants, then either p,q simultaneous
  (i.e. the same instant),
  or (exclusive) p precedes q, or q precedes p

- {T} is *dense*:
  at least 1 instant q between p and r
  iff p and q are not simultaneous

Instants are totally ordered ...

Temporal Order, Timestamps, Duration, Clocks

... Instants are totally ordered

Events occur at an instant of the timeline => *Timestamp*.

Events in a distributed system are partially ordered.

Total order can be achieved by addition of process id.

Duration is a section of the timeline.

Clocks measure time imperfectly, create imperfect Timestamps.

Clocks: Physical and Logical

Physical Clocks

- devices to measure time
- necessarily imperfect (more later)

Problems:

- **how to create knowledge about causal dependency of computer events without relying on physical clocks ?**
  => *Logical Clocks*
- **how to establish a *x certainly occurred after y* relation (temporal order) for environmental events**
  => *Global Time*

Logical Clocks

Definitions:

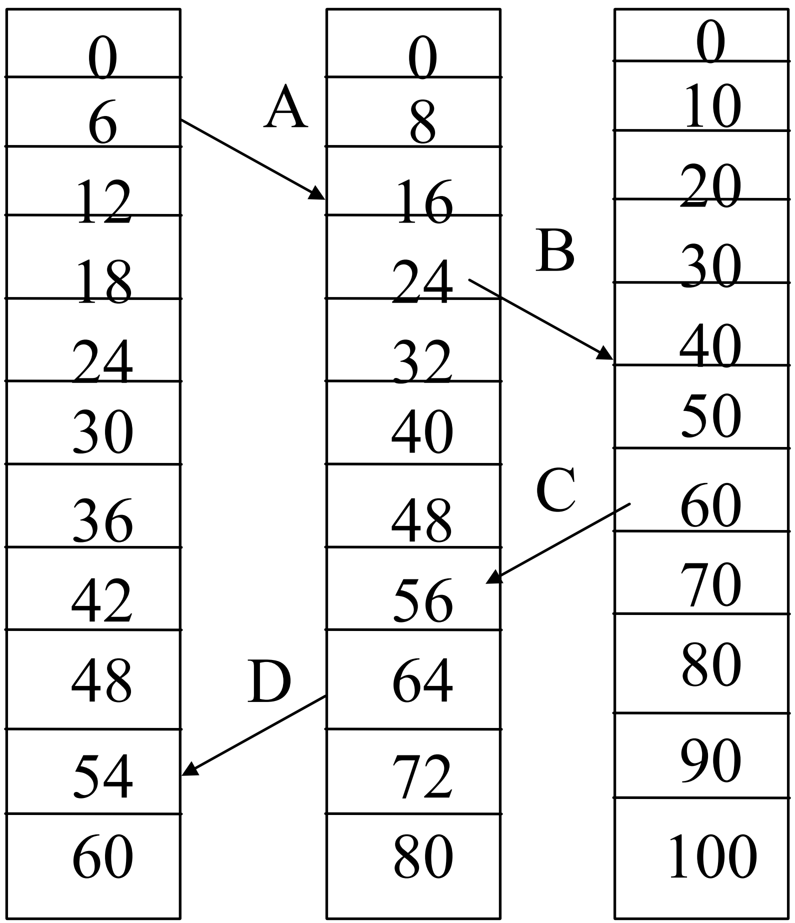
- monotonically increasing SW counters (COULOURIS)
- clocks on different computers that are somehow consistent (LAMPORT)

Events: a,b: a —> b: a *causes* b (causally dependent)

Timestamps: C(a), C(b)

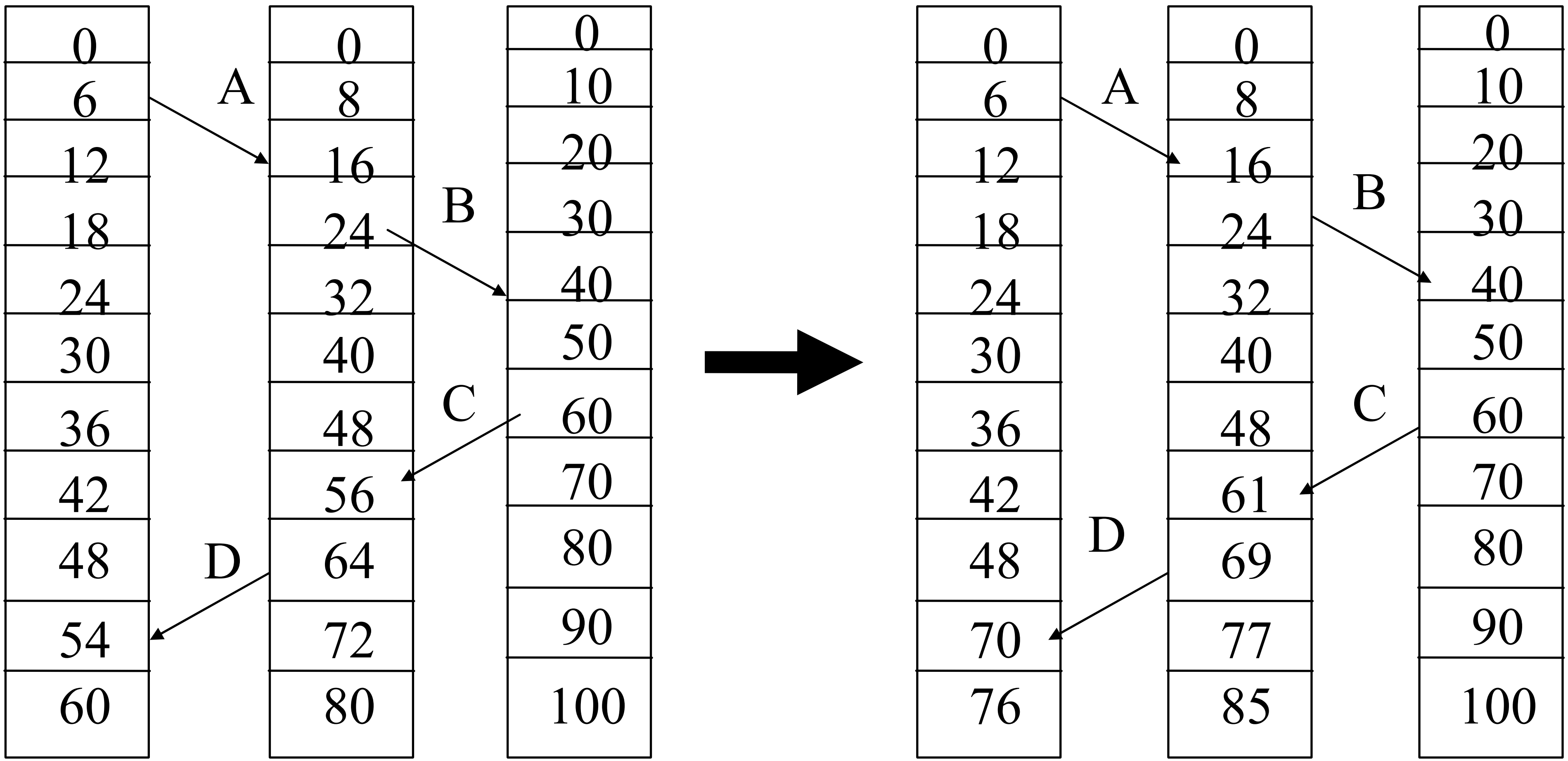
Potential Requirements for logical clocks:

- **a —> b => C(a) < C(b)**
- **a —> b <=> C(a) < C(b)**


0
6
12
18
24
30
36
42
48
54
60
A
0
8
16
24
32
40
48
56
64
72
80
B
C
D
0
10
20
30
40
50
60
70
80
90
100

Lamport‘s Logical Clocks

- **each Process has local clock LC_i**
- **tick:**
  - with each local event e:
    $LC_i := LC_i + 1;\ e$
  - with each sending of a message by process Pi:
    $LC_i := LC_i +1;\ \ send\ (LC_i,m)$
  - with each reception of a message „(M,LC_m)" by Pj:
    $LC_j := MAX(LC_m, LC_j);\ LC_j := LC_j +1$


0
6
12
18
24
30
36
42
48
54
60
A
D
0
8
16
24
32
40
48
56
64
72
80
B
C
0
10
20
30
40
50
60
70
80
90
100
0
6
12
18
24
30
36
42
48
70
76
A
D
0
8
16
24
32
40
48
61
69
77
85
B
C
0
10
20
30
40
50
60
70
80
90
100

Lamport Clocks

Properties:

- $a \longrightarrow b \Rightarrow C(a) < C(b)$,
  but not:
  $C(a) < C(b) \Rightarrow a \longrightarrow b$
- partial order

sometimes total order convenient:
extend it: e.g. by LC.Process_number

Vectortime (Mattern 1989)

Each process P_i has its own vector clock C_i.

C_i: n-dimensional vector (n: number of processes).

