Real-Time Systems

Time and Order

(following Tanenbaum/Coulouris for Logical and Kopetz for Physical Time)

Hermann Härtig

WS 2016/17

TECHNISCHE UNIVERSITÄT DRESDEN

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 behavior / order.
- Consistent order
  - temporal order
  - causal order
- Global Time Base

Events in Computers

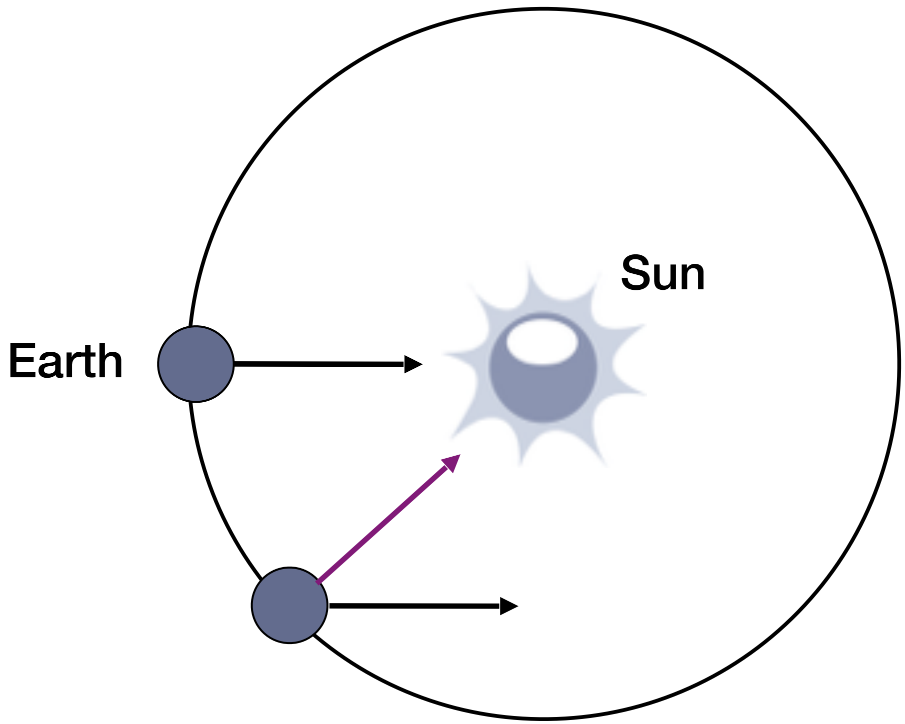
- Computer Generated Events:
  - execution of statement
  - sending/receiving a message
  - start and end of a compilation
  - creation/modification of a file
- Sequence of states is determined by
  - instructions, disk accesses
  - discrete steps

Events in the Real World

- 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"
- GPS clock is based on TAI

Time Standard(s)

- UTC ... Coordinated Universal Time
  - TAI adjusted with leap seconds to compensate for slowing earth rotation

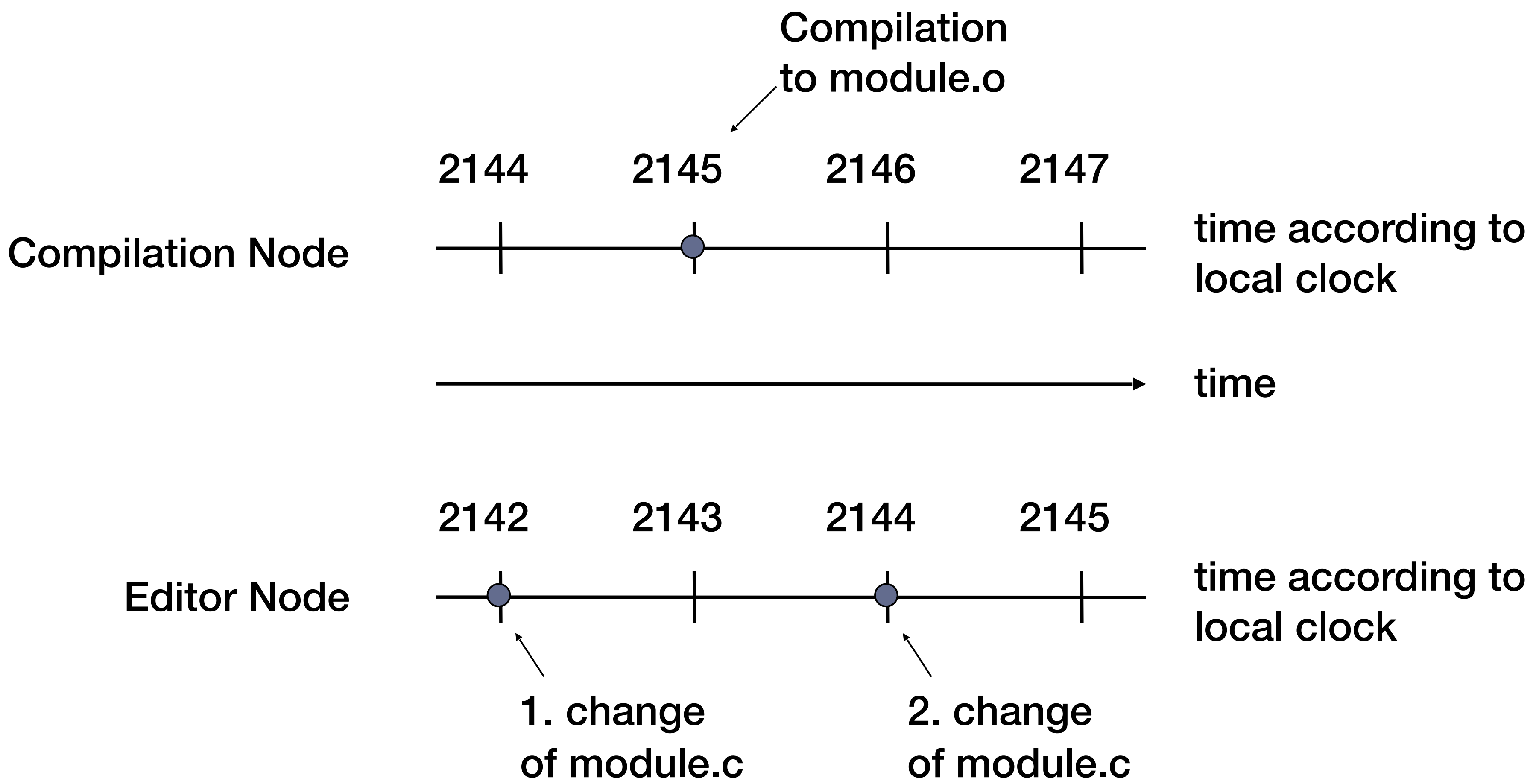
- Sources:
  - earth-bound radio
  - Geosynchronous satellites
  - GPS

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 (example by A.S. Tanenbaum)



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

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

- Events in a distributed system are partially ordered.

- Duration is a section of the timeline.
- Clocks measure time imperfectly, create imperfect Timestamps.

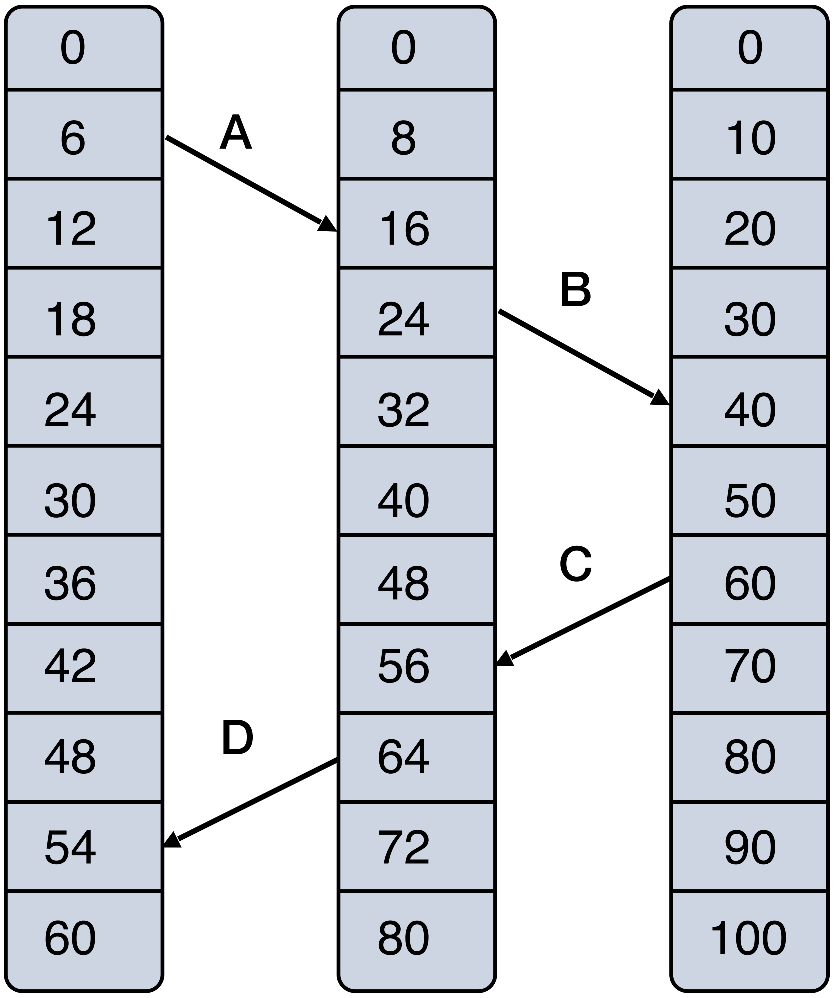
Clocks: Physical and Logical

- 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: $a \rightarrow b$: a causes b (causally dependent)
- Timestamps: C(a), C(b)
- Potential Requirements for logical clocks:
  - $a \rightarrow b \quad \Rightarrow \quad C(a) < C(b)$
  - $a \rightarrow b \quad \Leftrightarrow \quad C(a) < C(b)$

