Time and Order

slide credits: H. Kopetz, P. Puschner

Why do we need a notion of time?

- Event identification and generation
  - State before vs. after the event
- Event ordering
  - Causal order (e.g., *a* may only have caused *b* if *a* happened before *b*)
  - Temporal order (e.g., flight booking: who was first, *A* in VIE or *B* in LA?)
- Coordination – coordinated action at specified time
- Duration – measurement / control

  (e.g., X-ray: exposure time, video: gap between frames)
- Modeling of physical time
  - Comply to laws/dynamics of physics (*second*, physical time, real time)
  - Read input, produce output “at the right time” (e.g., control loops)

What is tricky?

- Order vs. causality
- Determining order or simultaneity under different conditions:
  - Event numbering vs. timestamping
  - Central system vs. distributed system
- Timestamping
- Measuring durations

Causal and Temporal Order

Example

> Two events
>
> *e1* … someone enters a room
> *e2* … the telephone starts to ring
>
> Two cases
>
> *e1* occurs after *e2* → causal dependency possible
> *e2* occurs after *e1* → causal dependency unlikely

- Causal order implies temporal order
- Temporal order is necessary but not sufficient to establish causal order

Causal and Temporal Order

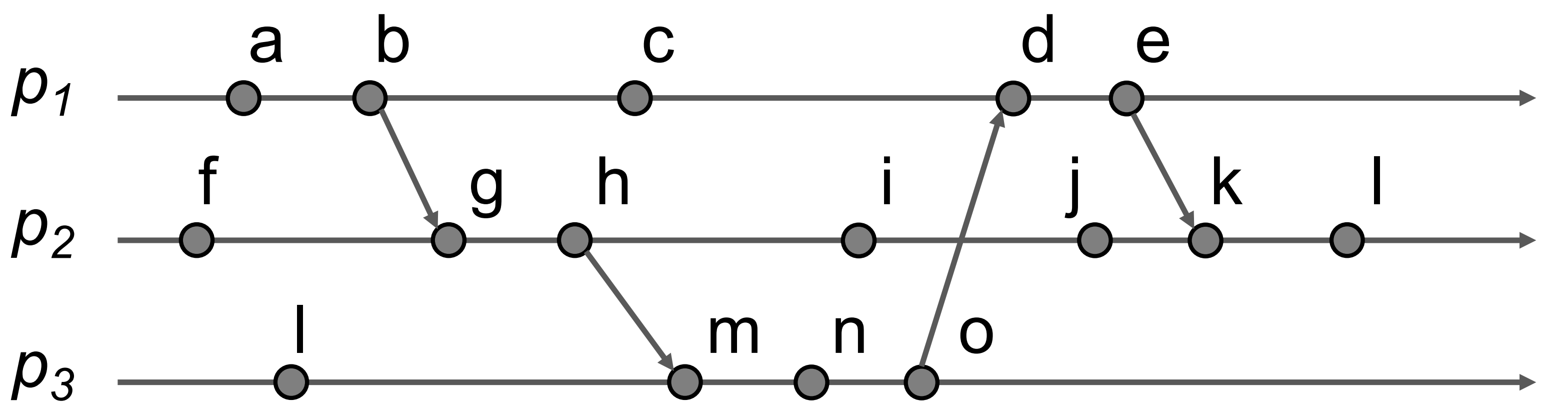
Causal Order

- Deduced from “causal dependency” between events
- Reichenbach: *“If event e1 is a cause of event e2, then a small variation (a mark) in e1 is associated with a small variation in e2, whereas small variations in e2 are not necessarily associated with small variations in e1.”*
- Bunge: *“If a Cause happens, then (and only then) the Event is always produced by it.”*

Temporal Order

- Deduced from timestamps of physical time

Causal Order of Computer-generated Events



Causal Order of Computer-generated Events

Partial order for computer-generated events

$a \rightarrow b$ … a causes b (happened before, causal dependence)

1. If a, b … events within a sequential process and
   a is executed before b
   then: $a \rightarrow b$
2. If a … send event of a message by process p_i and
   b … receive event of the message by process p_k
   then: $a \rightarrow b$
3. $\rightarrow$ is transitive

Logical Clocks

- Represent information about causal dependency
- Do not use physical time
- Events are “time”-stamped using monotonically increasing counters

Events	a, b with $a \rightarrow b$
Timestamps	$C(a)$, $C(b)$

- Desirable properties
  - $a \rightarrow b \Rightarrow C(a) < C(b)$ … monotonicity, consistency
  - $a \rightarrow b \Leftrightarrow C(a) < C(b)$ … strong consistency

Lamport‘s Logical Clocks

- Logical clocks of processes p_i represent the local views of global time
- Non-negative integer C_i represents the local clock of p_i
- Clock update rules:

  R1: p_i increments C_i for each local event (e.g., event, send):