<u>Intuition</u>

$C_i[j]$:

the timestamp of the last event in P_j
by which P_i has potentially been effected

Vectortime Ticks

Initial:

$C_i := (0, \ldots, 0)$ for all i

Local event in P_i:

$C_i[i] := C_i[i] + 1;$

Sending message m in P_i:

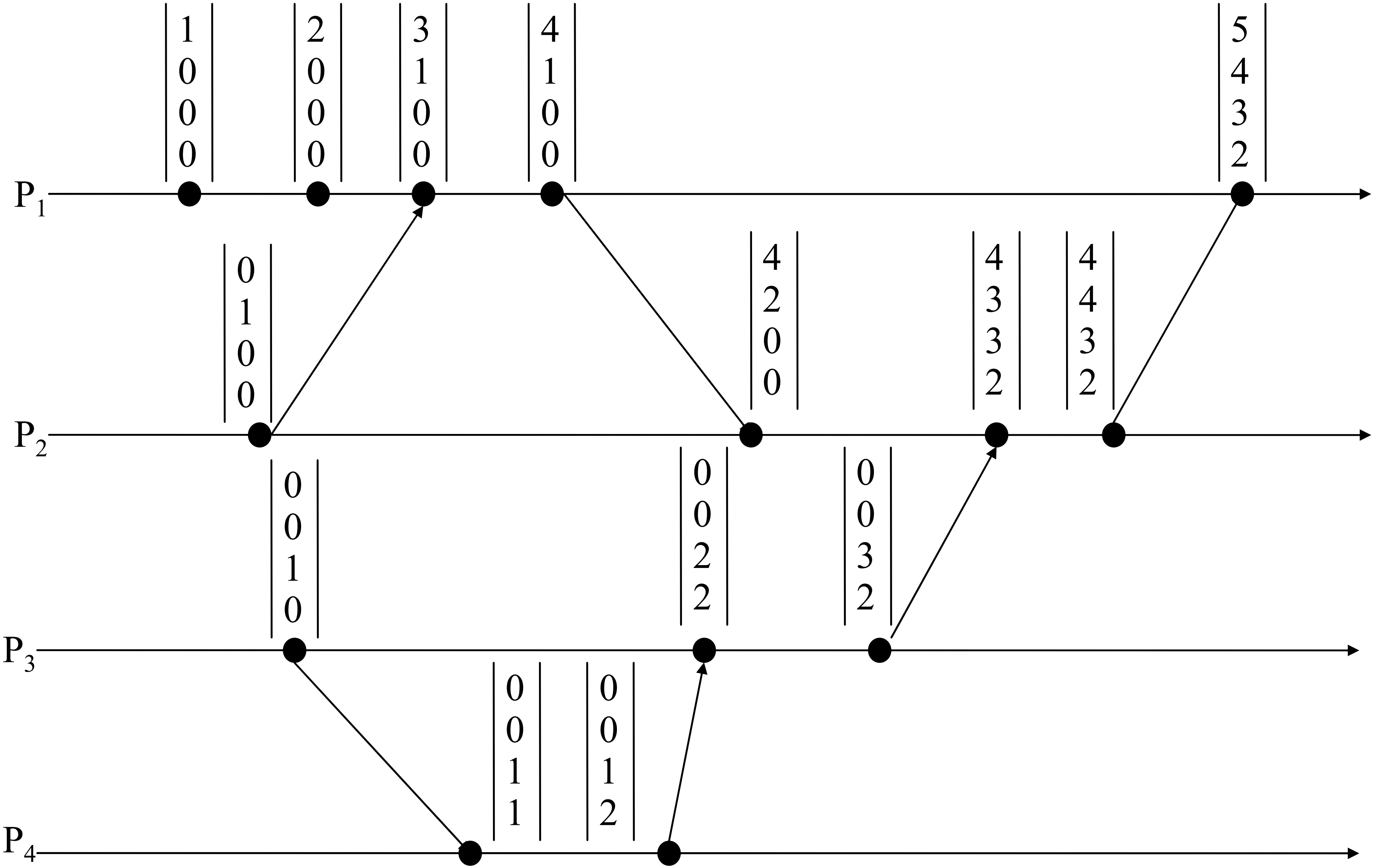
$C_i[i] := C_i[i] + 1;$ send (m, C_i)

Receiving a message „(m, C_m)" in P_j:

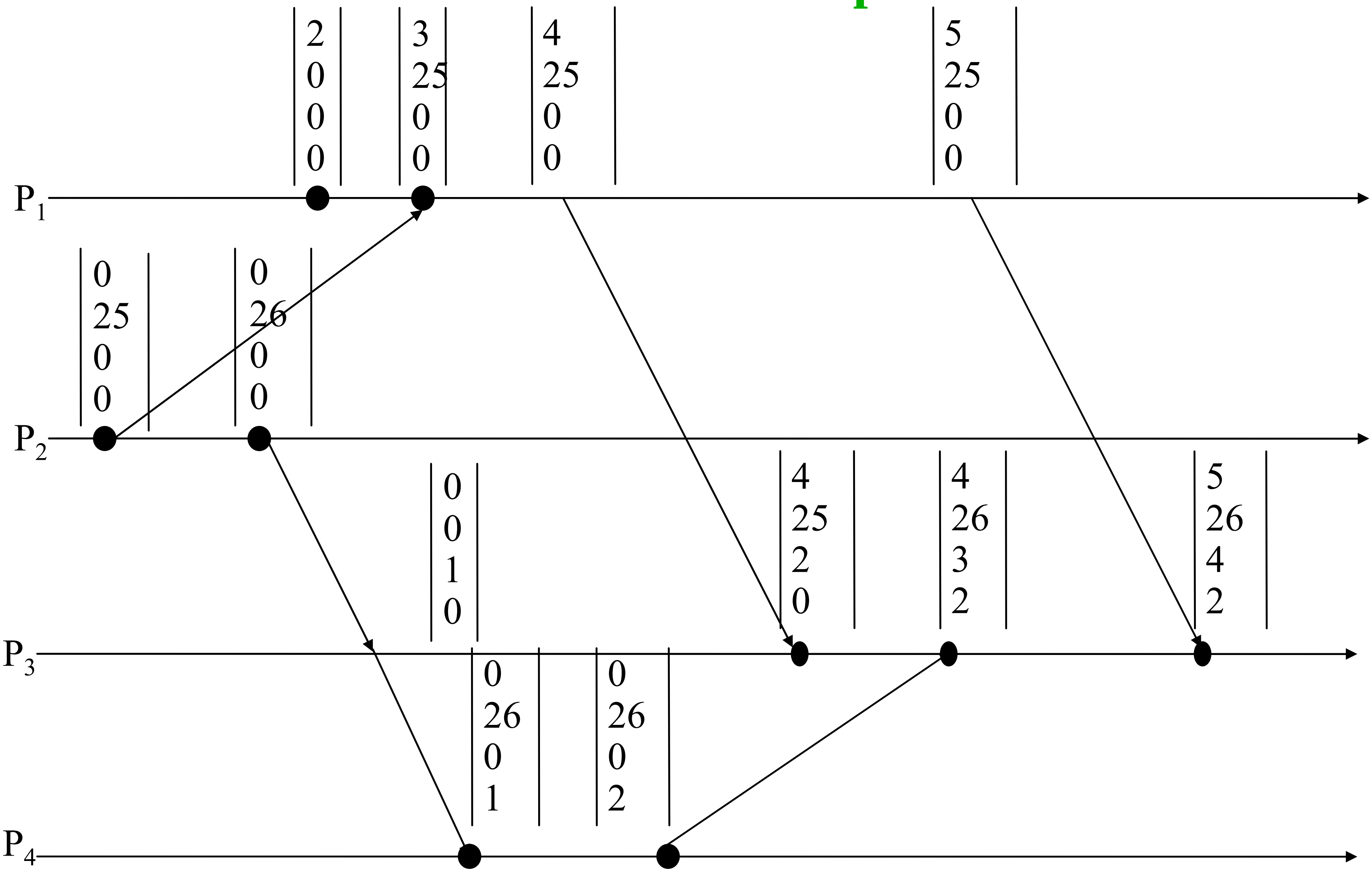
$C_j[j] := C_j[j] + 1;$

$C_{j[k]} := \max(C_m[k], C_j[k])$, for all k

Example


1 0 0 0
2 0 0 0
3 1 0 0
4 1 0 0
5 4 3 2
P1
0 1 0 0
4 2 0 0
4 3 3 2
4 4 3 2
P2
0 0 1 0
0 0 2 2
0 0 3 2
P3
0 0 1 1
0 0 1 2
P4

Another Example



Properties of Vector Time

Definitions:

- $$C_a \le C_b :\Leftrightarrow \forall k : C_a[k] \le C_b[k]$$

- $$C_a < C_b :\Leftrightarrow C_a \le C_b \text{ and } C_a \ne C_b$$

$$C_a || C_b :\Leftrightarrow \quad \text{not } (a < b) \text{ and not } \quad (b < a)$$

Property:

$$C_a < C_b \Leftrightarrow a \to b$$

Mathematics: Quantitative Aspekte (Claude-J. Hamann)

Physical Clocks and Their Properties

Physical Clock

- device for measuring time
- counter + oscillator => „microtick"
- time between microticks:
  granularity leads to digitalization error

Notation:

g^{clock}, $microtick^{clock}_{number\ of\ tick}$

To discuss properties of physical clocks,
we invent the perfect *reference clock*

Reference Clock, Notations (Kopetz)

Reference Clock z

- perfect with regard to UTC
- very small granularity
  (to disregard digitalisation error)
- Reference Ticks:Ticks of the perfect reference clock

z(event): (Absolute) Timestamp from reference clock
establishes temporal order

g^k granularity of clock k in
microticks of ref. clock

Reference Clock, Notations (Daum)

Reference Clock z

- perfect with regard to UTC
- dense (no ticks, to avoid digitalisation error)