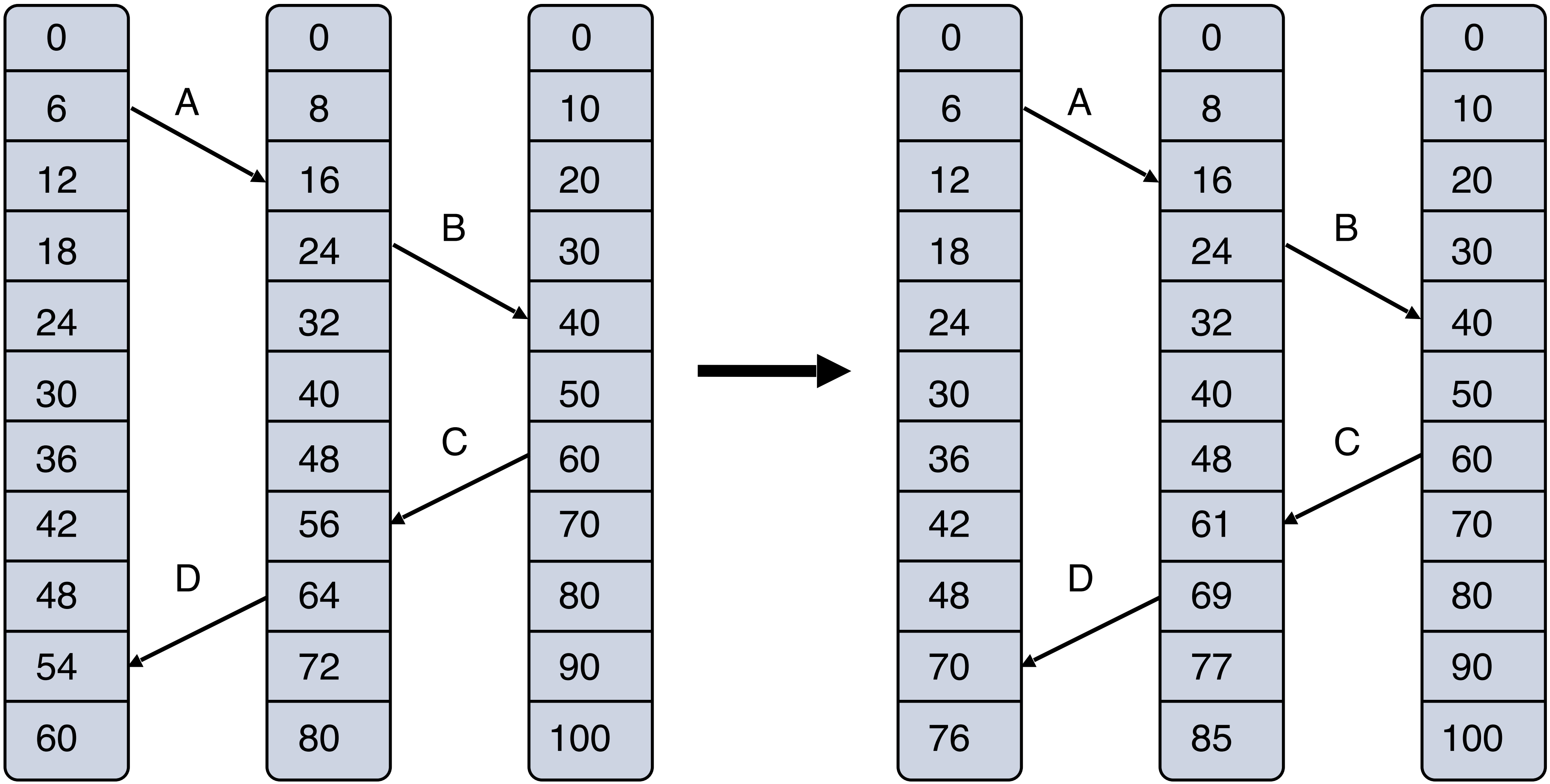
Logical Clock Example



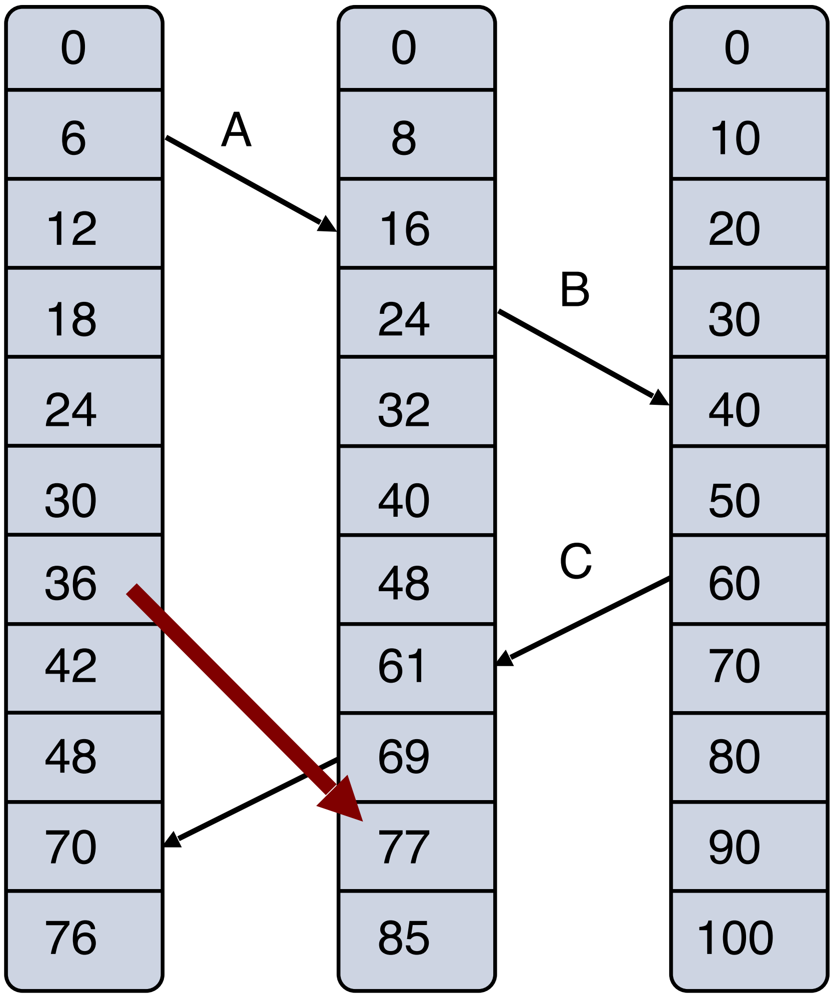
Lamport's Logical Clocks

- 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 P_i:
    $LC_i := LC_i + 1$; send(m, LC_i)
  - with each reception of a message (m, LC_m) by P_j:
    $LC_j := \max(LC_m, LC_j)$; $LC_j := LC_j + 1$

Lamport's Logical Clocks



Partial Timestamp Order



Lamport Clocks

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

Vector Time (Mattern 1989)

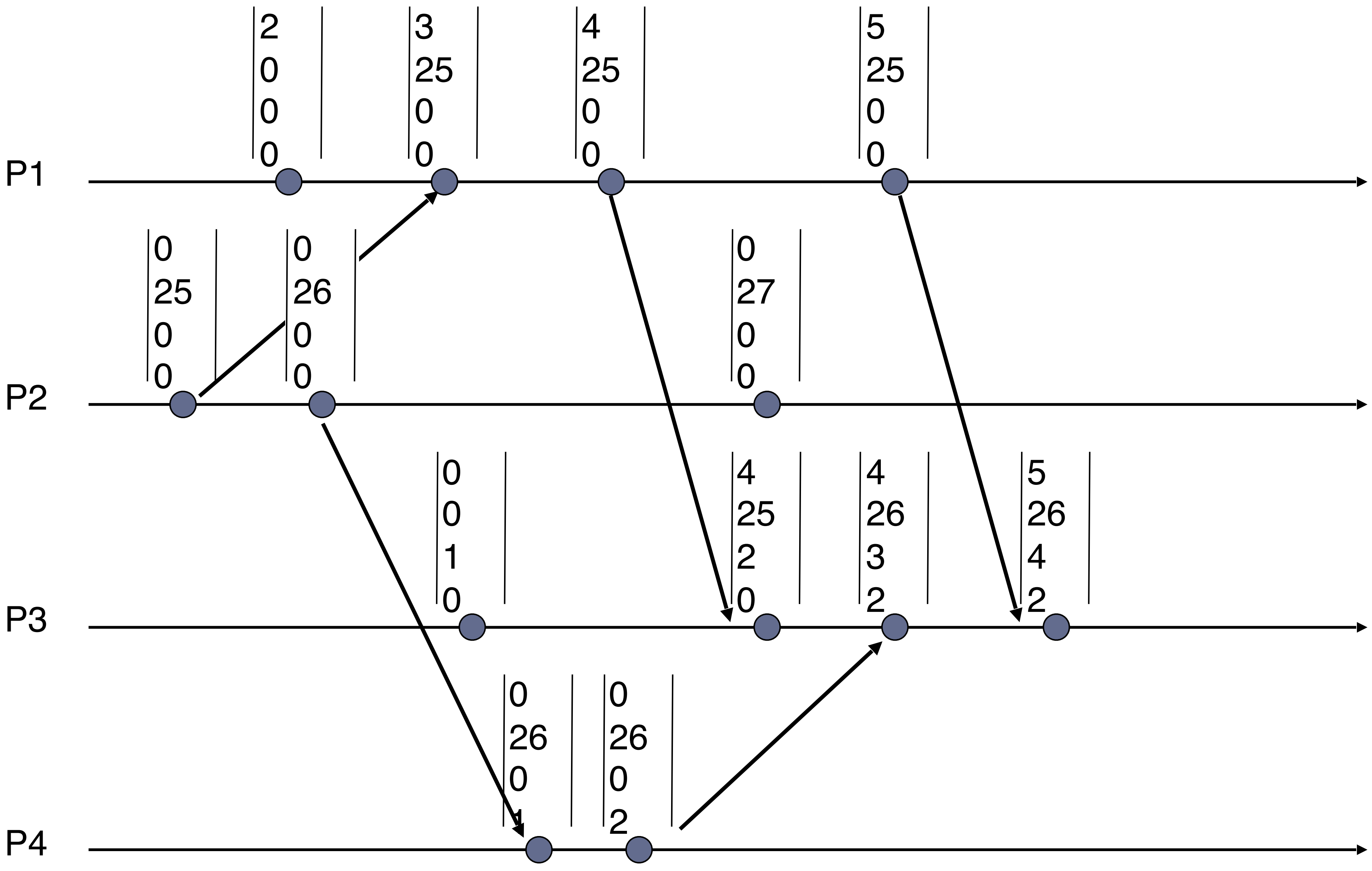
- Each process P_i has its own vector clock C_i.
- C_i: n-dimensional vector (n: number of processes).