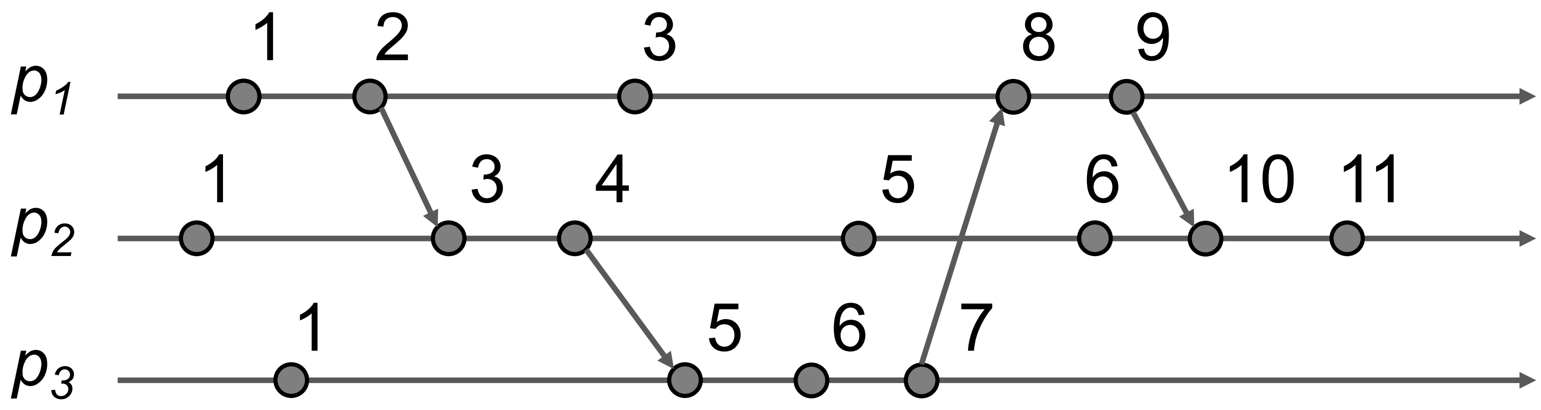
  $C_i = C_i + 1;$

  R2: each message transports the value of the sender’s clock, C_{msg}

  R3: when p_i receives a message with timestamp C_{msg}:

  $C_i = \max(C_i, C_{msg}); C_i = C_i + 1;$

Lamport‘s Logical Clocks (2)



- Consistency: $a \rightarrow b \;\Rightarrow\; C(a) < C(b)$
- Total ordering: timestamps (t, i): t … time, i … process number
  total order relation $\prec$ on events a, b with timestamps (t, i), (u, j)

  $$a \prec b \;\Leftrightarrow\; (t < u \text{ or } (t = u \text{ and } i < j))$$
- No strong consistency: $C(a) < C(b) \not\Rightarrow\; a \rightarrow b$

Vector Time (Fidge, Mattern, Schmuck)

- n-dimensional vector $V_i\,[1..n]$ at p_i with

  $V_i\,[\,i\,]$ … value of local logical clock of p_i

  $V_i\,[\,k\,]$ … p_i's knowledge about local time at p_k

$$V_i \begin{bmatrix} 1 \\ \boldsymbol{i} \\ n \end{bmatrix}$$

- Clock update rules:

  R1: p_i updates $V_i\,[\,i\,]$ for each local event:

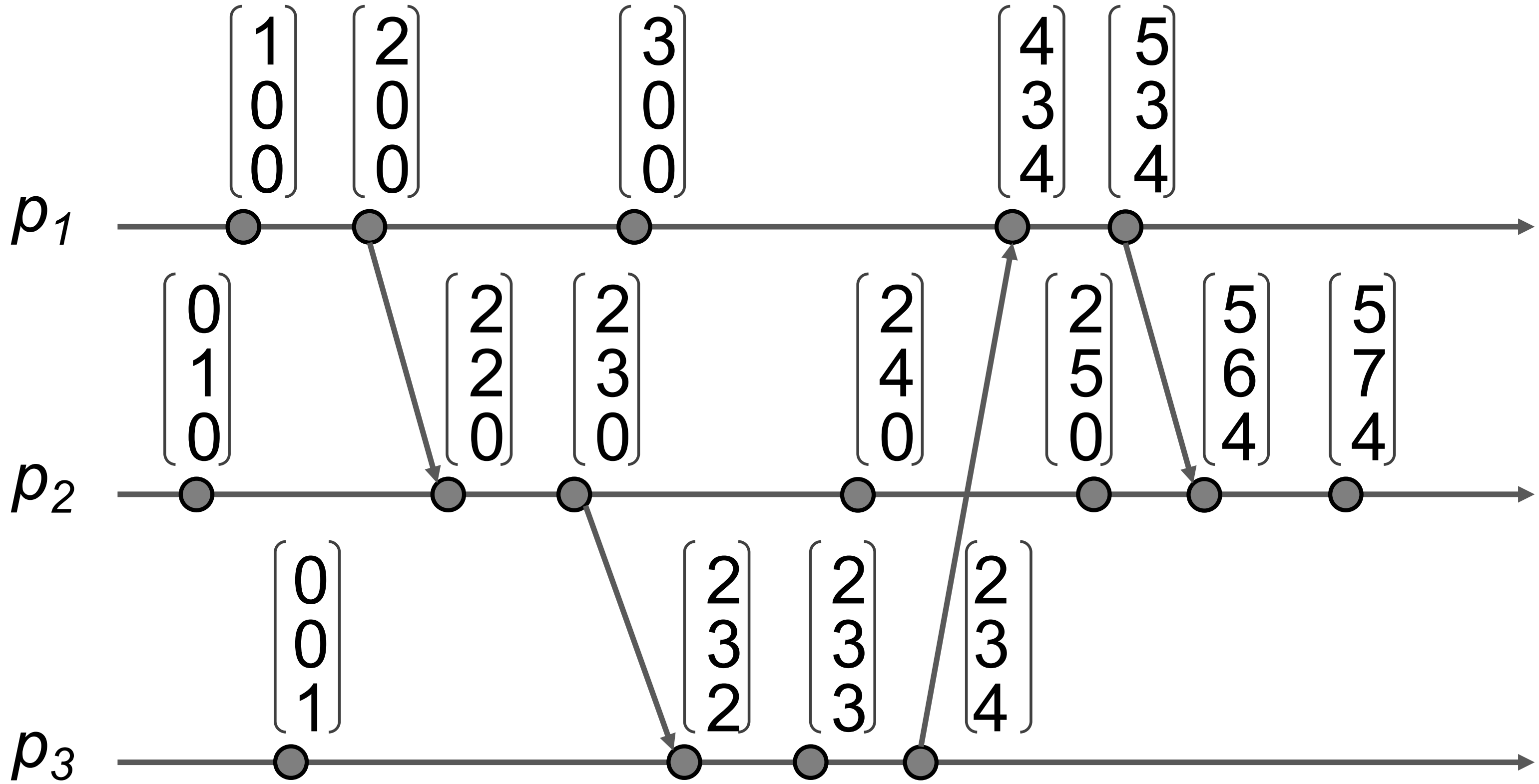
  $$V_i\,[\,i\,] = V_i\,[\,i\,] + 1;$$

  R2: each message transports sender's clock values

  R3: when p_i receives a message with timestamp V_{msg}:

  $$1 \le k \le n:\ V_i\,[\,k\,] = \max\,(V_i\,[\,k\,], V_{msg}\,[\,k\,]);$$
  $$V_i\,[\,i\,] = V_i\,[\,i\,] + 1;$$

Vector Time (2)



Vector Time (3)

Event relations

event a on p_i with timestamp Va
event b on p_k with timestamp Vb

- $a \rightarrow b \;\Leftrightarrow\; \forall i\colon Va[\,i\,] \leq Vb[\,i\,]$ and $\exists i\colon Va[\,i\,] < Vb[\,i\,]$
- $a \,||\, b \;\Leftrightarrow\; \exists i,k\colon \mathrm{Va}[\,i\,] > Vb[\,i\,]$ and $Va[\,k\,] < Vb[\,k\,]$

- Vector clocks are strongly consistent:
  By examining the timestamps of two events *a* and *b* one can determine if *a* and *b* are causally related

Temporal Order

Continuum of real time modeled by

- a directed timeline, consisting of
- an infinite set $\{T\}$ of instants with
  i. $\{T\}$ is an ordered set,
  i.e., for any two instants p and q either: p and q are simultaneous, p precedes q, or q precedes p
  ii. $\{T\}$ is a dense set,
  for any instants $p \neq r$ there is at least one q between p and r

Temporal order: total order of instants on the timeline

Events and Durations

Event … is happening at an instant of time

Duration … section of the timeline

Note

- An event does not have a duration
- If two events occur at the identical instant they are called simultaneous
- Events are partially ordered
  In a distributed system, a total order can be established by using process numbers (see Lamport's order)

Physical Clocks

Clock

- Counter plus oscillator
- Microticks are generated by periodical increments of the counter, following some law of physics
- Reference clock (z)

  Perfect clock of an external observer

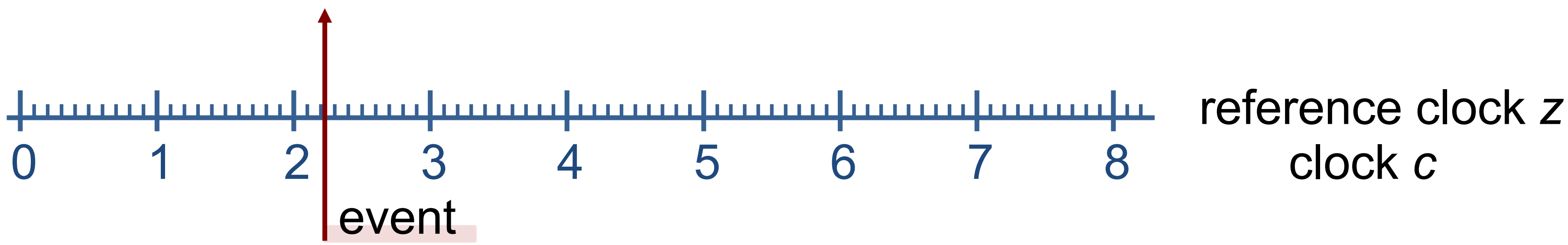
  Duration between two ticks is much smaller than duration of any interval to be observed with our clocks (e.g., 10^{-15} sec)
- Granularity of a clock c: nominal number of microticks of z between any consecutive microticks of c

$$g^c = z(microtick^c_{i+1}) - z(microtick^c_i)$$

Physical Clocks (2)