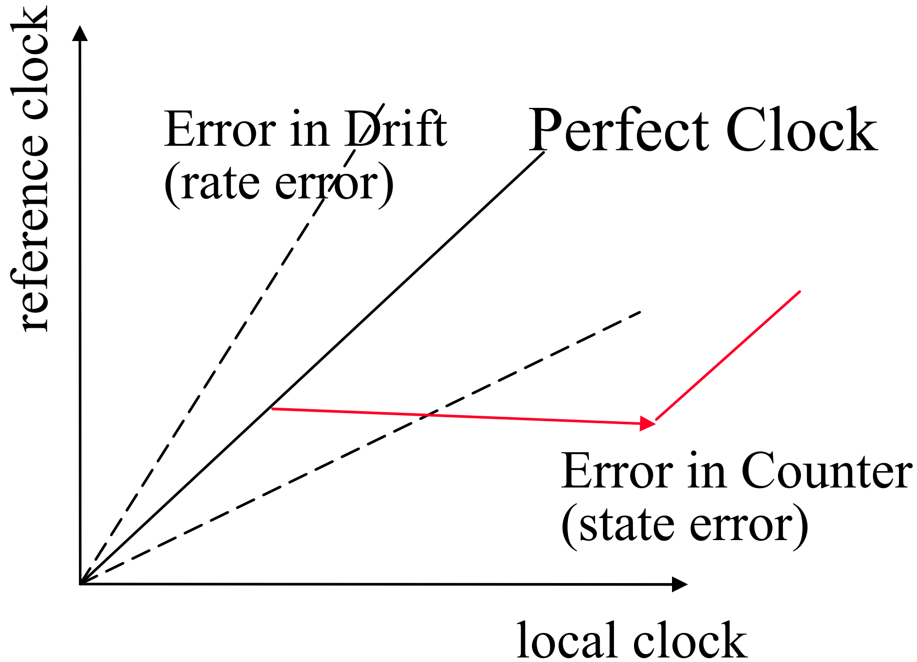
z(event): (Absolute) Timestamp from reference clock establishes temporal order

g^k granularity of clock k in terms of z-durations as specified (vs. real behavior)

Tick Tack Terms

Micro Ticks	Ticks generated by the physical oscillator of a clock
Macro Ticks	Multiple of Micro Ticks chosen by designer of clock
t^k(event):	Timestamp in number of macro ticks of clock k
Granularity	Unit of a clock Value exposed by a clock micro or macro (dep on context)

Failure Modes of Clocks: Drift and Counter Errors



Maximum Drift Rate

- Drift-Rate:

$$\rho_i^k = \left| \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{g^k} - 1 \right|$$

- varying
- influenced by environmental conditions (temperature,...)
- clocks specify maximum drift rate (10^{-2} ... 10^{-7})

Precision of an Ensemble of Clocks

Offset

between two clocks j,k of same granularity at microtick i:

$$offset_i^{jk} = \left| z(microtick_i^j) - z(microtick_i^k) \right|$$

in the period of interest: $$offset^{jk} = \max_i \left(offset_i^{jk} \right)$$

Precision

of an ensemble of clocks {1,2, ...,n} in the period of interest:

$$\Pi = \max_{1 \le j,k \le n} \left(offset^{jk} \right)$$

maximum offset for any two clocks

Accuracy

Accuracy

of a given clock in the period of interest:

$$accuracy^k = \max_i \left| z(microtick_i^k) - i \cdot g^k \right|$$

If all clocks of an ensemble have accuracy A,

the precision of the ensemble is ??

Resynchronisation

External Resynchronization

resynchronization with reference clock

Internal Resynchronization

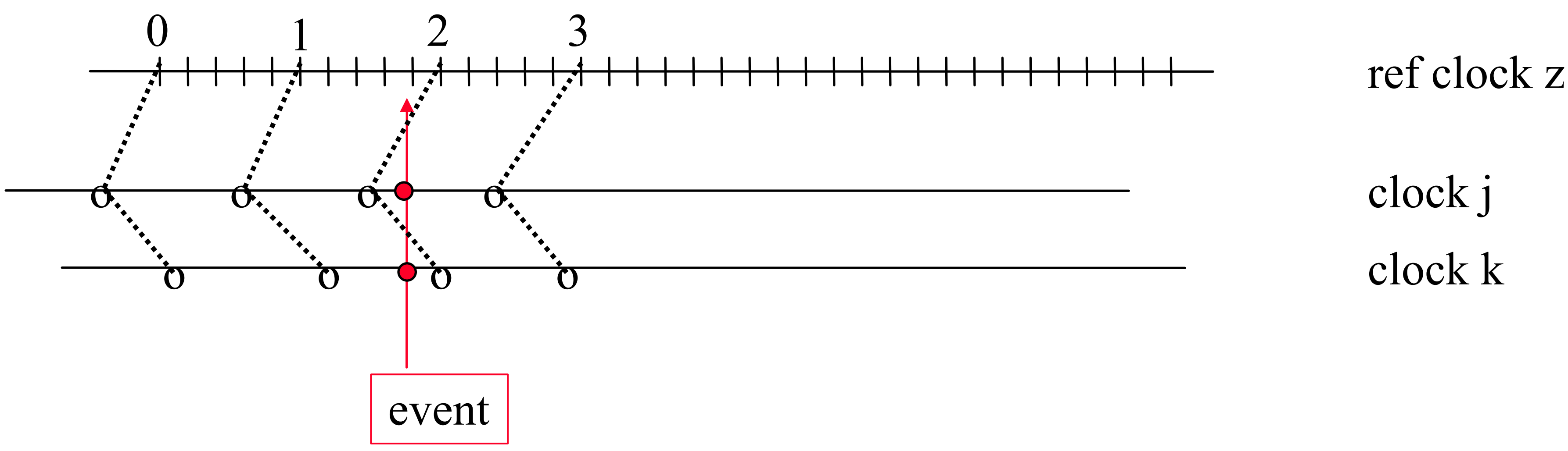
mutual resynchronization of an ensemble to maintain a bounded precision

Global Time

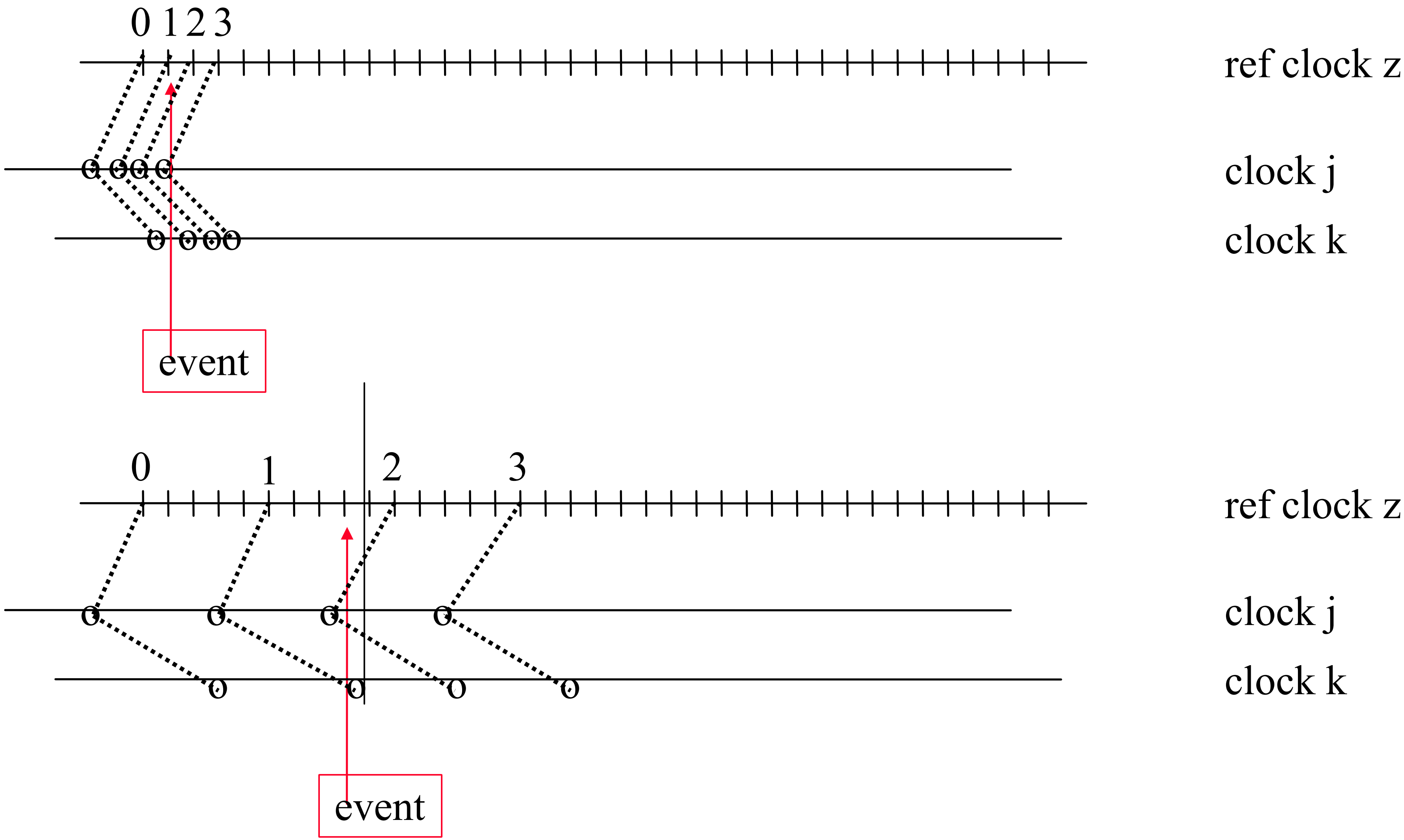
Given an ensemble of clocks
(internally) synchronized with precision Π ...