- Intuition
- $C_i[j]$: the timestamp of the last event in P_j
  by which P_i has potentially been effected

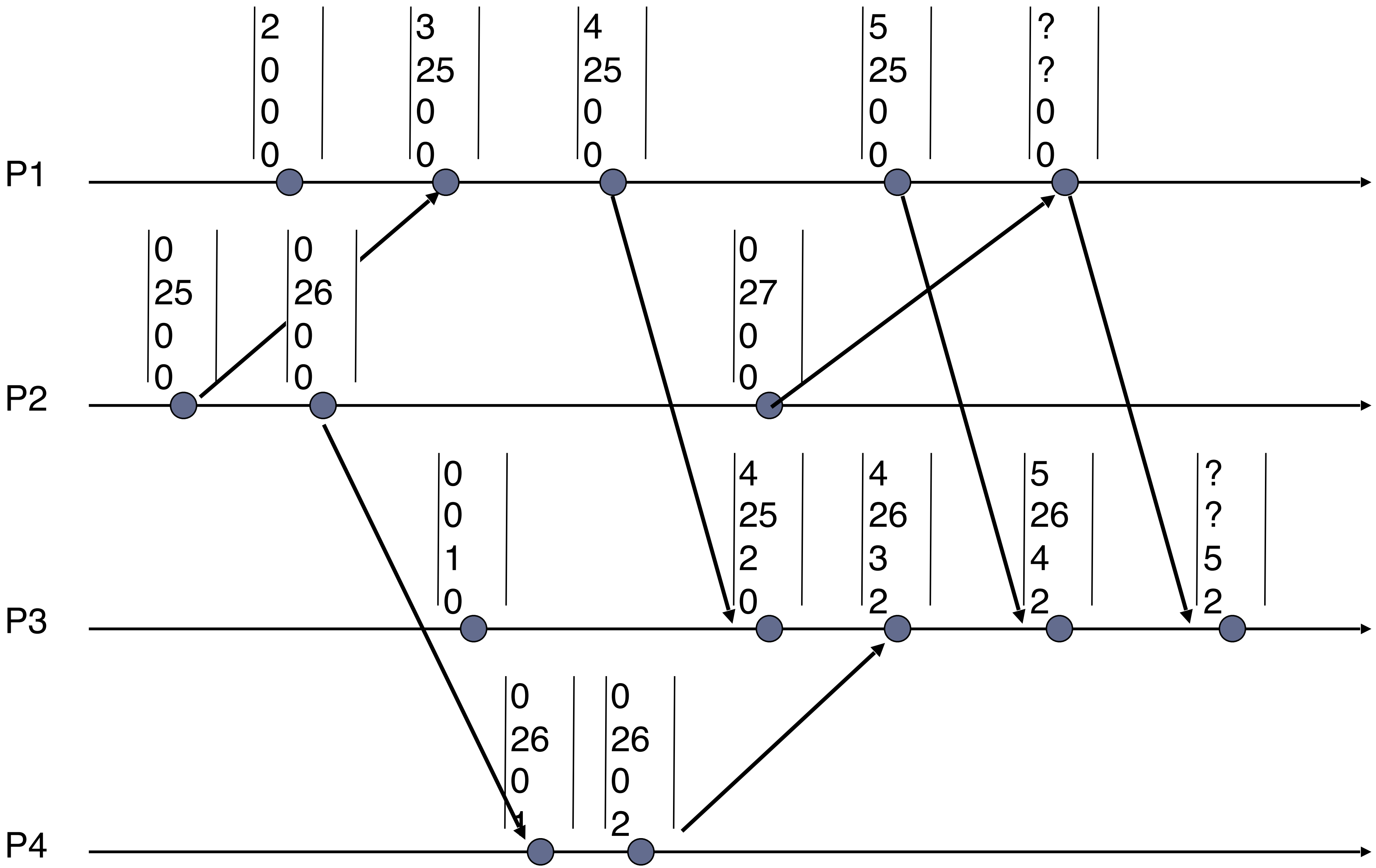
Vector Time Ticks

- Initial:
  C_i:= (0, ...,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


P1
P2
P3
P4
2 0 0 0
3 25 0 0
4 25 0 0
5 25 0 0
0 25 0 0
0 26 0 0
0 27 0 0
0 0 1 0
4 25 2 0
4 26 3 2
5 26 4 2
0 26 0 1
0 26 0 2

Example



P1
P2
P3
P4
2 0 0 0
3 25 0 0
4 25 0 0
5 25 0 0
? ? 0 0
0 25 0 0
0 26 0 0
0 27 0 0
0 0 1 0
4 25 2 0
4 26 3 2
5 26 4 2
? ? 5 2
0 26 0 1
0 26 0 2

Properties of Vector Time

- Definition
  - $C_a \leq C_b :\Leftrightarrow \forall k: C_a[k] \leq C_b[k]$
  - $C_a < C_b :\Leftrightarrow C_a \leq C_b \wedge C_a \neq C_b$
  - $C_a \parallel C_b :\Leftrightarrow C_a \nless C_b \wedge C_b \nless C_a$
- Property
  - $C_a < C_b \Leftrightarrow a \rightarrow b$

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 as purely theoretical construct

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)

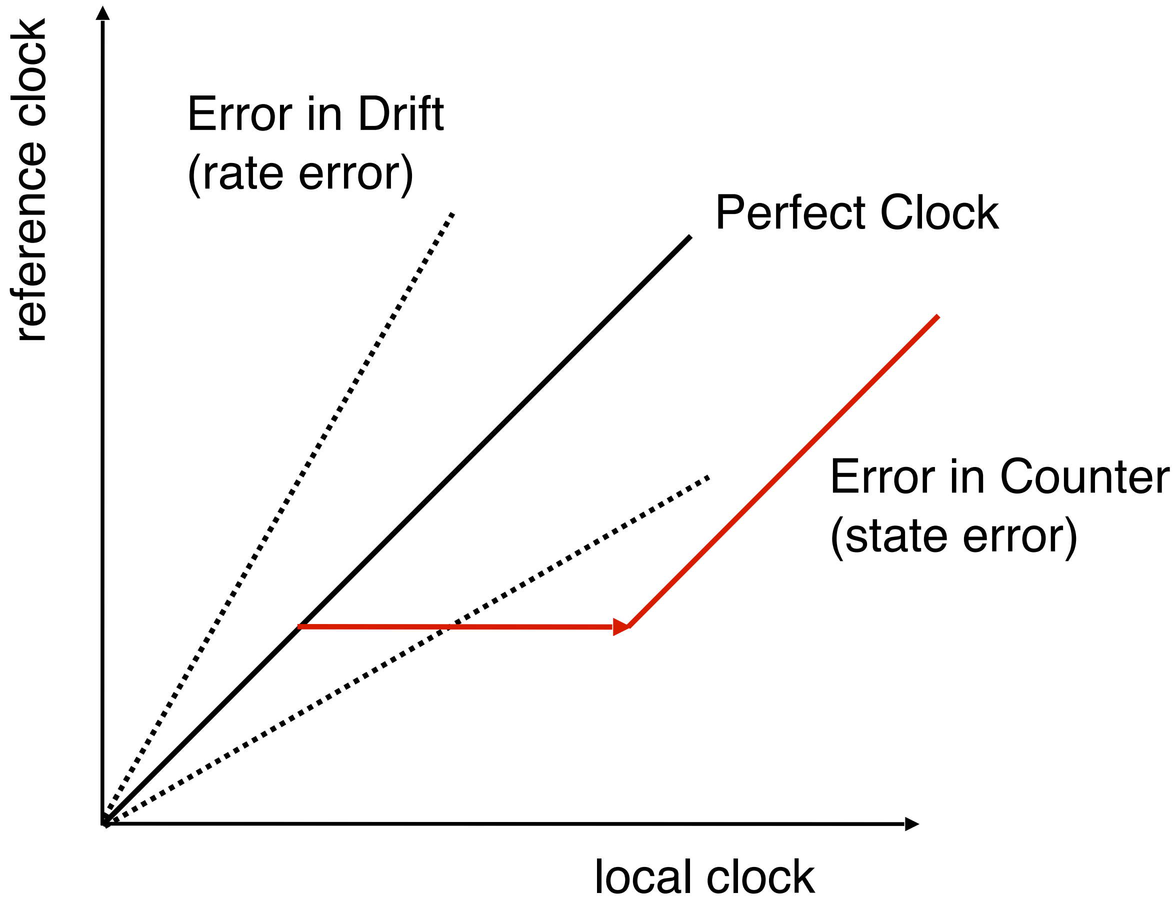
- Reference Clock z
  - perfect with regard to REAL TIME
  - dense (no ticks, to avoid digitalisation error)

- 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 Ticks of clock k
• Granularity	distance between adjacent Ticks

Failure Modes: Drift and Counter Errors



Maximum Drift Rate

Drift-Rate

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

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

Precision of an Ensemble of Clocks

Offset

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

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

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

Precision

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

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

- maximum offset for any two clocks

Accuracy

Accuracy

- of a given clock in the period of interest:

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

- maximum offset to reference clock

- If all clocks of an ensemble have accuracy A, the precision of the ensemble is ??