Timestamp

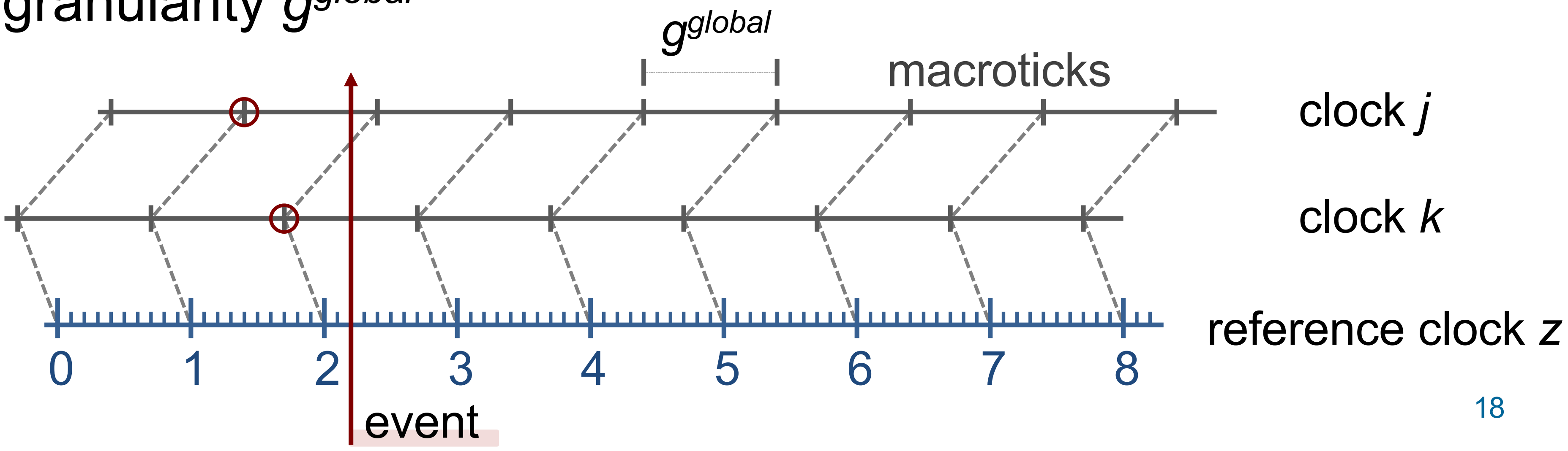
- The timestamp of an event is the state of the clock immediately after the occurrence of the event
- Notation: *clock*(*event*), e.g., *z*(*event*)
- Digitalization error of timestamps due to clock granularity



Global Time

In a distributed system we need a global notion of time to generate event timestamps ➪ “Global Time”

- Global time is an abstract notion, real clocks are not perfect
- Local clocks of nodes approximate global time
- Macroticks form the local representation of global time with granularity g^{global}



Precision

Offset between two clocks *j* and *k* at tick *i*

$$offset^{jk}_{i} = \left| z(microtick^{j}_{i}) - z(microtick^{k}_{i}) \right|$$

Precision of an ensemble of clocks {1,…,*n*} at microtick *i*

$$\Pi_{i} = \max_{j,\,k} \{ offset^{jk}_{i} \}$$

Accuracy

Offset between clock *k* and the reference clock *z* at tick *i*

$$offset^{k,z(k)}_i = \left| z(microtick^k_i) - z(microtick^{z(k)}_i) \right|$$

Accuracy denotes the maximum offset of a given clock from the reference clock during a time interval of interest

If all clocks of an ensemble are within accuracy *A*, then the ensemble has a precision $\Pi \leq 2A$.