For each clock select
macrotick as local implementation of a global notion of time
with granularity g^{global}

We note ref clock time (real-time, UTC) in units of g^{global}



Examples for Bad Choice for Global Time



Reasonable => One Tick Difference

Reasonableness Condition:

global time t is <u>reasonable</u> if

$g^{global} > \Pi$ holds for all local implementations

t^j(event):

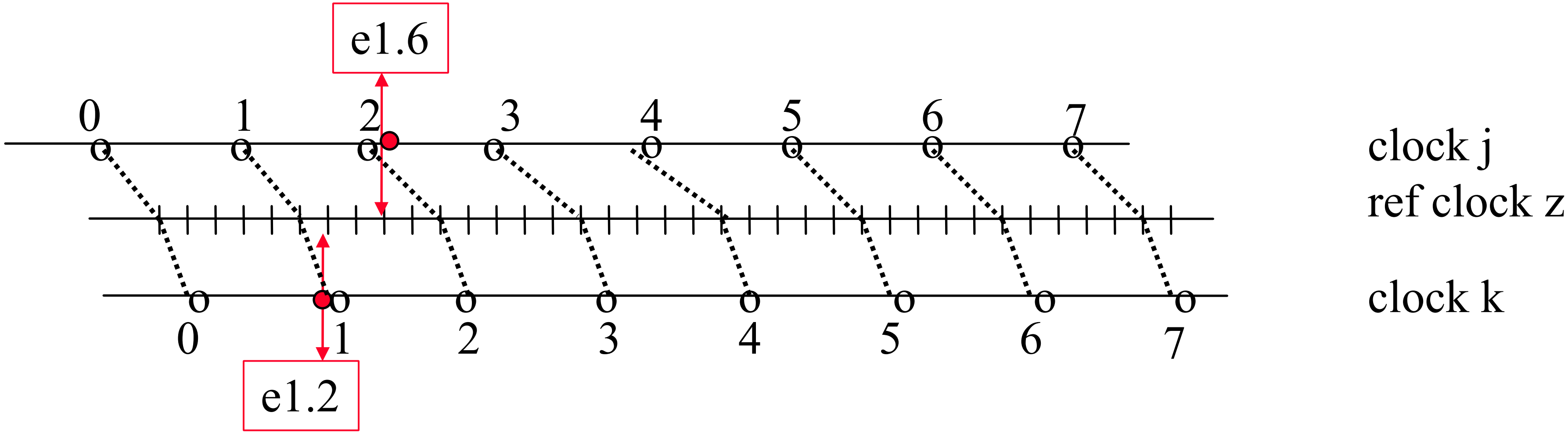
denotes global time for the implementation at clock j

Then: For any single event e, holds $\left| t^j(e) - t^k(e) \right| \leq 1$

Global timestamps differ at most by one (macro-)tick.

Best one can achieve!

Interpretation for Temporal Order



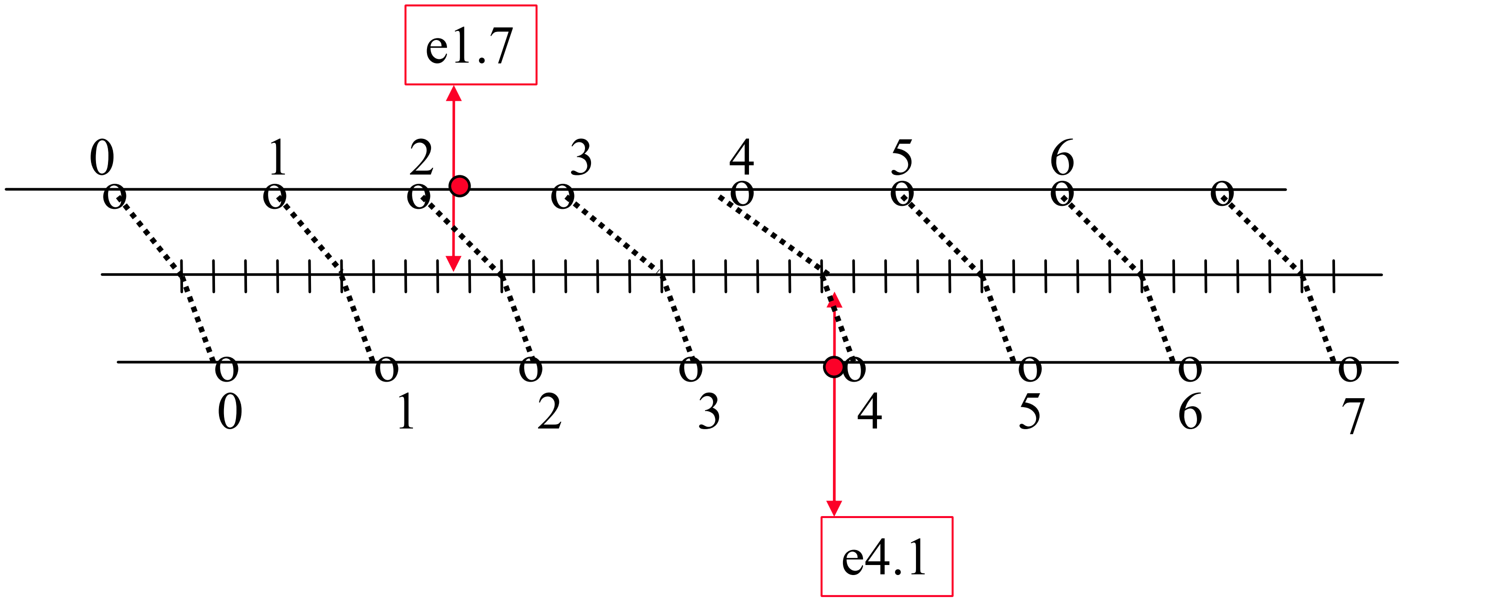
z(e1.6) - z(e1.2): 0.4 g^{global} ref clock

t^j(e1.6) - t^k(e1.2): 2 global time ticks

Temporal order can be established because
$Tick^k_1$ must be before $Tick^j_2$ (Reasonabless Condition)

Hence: If the (global) timestamps differ by two ticks, the temporal order can be established.

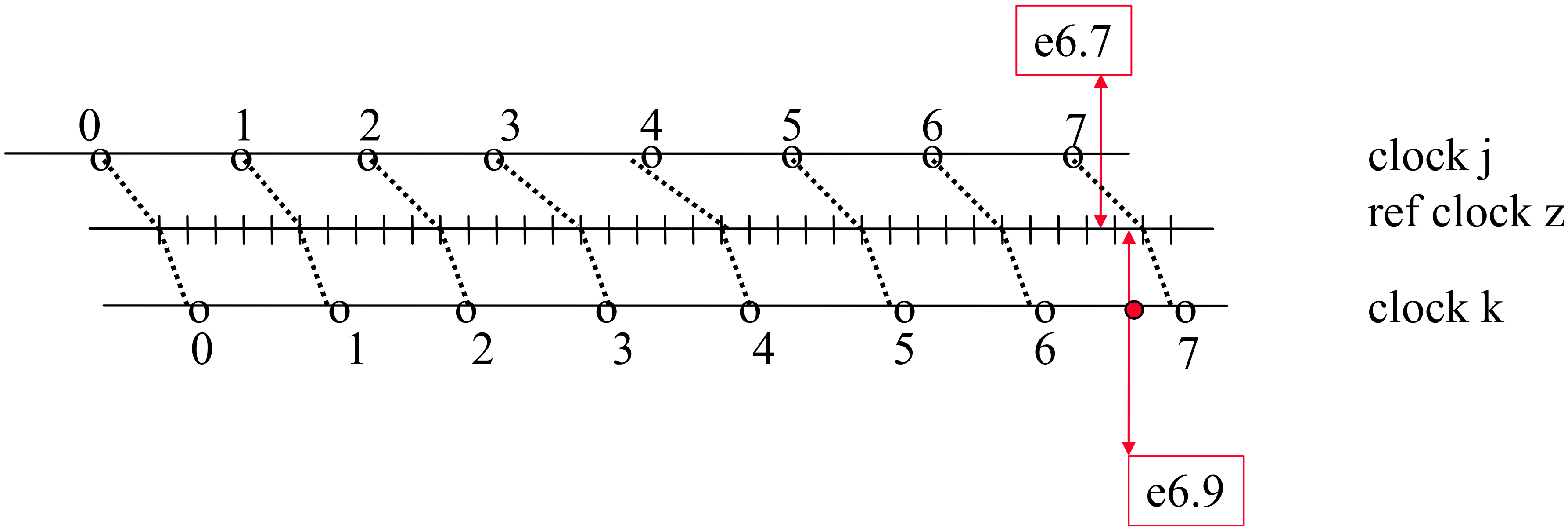
Caution: Example



z(e4.1) - z(e1.7):	2.4	g^{global} ref clock
t^k(e4.1) - t^j(e1.7):	1	global tick

A distance of 2*g^{global} between two events does not suffice to determine temporal order.