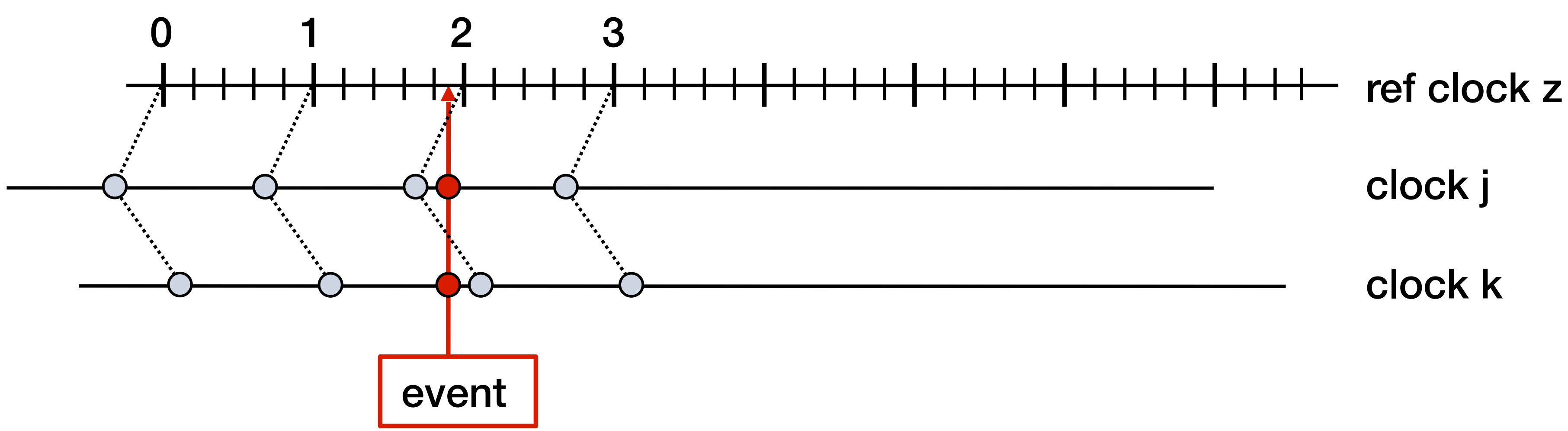
Resynchronisation

- External Resynchronization
- resynchronization with reference clock
- to maintain bounded accuracy

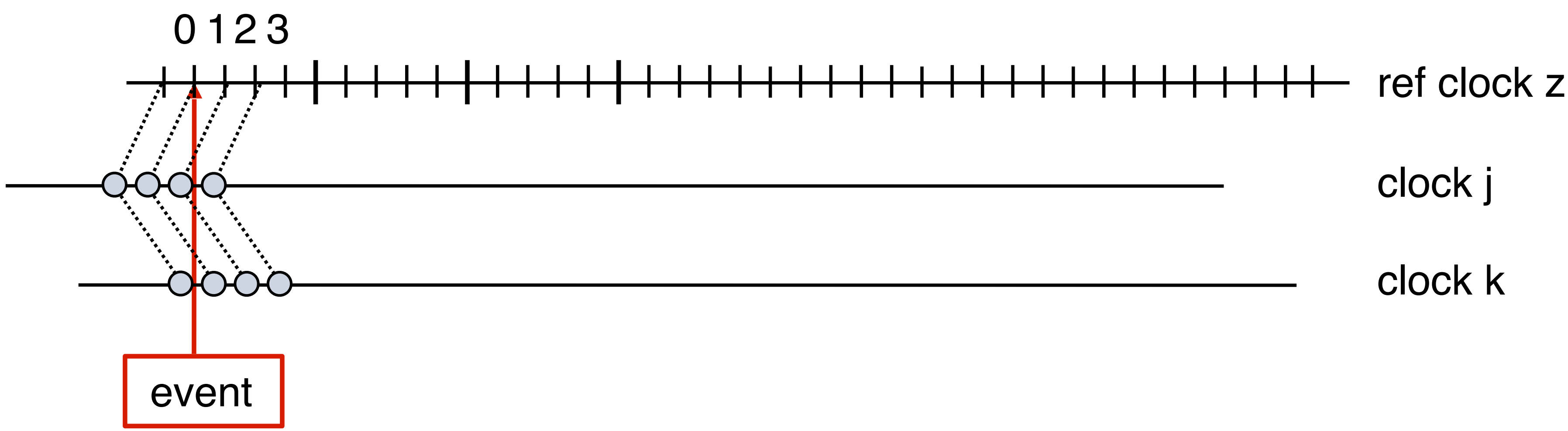
- Internal Resynchronization
- mutual resynchronization of an ensemble
- to maintain bounded precision