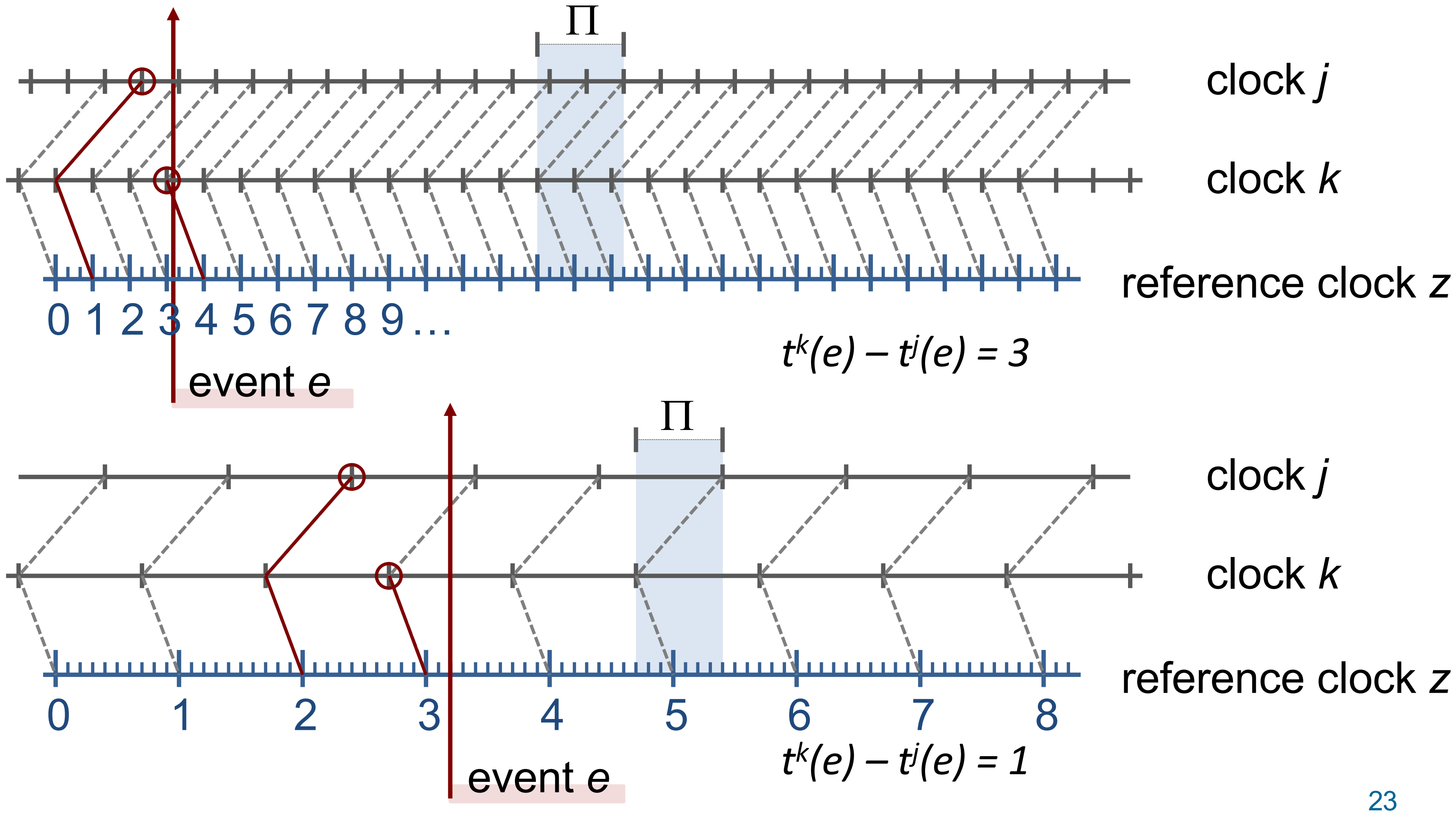
Absence of a Global Timebase

- *n* independent local time references
  ➪ only timestamps from the same clock can be related.
- Interval measurements between events observed at different nodes are limited by the end-to-end communication jitter.
- Delay jitter of communication system determines the jitter in non-local control loops
  ➪ unacceptable for many real-time control applications.
- No knowledge of precise point in time of measurement of process variables
  ➪ state estimation is very difficult

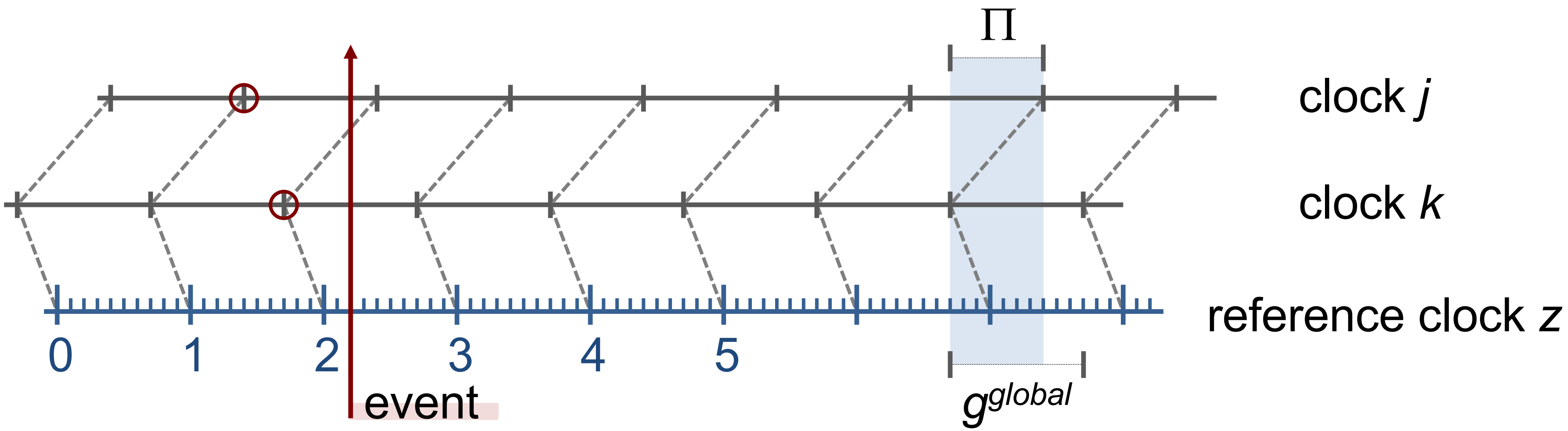
Requirements for a Global Timebase

- Chronoscopic behaviour
  (i.e., no discontinuities, even at points of resynchronization)
- Known precision Π
- High dependability
- Metric of physical second

Choosing the Right Granularity



Reasonableness Condition



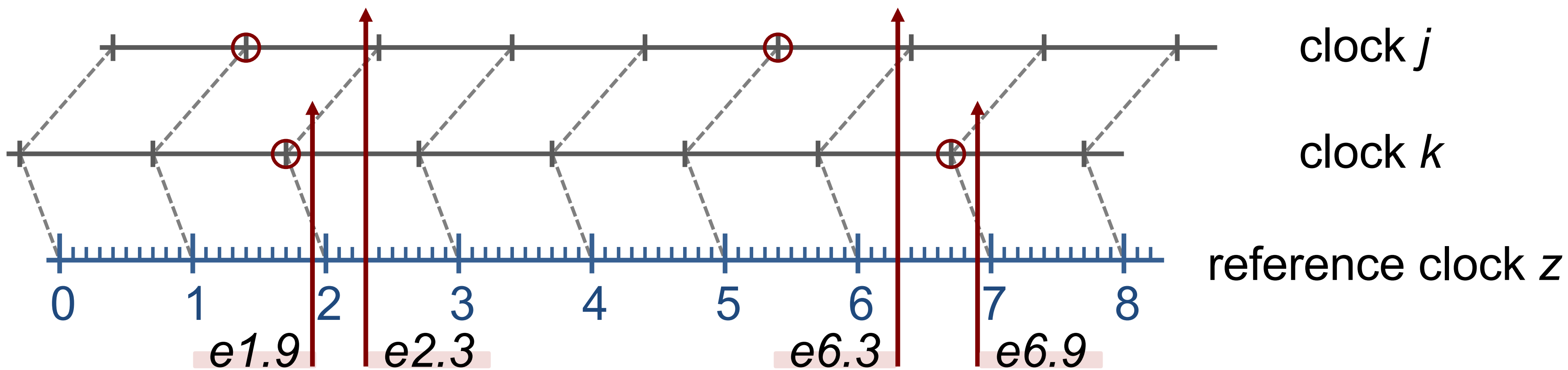
Global time t is reasonable if for all local implementations:

$$g^{global} > \Pi$$

The reasonableness condition ensures that:

- the synchronization error is less than one macrogranule
- for any event e: $| t^j(e) - t^k(e) | \leq 1$

Reconstructing Temp. Order from Timestamps



$z(e1.9) < z(e2.3)$

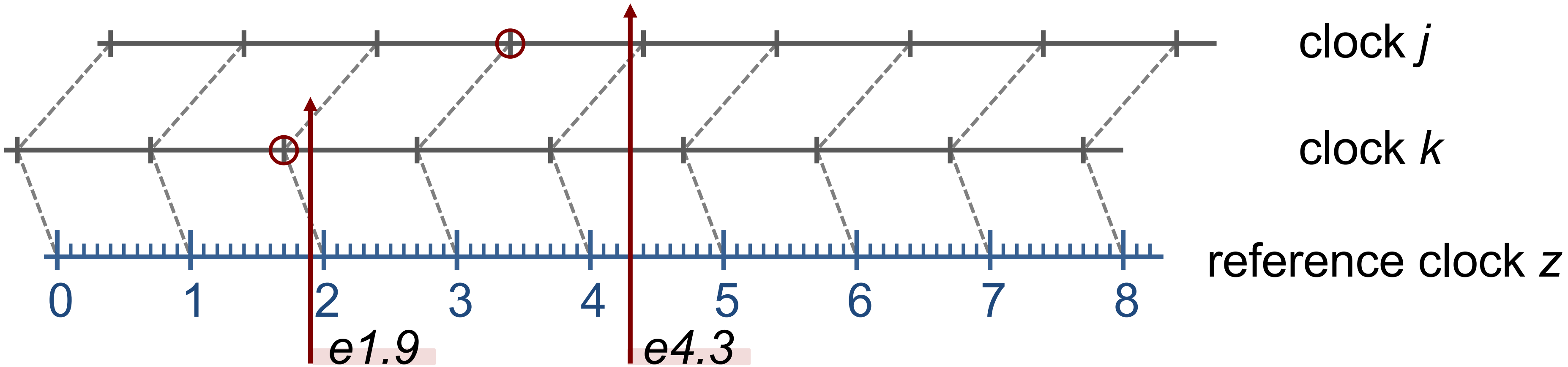
$t^k(e1.9) > t^j(e2.3)$

$z(e6.9) - z(e6.3) = 0.6$

$t^k(e6.9) - t^j(e6.3) = 2$

To reconstruct the temporal order of two events, the (global) timestamps of the events have to differ by at least two ticks.

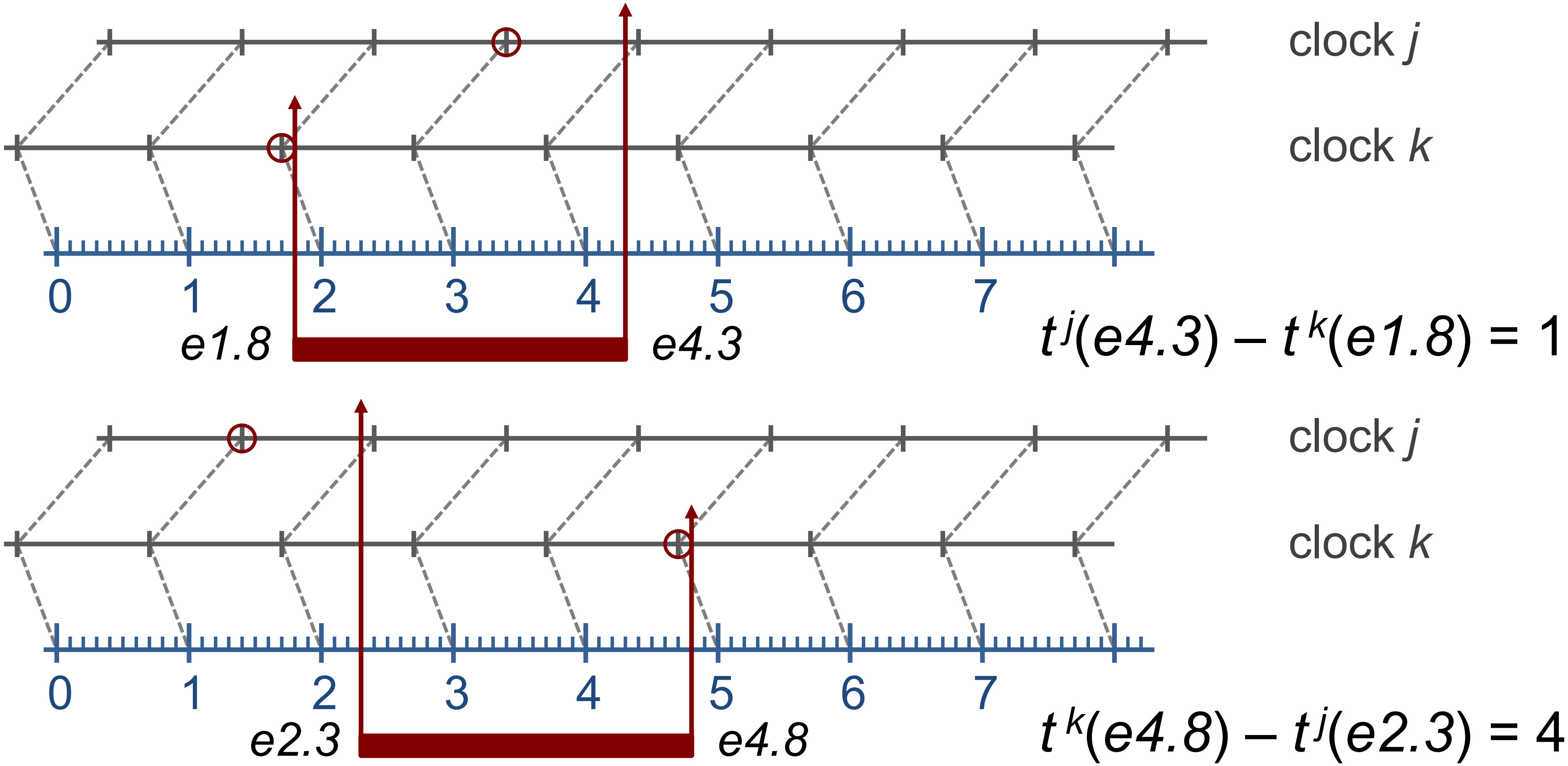
Reconstructing Temp. Order from Timestamps (2)