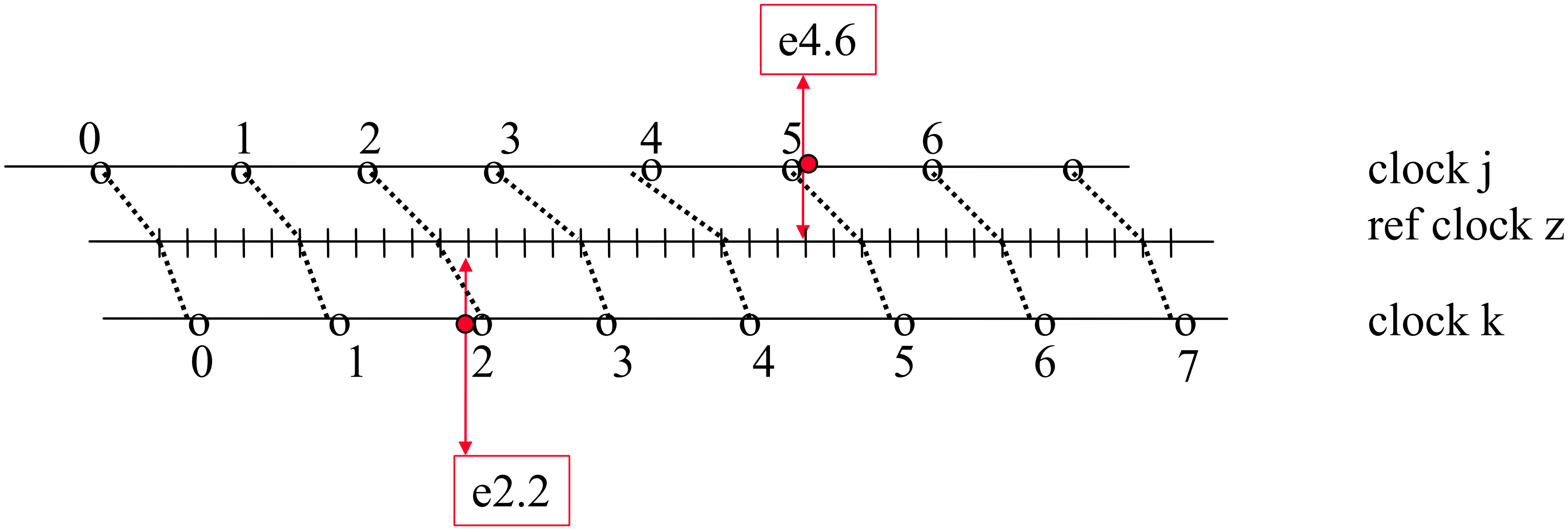
Another Example



$z(e6.9) > z(e6.7)$

$t^k(e6.9) < t^j(e6.7)$

Interpretation for Durations



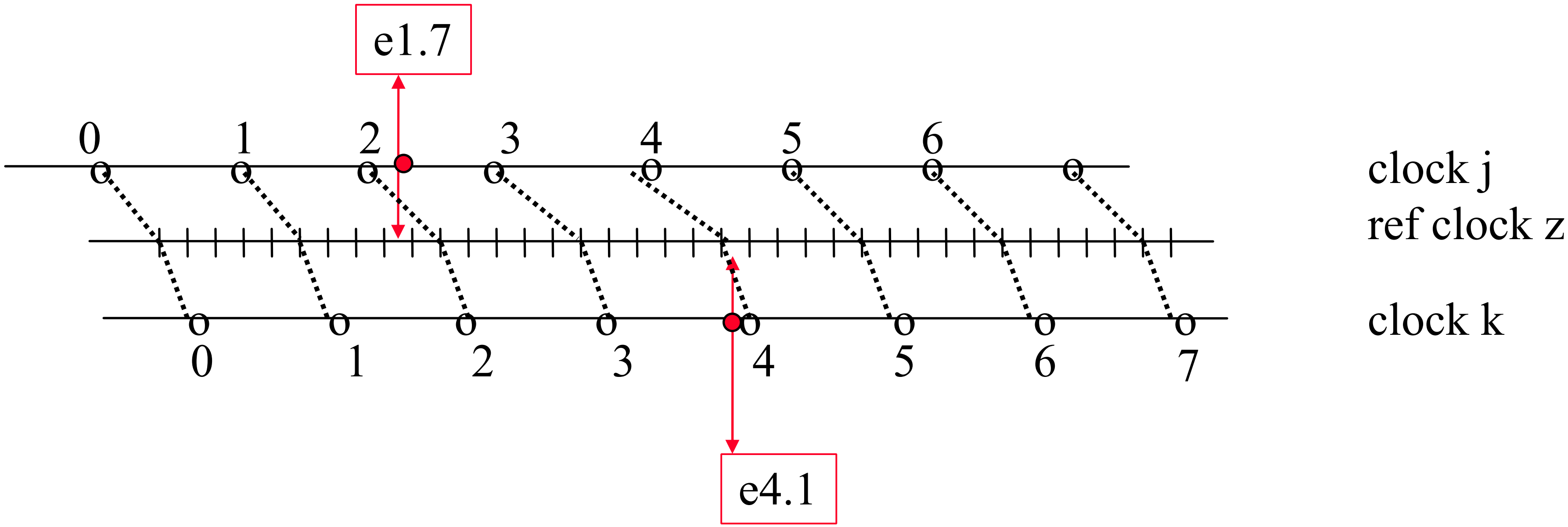
True duration: 2.4

Observed duration d: 5-1=4 ($t^j(e4.6) - t^k(e2.2)$)

Can be driven to true duration: 2+ε for small ε

$d_{obs} - 2*g^{global} < d_z$

Interpretation for Durations



True duration: 2.4

Observed duration d: 3-2=1 ($t^k(e4.1) - t^j(e1.7)$)

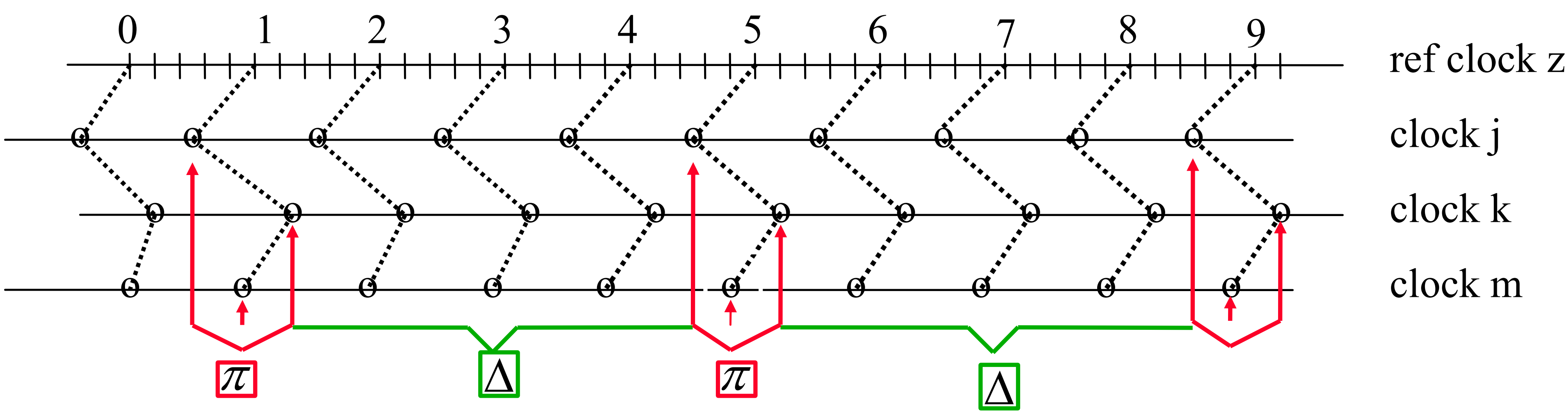
Can be driven to true duration: 3-ε for small ε

$$d_{obs} - 2*g^{global} < d_z < d_{obs} + 2*g^{global}$$

Generated Events

Cluster of three nodes:
each generates event at the same global tick
 t= 1, 5, 9

observation:



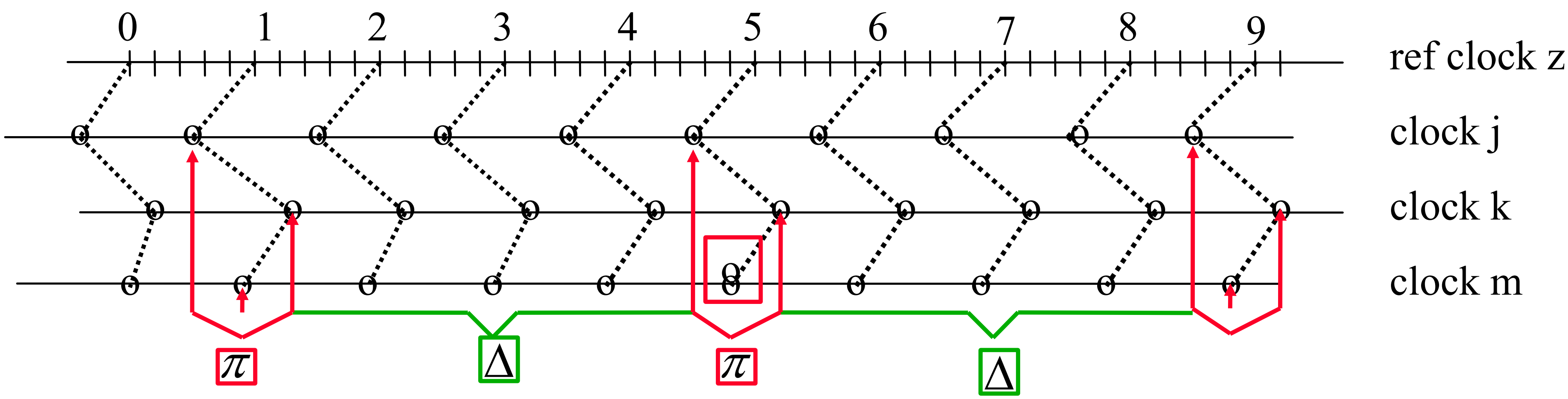
π/Δ-Precedence of Sets of Events

Properties of sets of events:

How far apart (number of granules) must events be to enable reconstruction of order ....