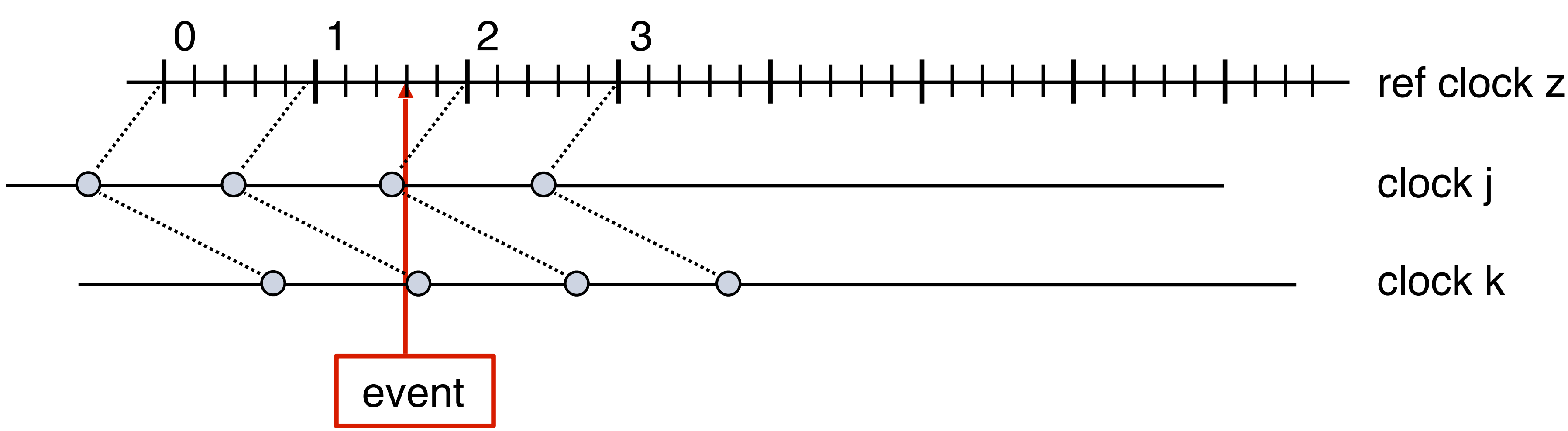
Global Time

- 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

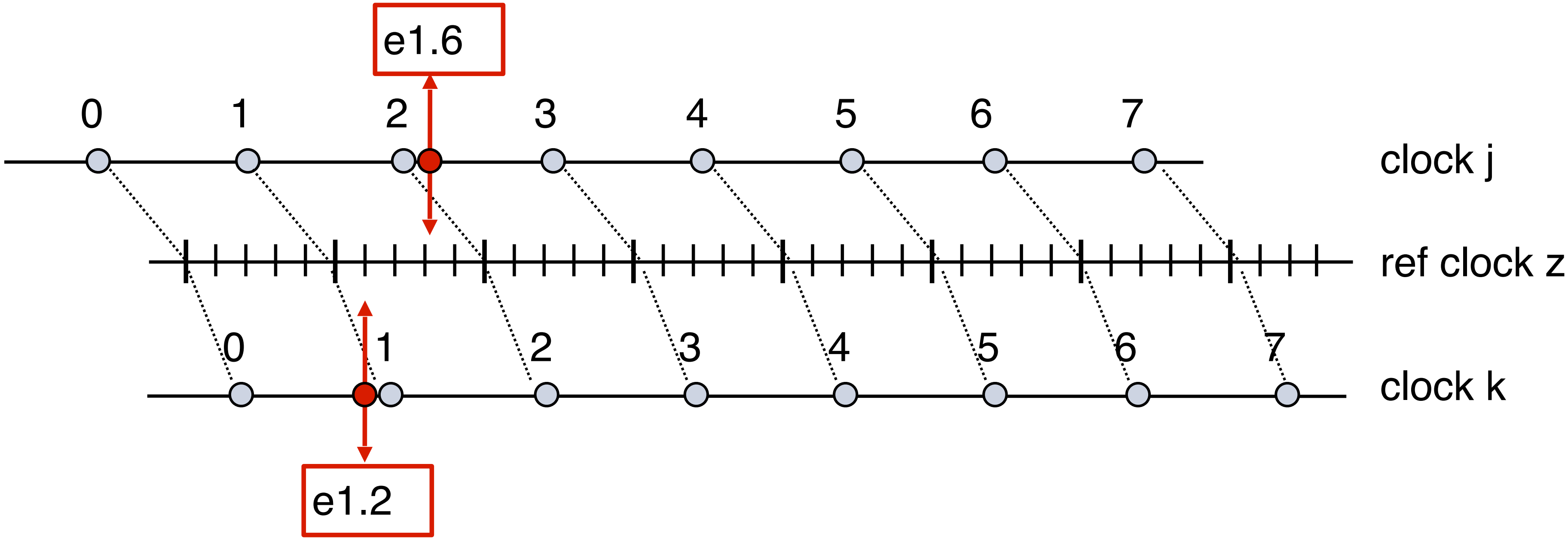




Reasonable: One Tick Difference

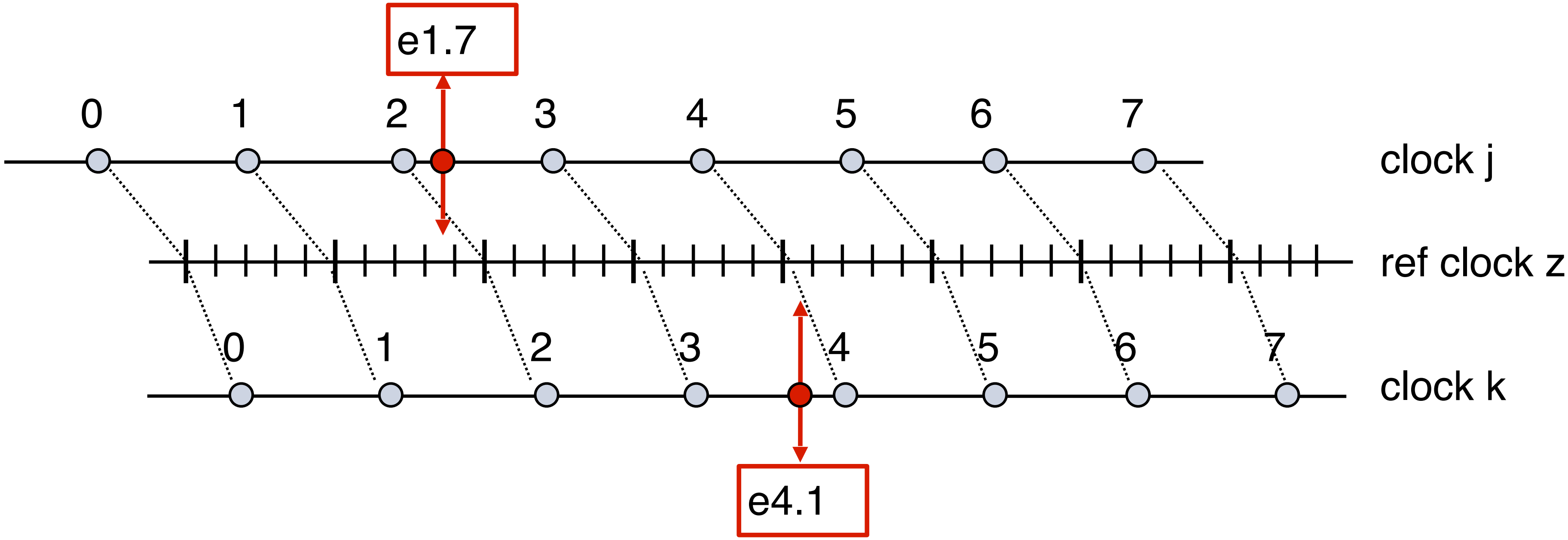
- Reasonableness Condition:
  - global time t is reasonable if
    $g^{global} > \pi$ holds for all local implementations
- t^j(event):
  - Denotes global time for the implementation at clock j
- Then: For any single event e, holds: $|t^j(e) - t^k(e)| \leq 1$
- Global timestamps differ at most by one (macro-)tick.
  Best one can achieve!

Interpretation for Temporal Order



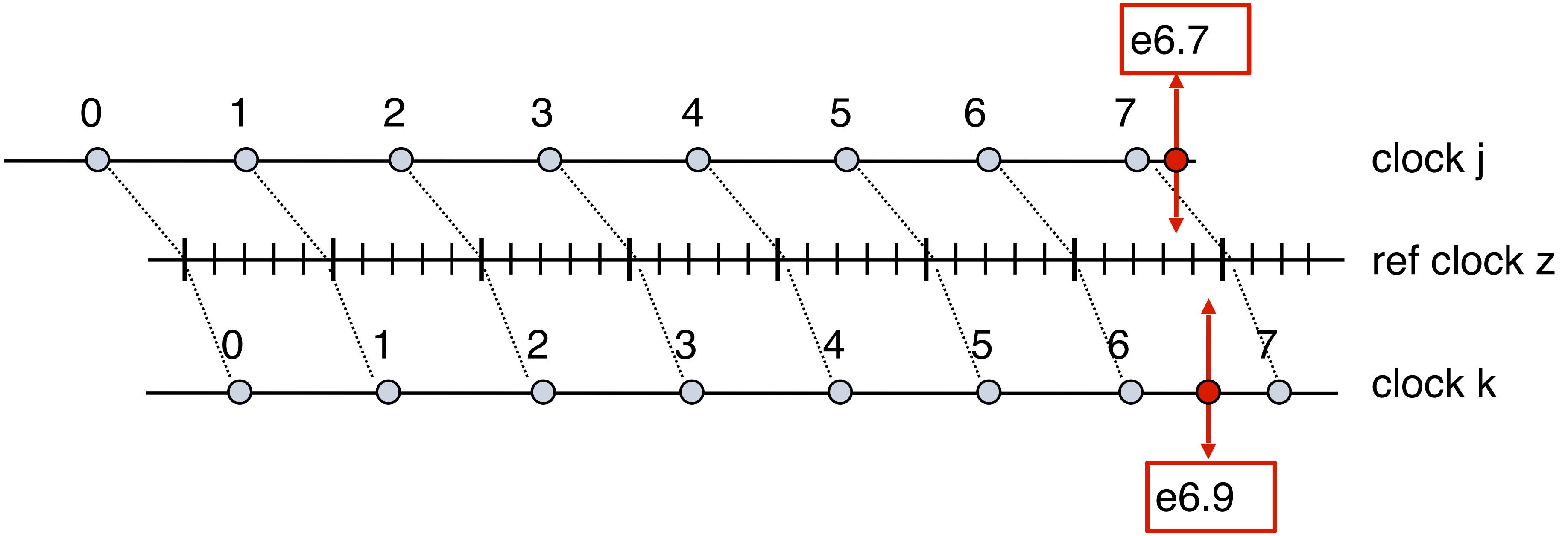
- z(e1.6) – z(e1.2): 0.4 g^{global} ref clock
- t^j(e1.6) - t^k(e1.2): 2 global time ticks
- Temporal order can be established because $Tick^k_1$ must be before $Tick^j_2$ (Reasonabless Condition)
- Hence: If the (global) timestamps differ by two ticks, the temporal order can be established.

Caution: Example



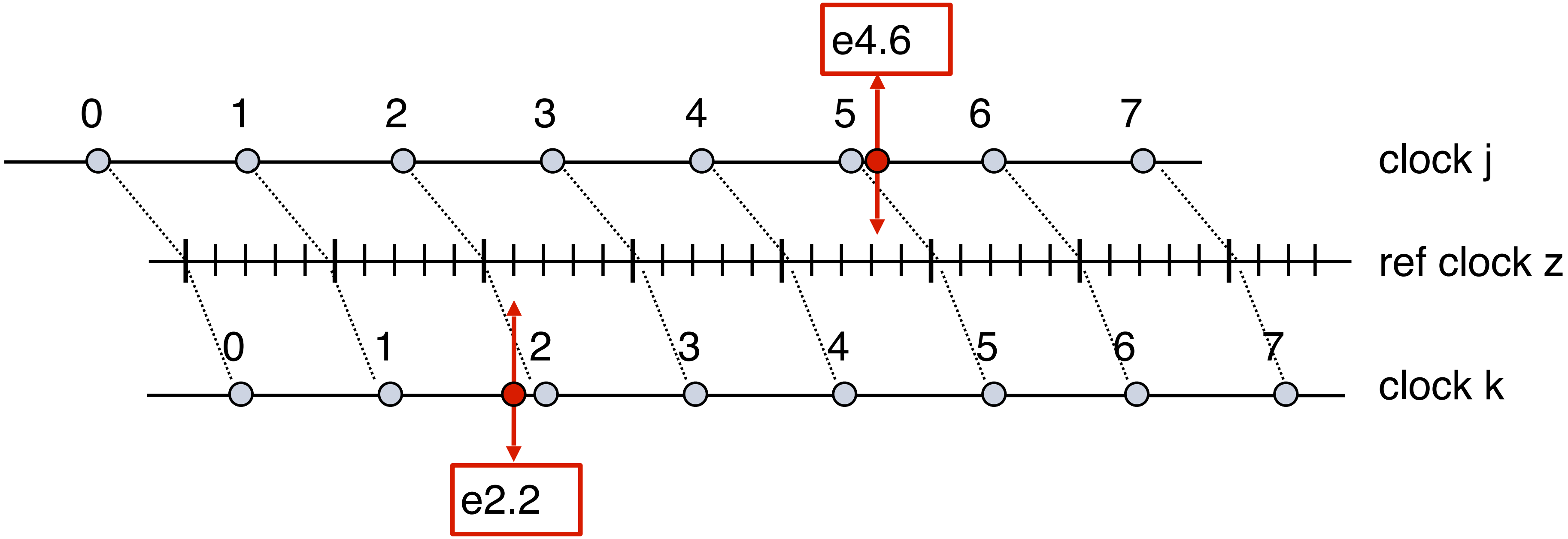
- z(e4.1) – z(e1.7): 2.4 g^{global} ref clock
- t^k(e4.1) - t^j(e1.7): 1 global tick
- A distance of 2*g^{global} between two events does not suffice to determine temporal order.