$z(e4.3) - z(e1.9) = 2.4$
$t^j(e4.3) - t^k(e1.9) = 1$

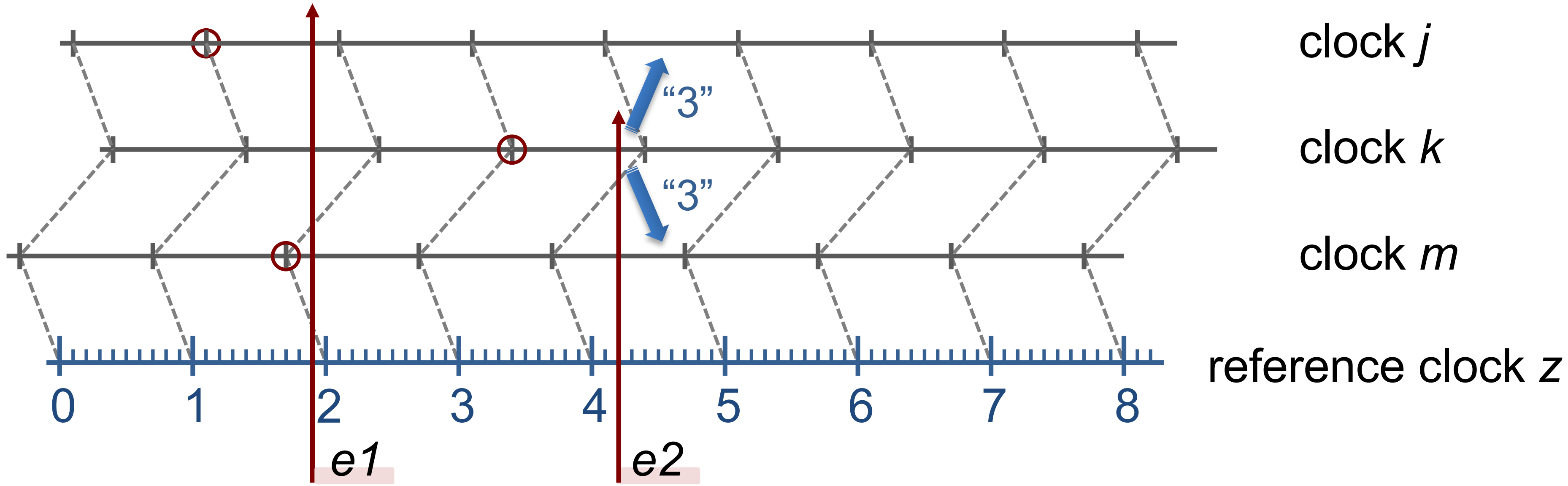
A time distance of $2g^{global}$ between two events is not sufficient to determine their temporal order (if $t^j(a) - t^k(b) = 1$) .

Measurement of Durations



Real duration: $d_{obs} - 2g^{global} < d^z_{true} < d_{obs} + 2g^{global}$

Agreement on Event Order – Dense Time



Nodes *j* and *m* observe *e1*, node *k* observes *e2.*

Node *k* reports observation about *e2* to nodes *j* and *m*.

➪ Nodes *j* and *m* draw different conclusions about event order.