A set of events is called π/Δ-precedent, if:

$$\left[\,|z(e_i)-z(e_j)|\le\pi\right]\vee\left[\,|z(e_i)-z(e_j)|>\Delta\right]$$



Temporal Order

Event Set	Observed timestamps of two nonsimultaneous events are always greater or equal to	Temporal order of the events can always be reestablished
0/1g precedent	$\left\|t^j(e_1) - t^k(e_2)\right\| \geq 0$	no
0/2g precedent	$\left\|t^j(e_1) - t^k(e_2)\right\| \geq 1$	no
0/3g precedent	$\left\|t^j(e_1) - t^k(e_2)\right\| \geq 2$	yes
0/4g precedent	$\left\|t^j(e_1) - t^k(e_2)\right\| \geq 3$	yes

Fundamental Results in Time Measurement

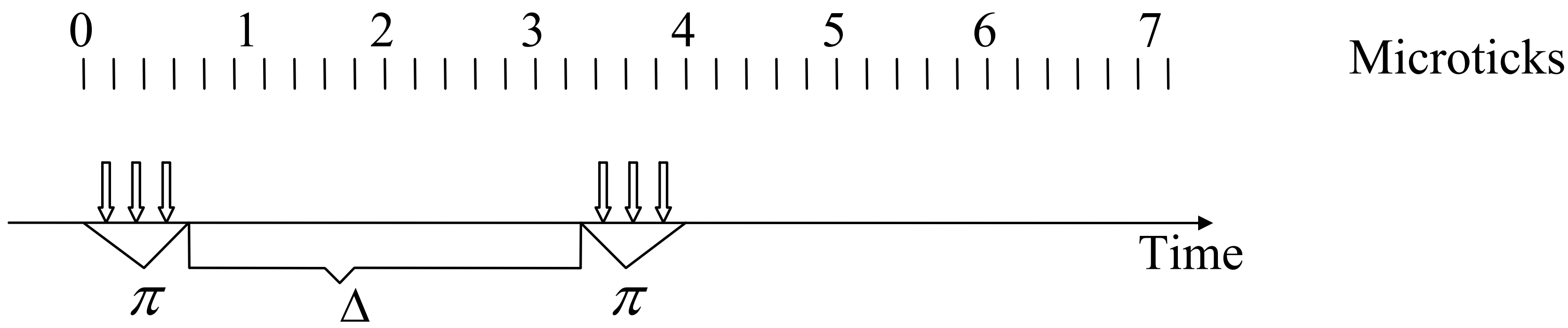
- A single event observed at different nodes may have timestamps differing by one; not sufficient to establish temporal order

- Duration: $d_{obs} - 2*g^{global} < d_z < d_{obs} + 2*g^{global}$

- temporal order can be recovered from their timestamps if they differ by 2 global time ticks

- temporal order of events can always be recovered from timestamps if the event set is 0/3g precedent

Dense Time vs. Sparse Time

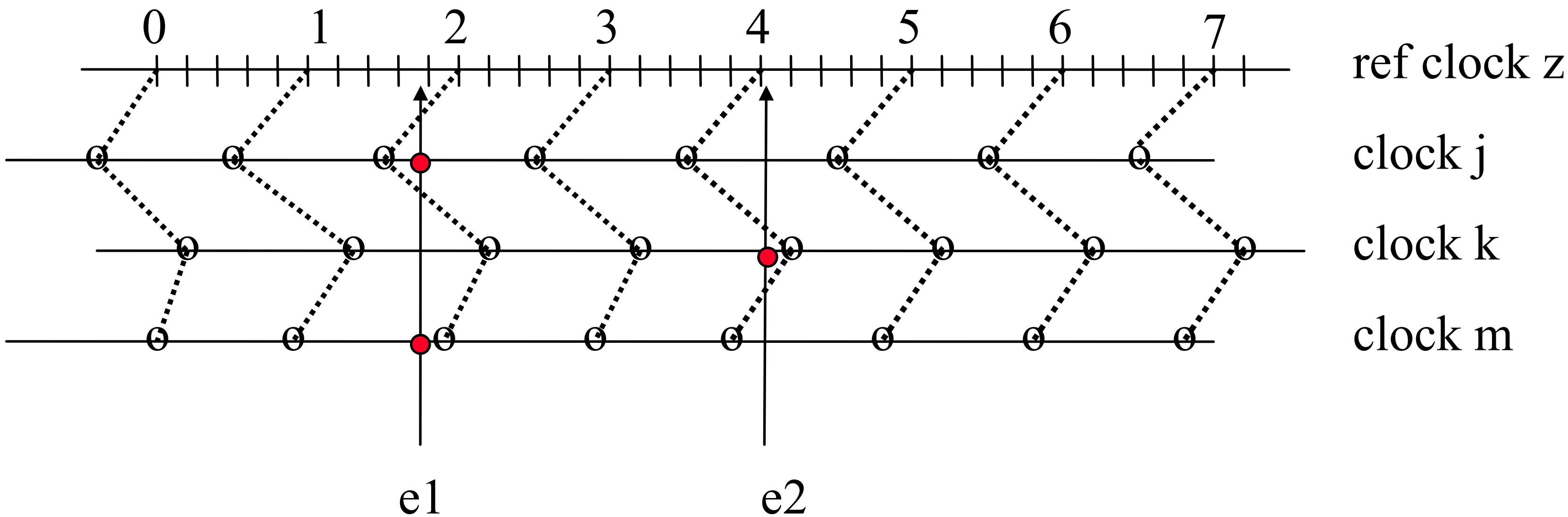


dense time: events are allowed at any time

sparse time: events are only allowed within *active* time intervals π

sparse time only possible for computer controlled events

Cooperation and Clocks



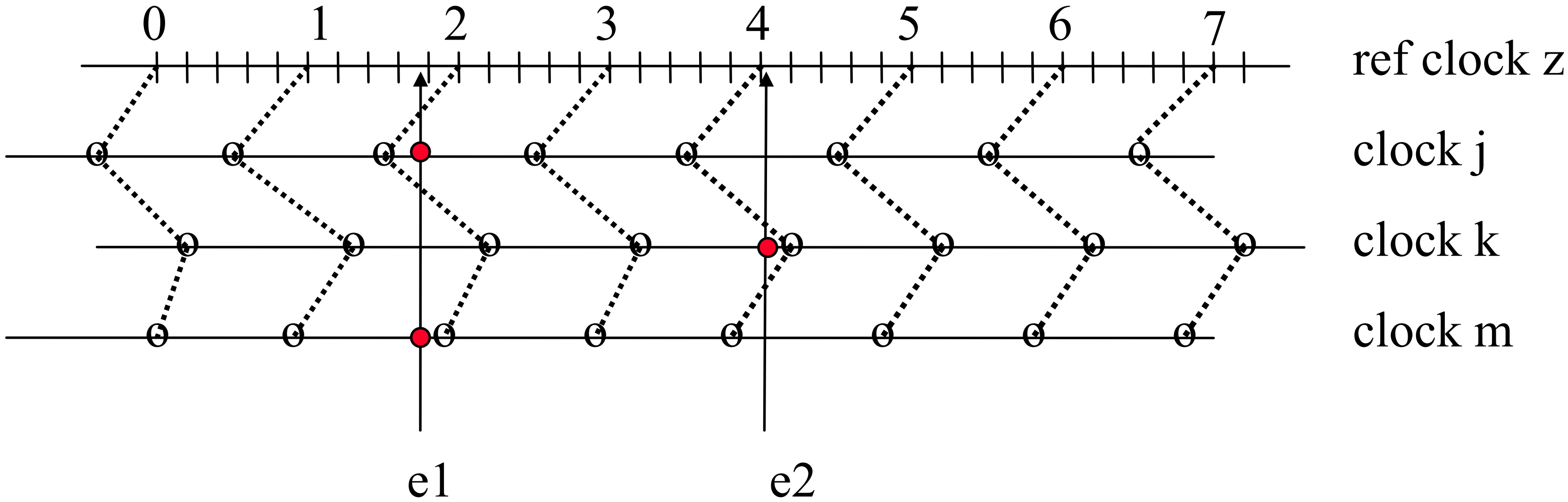
(only) nodes j and m can observe e1

(only) node k can observe e2

Node k tells nodes j and m about e2

Nodes j and m draw their conclusions …

Dense Time Requires *Agreement*



j observes e1 at t=2, m observes e1 at t=1

k observes e2 and reports to j and m: "e2 occurred at t=3"

j calculates a time difference of 1, hence concludes: "events cannot be ordered"

m calculates a time difference of 2, hence concludes: "events definitely ordered"

=> inconsistent view !!!

Agreement Protocols

- information interchange:
  each node acquires local *views* from all other nodes
- deterministic algorithm that lead to same result on all nodes

expansive !!

Sparse Time

Two clusters A,B with synch clocks of granularity g each, no clock synch between A and B

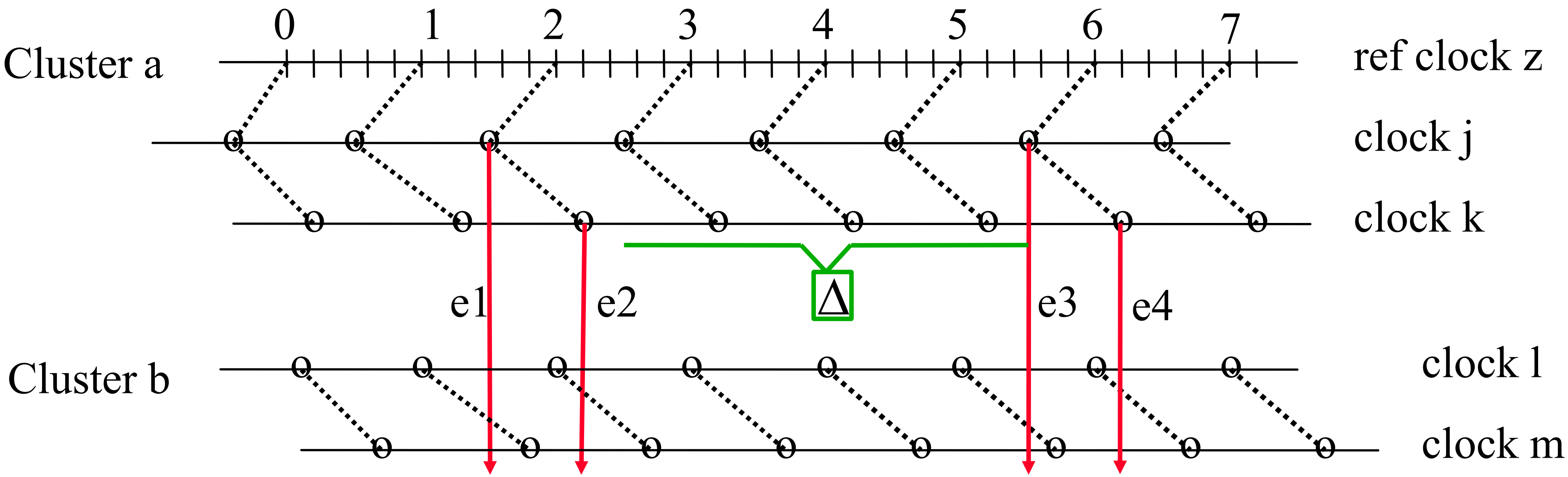
cluster A generates events, cluster B observes:

goal:

- if at cluster A events are generated at same cluster wide tick never should temporal order be concluded
- always establish temporal order otherwise

sufficient for A to generate 1g/3g precedent event set ??