Another Example



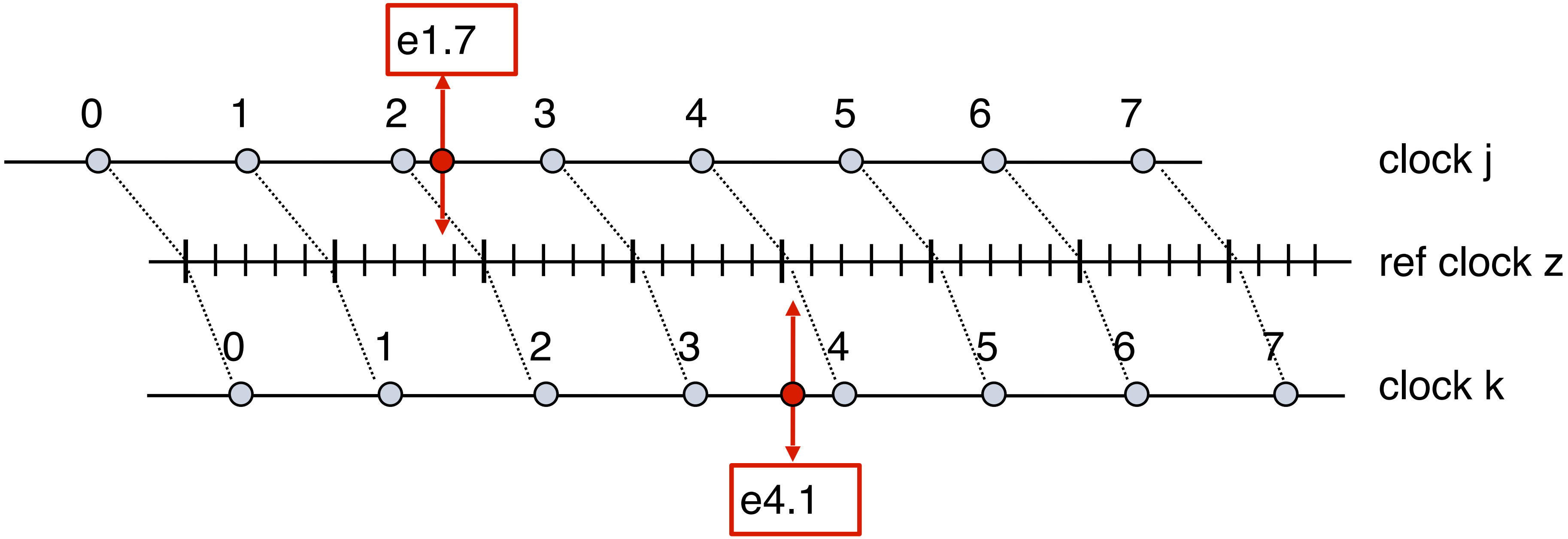
- $z(e6.9) > z(e6.7)$
- $t^k(e6.9) < t^j(e6.7)$

Interpretation for Durations



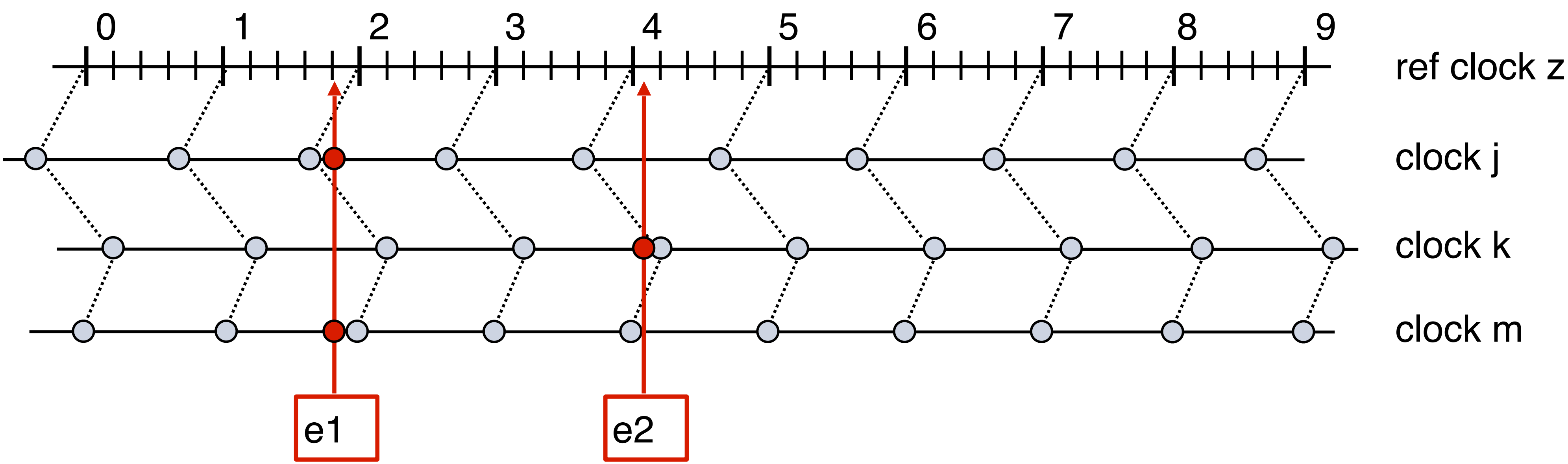
- True duration: 2.4
- Observed duration d: 5-1=4 ($t^j(e4.6) - t^k(e2.2)$)
- Can be driven to true duration: 2+ε for small ε
- $d^{obs} - 2^* g^{global} \leq d^z$

Interpretation for Durations



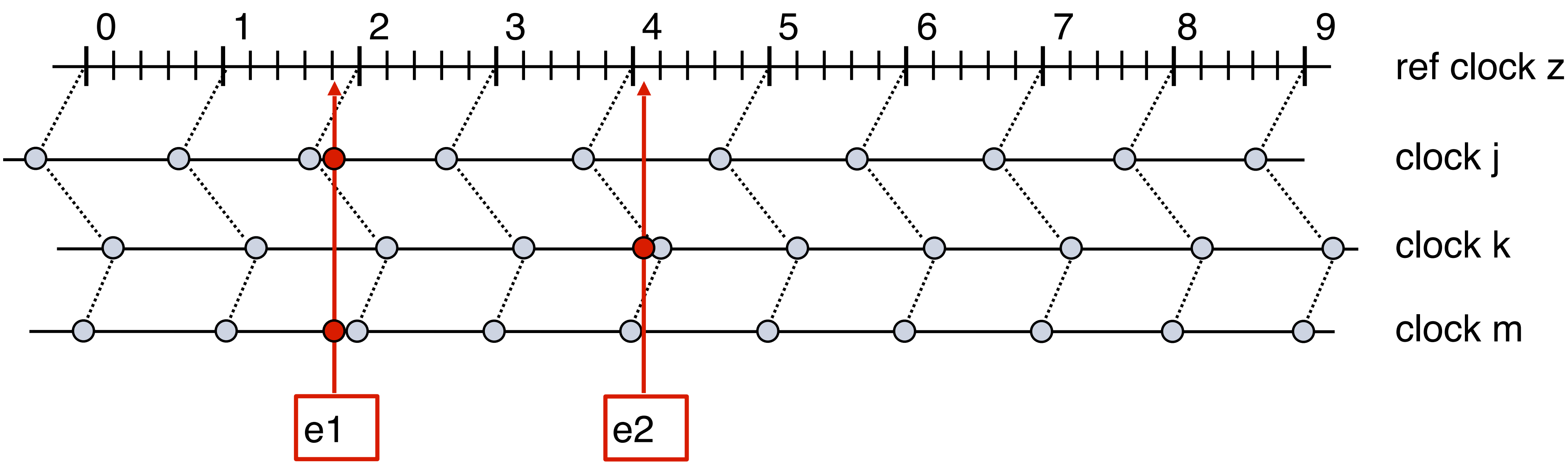
- True duration: 2.4
- Observed duration d: 3-2=1 ($t^k(e4.1) - t^j(e1.7)$)
- Example can be constructed for true duration: 3 - ε for small ε (and still observed duration is 1)
- $d^{obs} - 2*g^{global} < d^z < d^{obs} + 2*g^{global}$

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

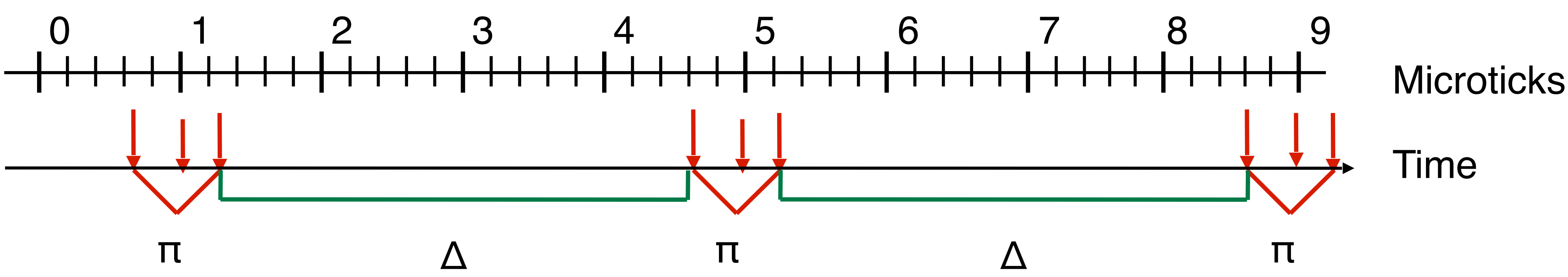
Agreement Protocols

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

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

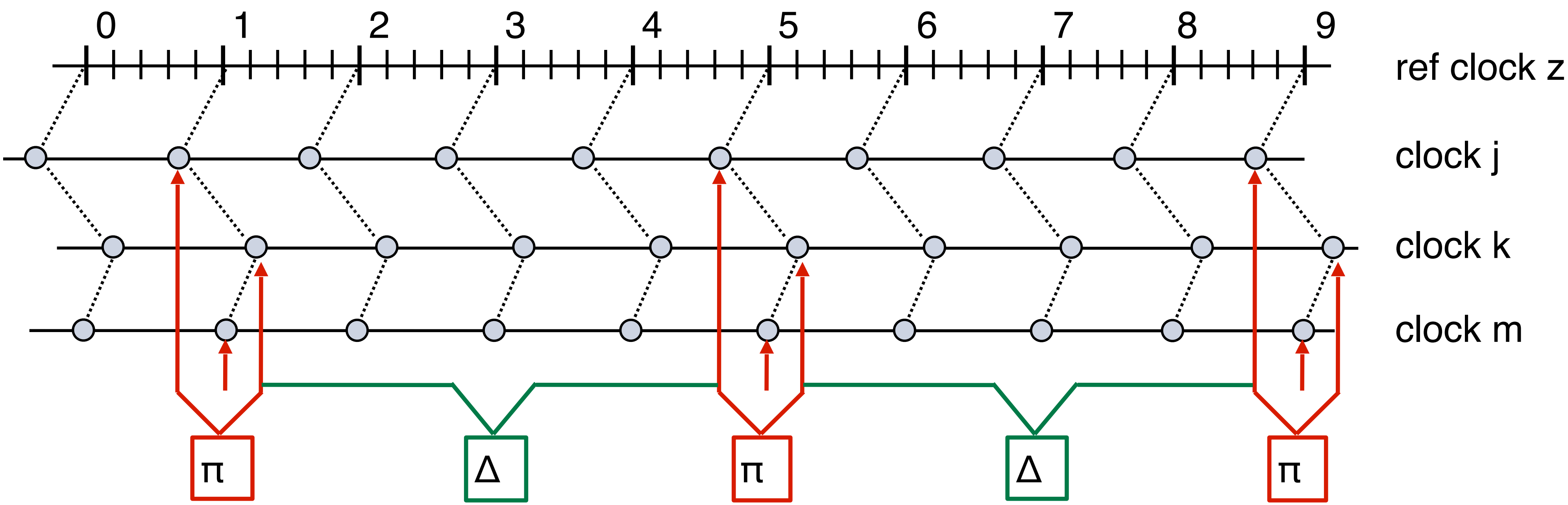
Dense Time vs. Sparse Time



- 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

Generated Events

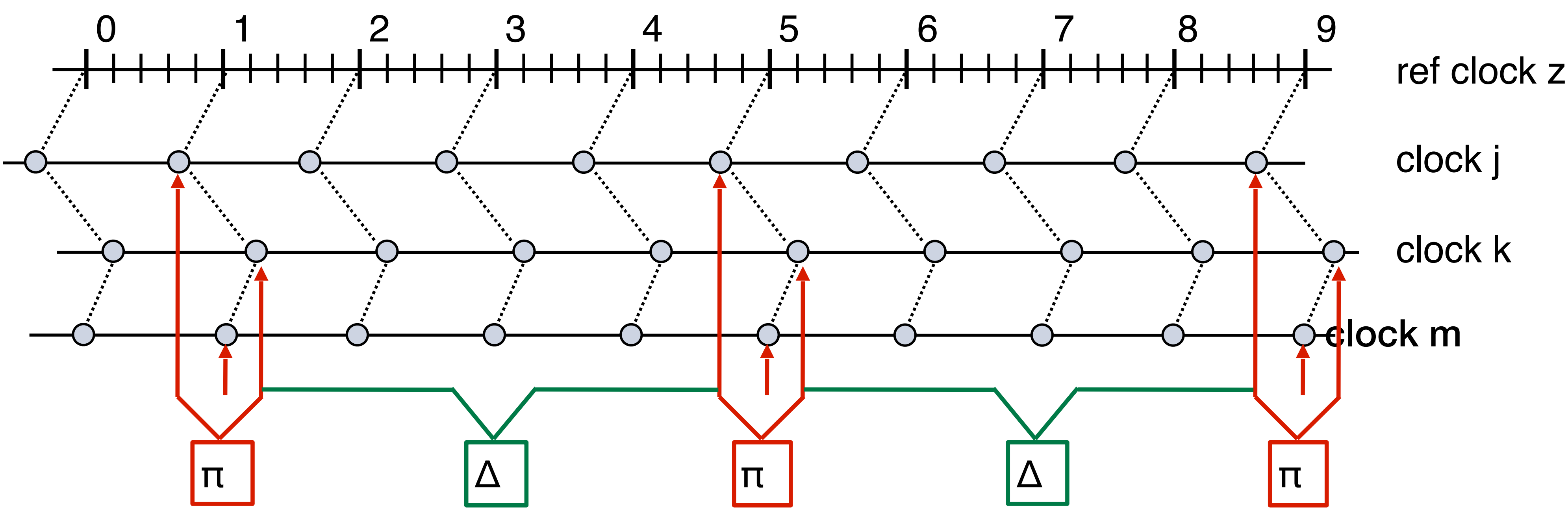
- Cluster of three nodes:
  - each generates event at the same global tick
  - t = 1, 5, 9
- observation:



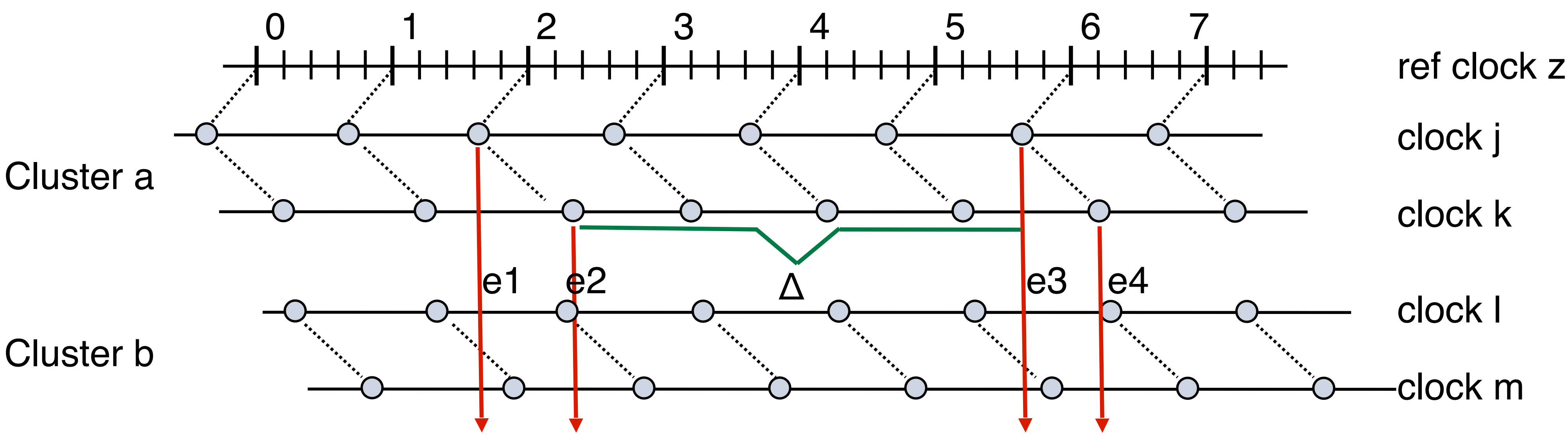
π/Δ-Precedence of Sets of Events

- 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:

$$[|z(e_i)-z(e_j)|\leqslant\pi]\vee[|z(e_i)-z(e_j)|>\Delta]$$



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

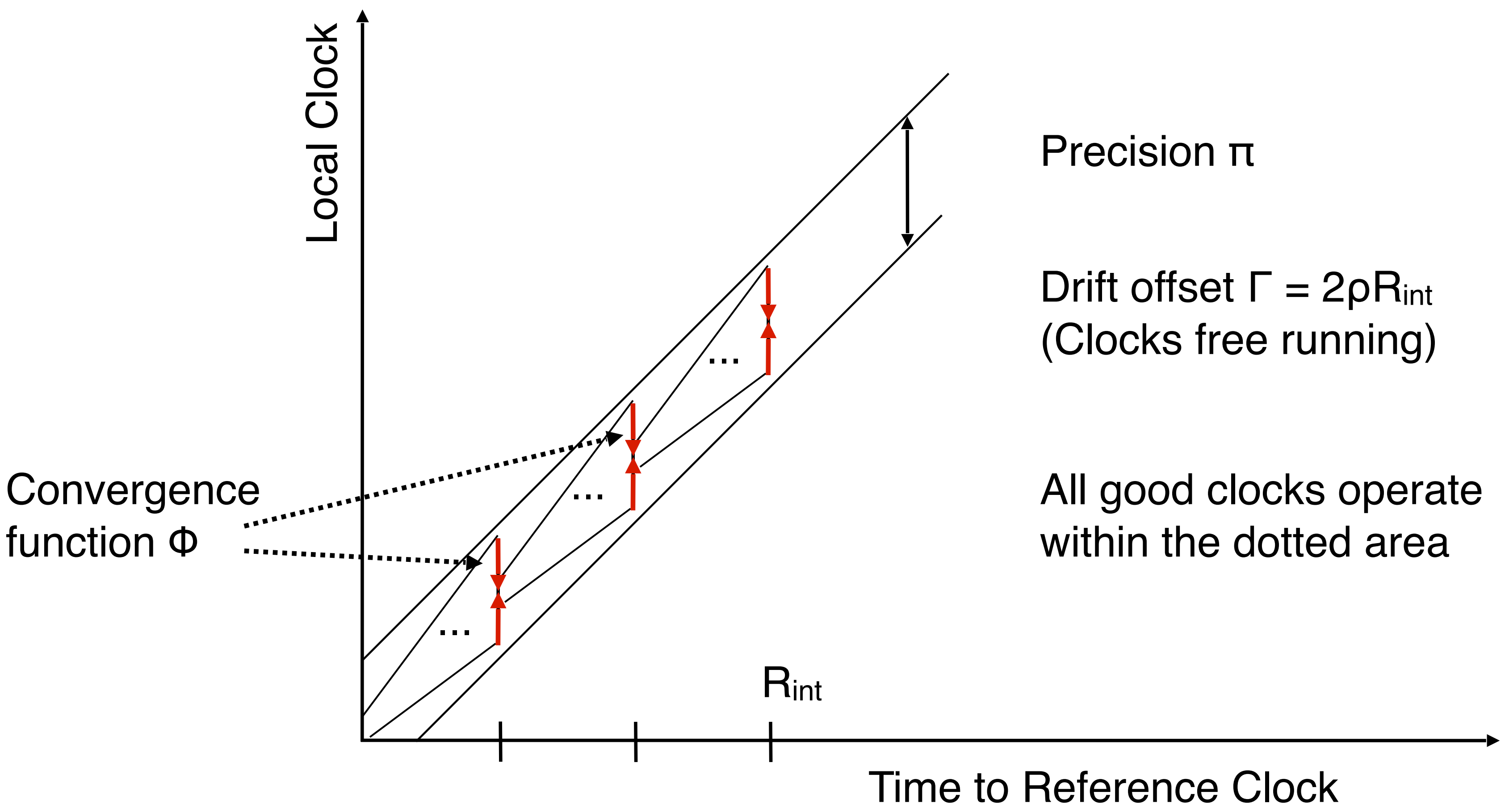
Temporal Order

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 precendent	$\lvert t^j(e_1) - t^k(e_2) \rvert \geqslant 0$	no
0/2g precendent	$\lvert t^j(e_1) - t^k(e_2) \rvert \geqslant 1$	no
0/3g precendent	$\lvert t^j(e_1) - t^k(e_2) \rvert \geqslant 2$	yes
0/4g precendent	$\lvert t^j(e_1) - t^k(e_2) \rvert \geqslant 3$	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

Internal Clock Synchronisation



Precision π

Drift offset $\Gamma = 2\rho R_{int}$
(Clocks free running)

All good clocks operate within the dotted area

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

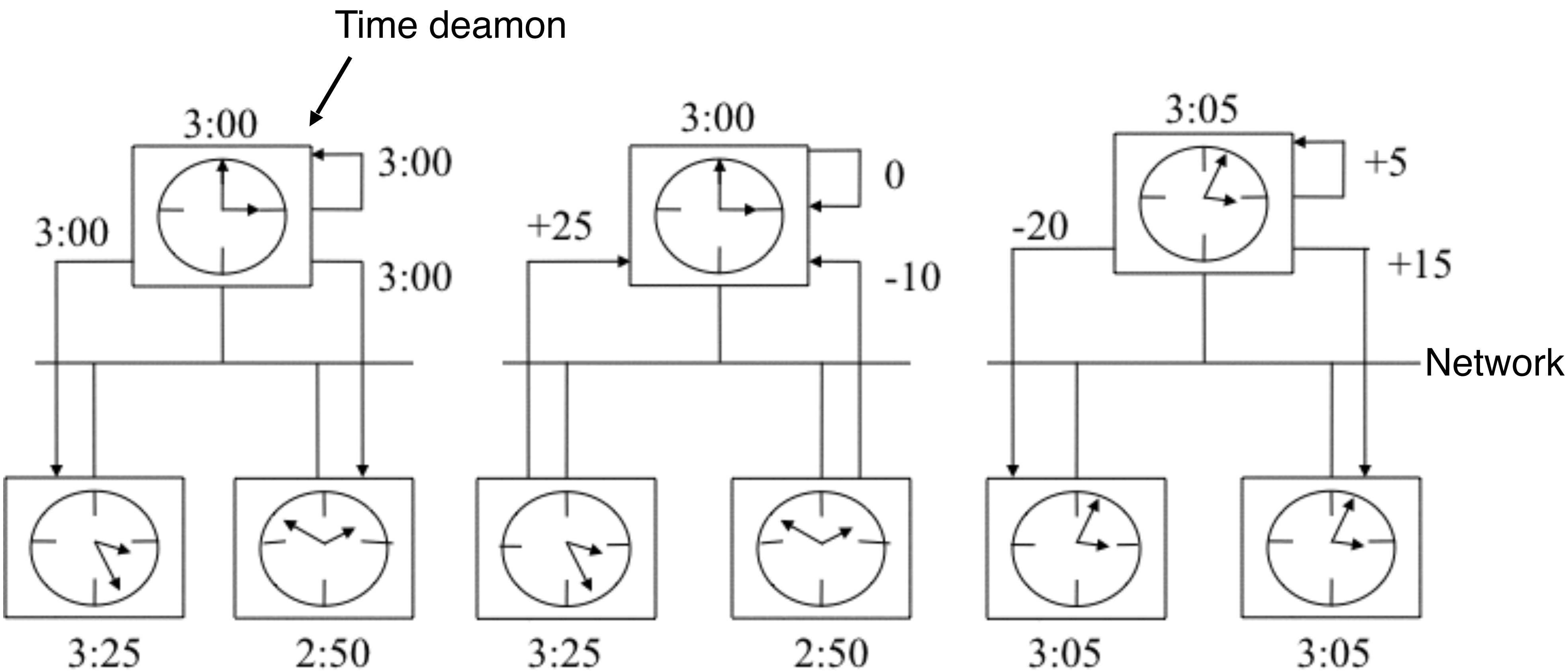
Synchronisation Condition

- resynchronization interval: R_{int}
- convergence function : Φ offset after resynch.
- drift offset: Γ

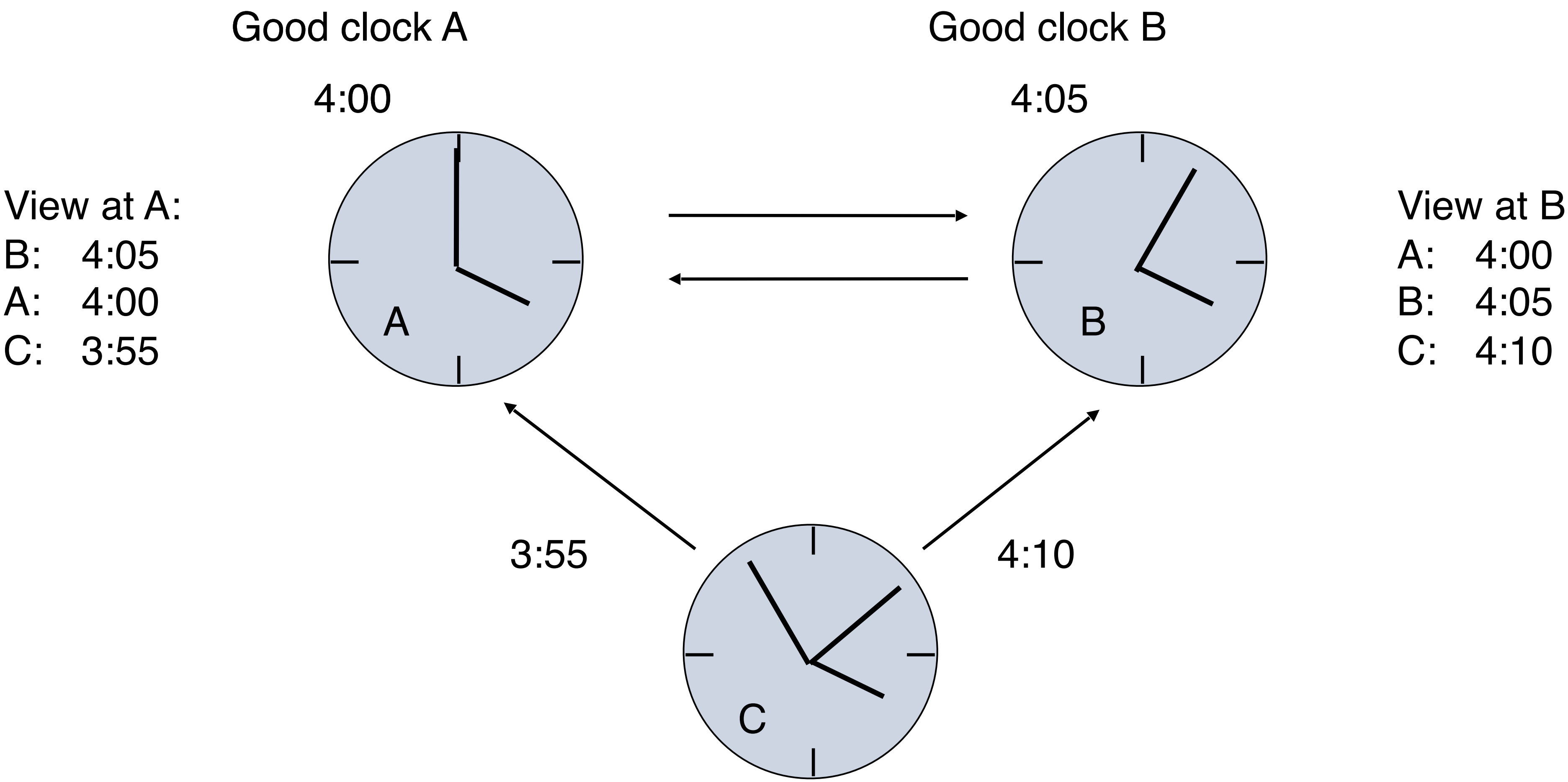
$$\Gamma = 2\rho R_{int}$$

- Required: $\Gamma + \Phi \leq \Pi$

Distributed Synchronisation: Berkeley



Byzantine Error



“two faced” malicious clock C

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).

Literature

- Logical Clocks:
  Standard text books: Coulouris, Tanenbaum

- Physical Clocks:
  This lecture followed strictly
  Hermann Kopetz, Distributed Real-Time-Systems

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