Example for 1g/3g



$t^l(e2) - t^m(e1) = 2$:

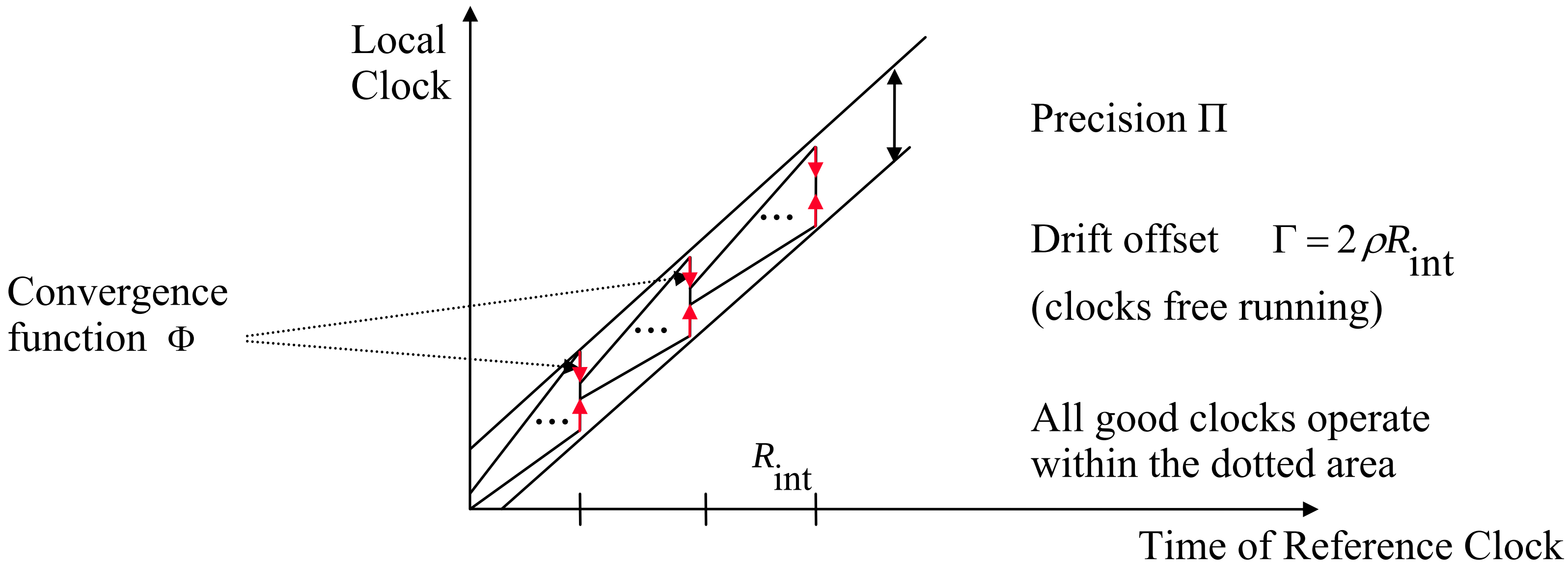
BUT: should not derive order because events were intended by cluster A for the same time

$t^m(e4) - t^l(e2) > 2$ BUT: $t^m(e3) - t^l(e2) = 2$:

BUT: temporal order is intended ($\Delta = 3g$)

=> 1g/3g precedence not sufficient => 1g/4g

Internal Clock Synchronisation



Precision Π

Drift offset $\Gamma = 2\rho R_{int}$

(clocks free running)

All good clocks operate within the dotted area

$$\Phi + \Gamma \leq \Pi$$

Synchronisation Condition

resynchronization interval: R_{int}

convergence function : ϕ, offset after resynch.

drift offset: Γ

$$\Gamma = 2\rho R_{\text{int}}$$

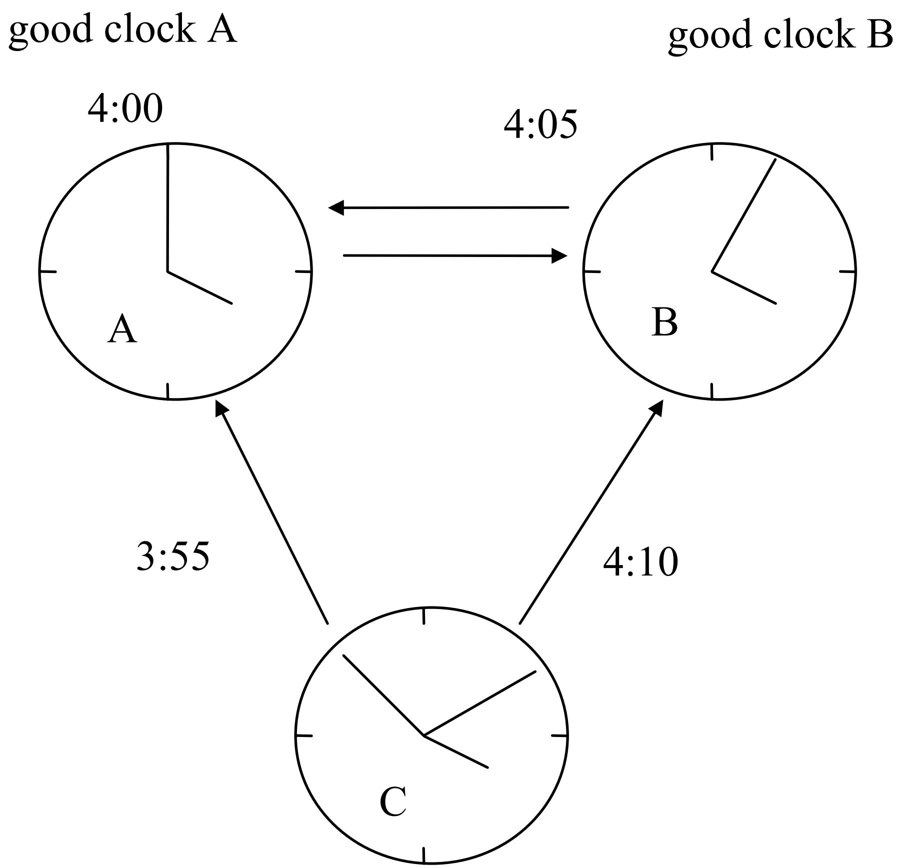
required:

$\Gamma + \Phi <= \Pi$

Byzantine Error

View at A

B	4:05
A	4:00
C	3:55



„two faced“ malicious clock C

View at B

A	4:00
B	4:05
C	4:10

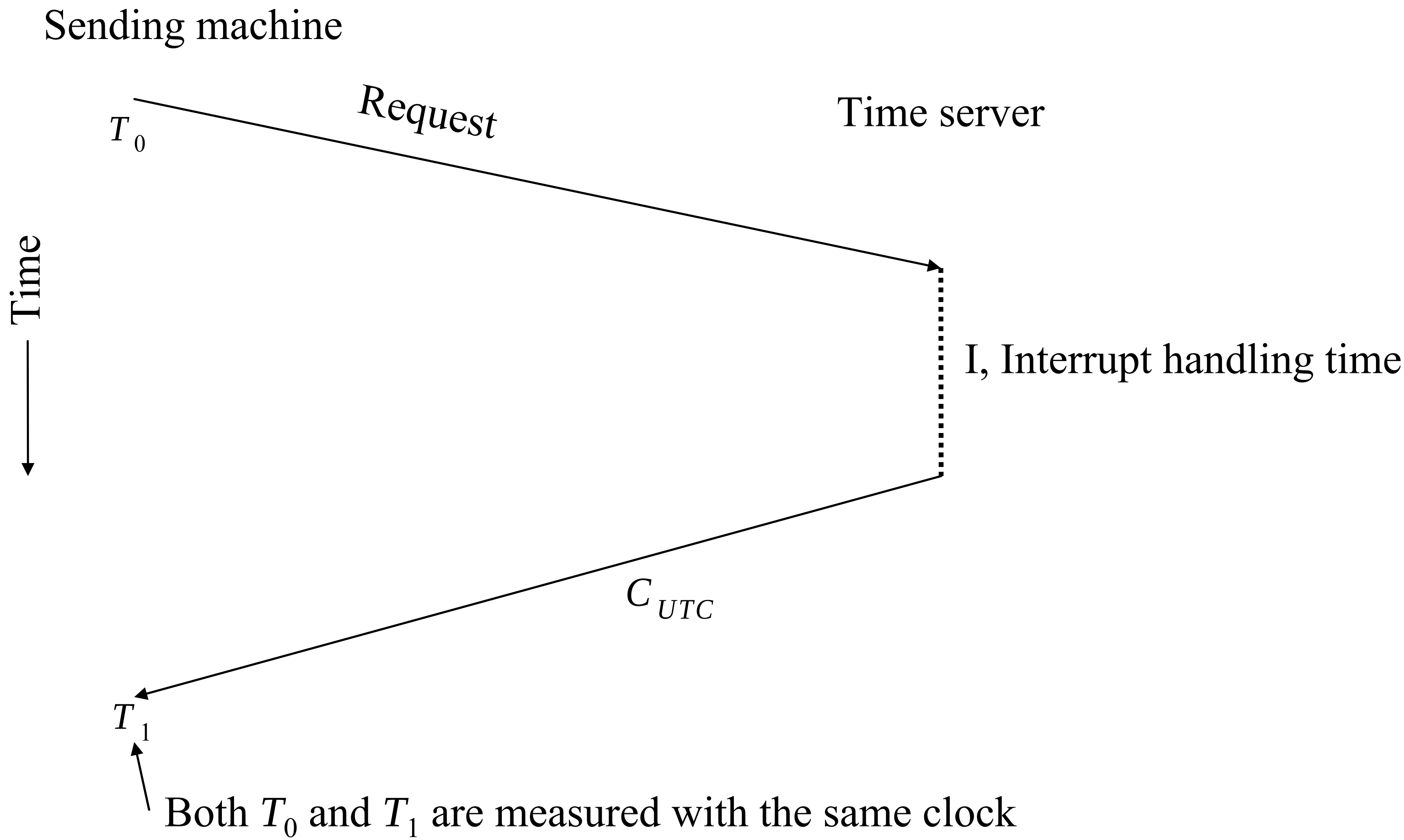
Central (Master) Synchronisation

master sends time, slaves correct

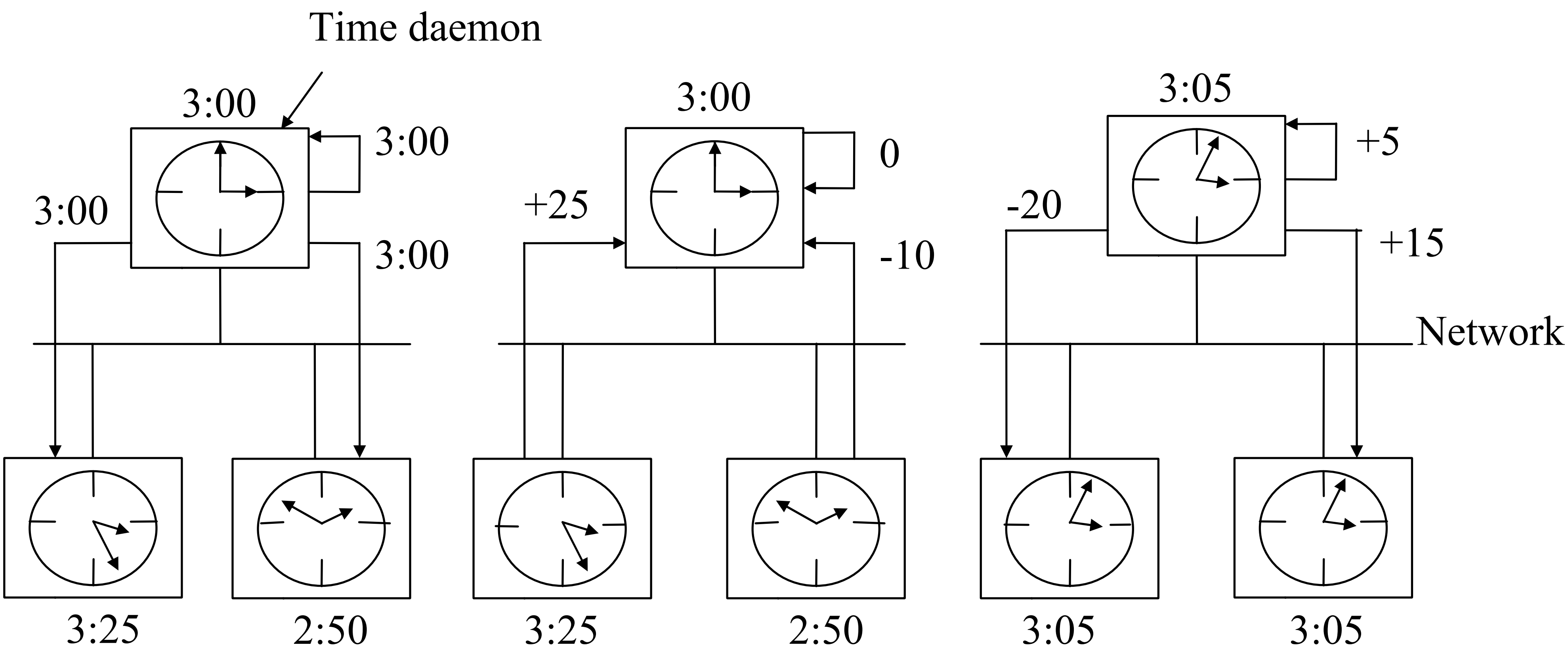
message latency jitter:
  ε, difference between fastest and slowest message

$\Pi_{central} = \varepsilon + \Gamma$

Distributed Synchronisation: Cristian



Distributed Synchronisation: Berkeley



Distributed Synchronisation: Kopetz' Tabelle

synchronisation message assembled and interpreted	approximate range of jitter
at the application software level	500 µs to 5 ms
in the kernel of the operating system	10 µs to 100 µs
in the hardware of the communication controller	less than 10 µs

Impossibility Result

$$\Pi = \varepsilon\left(1 - \frac{1}{N}\right)$$

No better precision can be achieved even with perfect clocks in all nodes (N number of nodes).

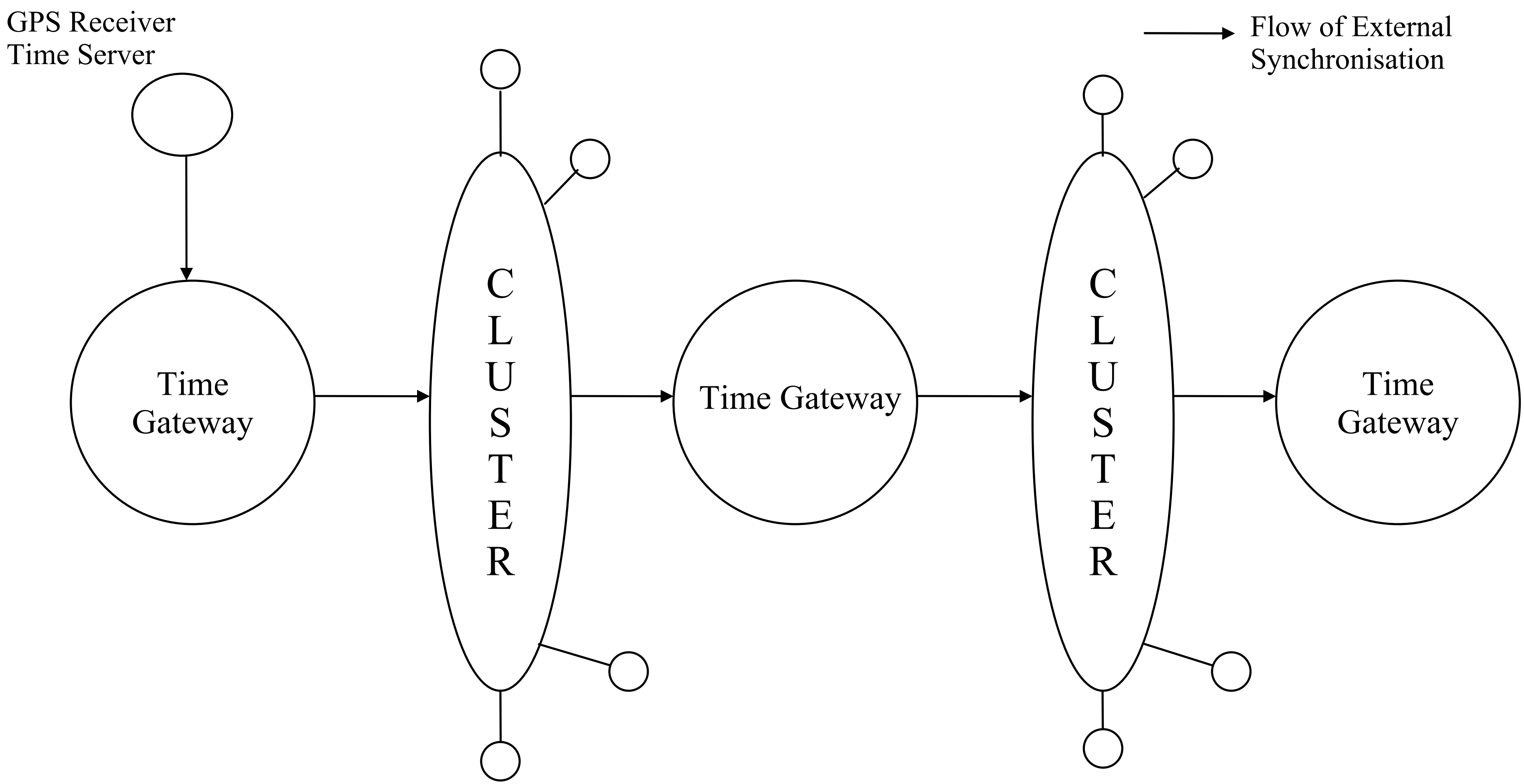
Correction: State vs. Rate

State: reset local clock

Rate: reset speed of clock

which should be used?

External Clock Synchronisation



Literature

Logical Clocks ...
  Standard text books: Coulouris, Tanenbaum, ...

Physical Clocks ...
  This lecture followed strictly
  Hermann Kopetz, Distr. RT-Systems

David Mills: Internet Time Synchronisation: the Network Time Protokol, IEEE Transactions Communic. 39,10 (Oktober 1991)