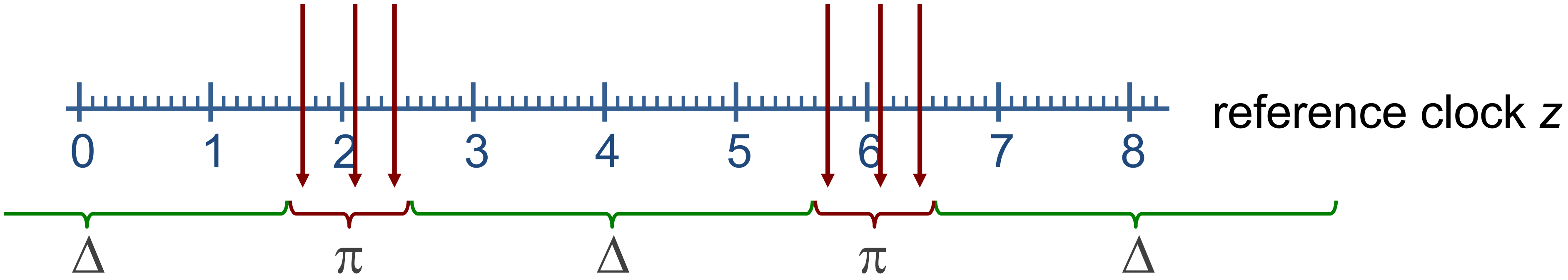
$$t^k(e2) - t^j(e1) = 2, \quad t^k(e2) - t^m(e1) = 1$$

Agreement on Event Order – Dense Time

Conclusions from observations:

- If a single event is observed by two nodes, the local timestamps for the event may differ by one tick.
  - ➪ an explicit agreement protocol (communication between the nodes) is needed to establish a consistent view about the global time of the event occurrence.
- If two events occur on a dense timeline, then it is impossible to consistently deduce the temporal order in all cases if the events occur within an interval of duration $< 3g^{global}$.
  - ➪ explicit agreement is needed for arbitrary event sets.
  - ➪ alternative: $0/\Delta$-precedent event set with $\Delta \geq 3g^{global}$.

π/Δ-Precedence of Sets of Events



Given durations π and Δ ($\pi << \Delta$), a set of events $E=\{e_i\}$ is π/Δ-precedent, if the following condition holds for all $e_j, e_k \in E$:

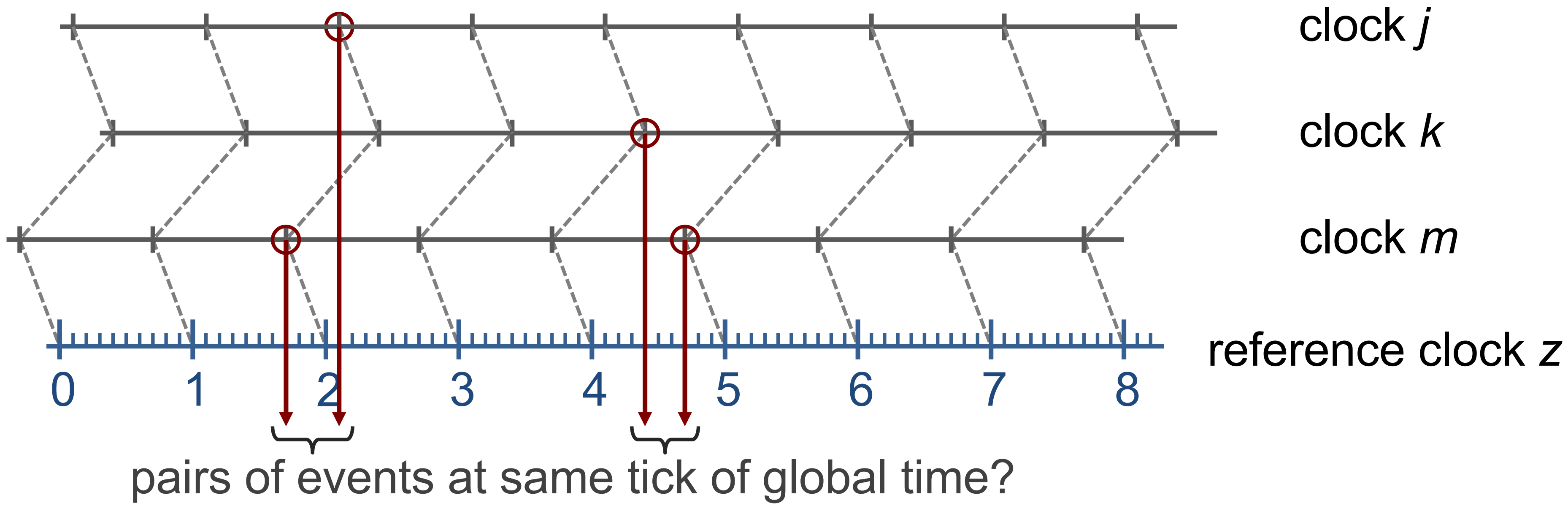
$$\left(|\, z(e_j) - z(e_k) \,| \leq \pi\right) \text{ or } \left(|\, z(e_j) - z(e_k) \,| > \Delta\right)$$

4 Fundamentals about Timestamping & Events

Given a distributed system with a reasonable global timebase, with granularity g^{global}:

- If a single event is observed by two nodes, the local timestamps for the event may differ by one tick.
- Duration measurement: $d_{obs} - 2g^{global} < d^{z}_{true} < d_{obs} + 2g^{global}$.
- The temporal order of two events e_1, e_2 can be deduced from their timestamps if $| t^j(e_1) - t^k(e_2) | \geq 2$.
- The temporal order of events can always be deduced if the event set is 0/Δ-precedent with $\Delta \geq 3g^{global}$.

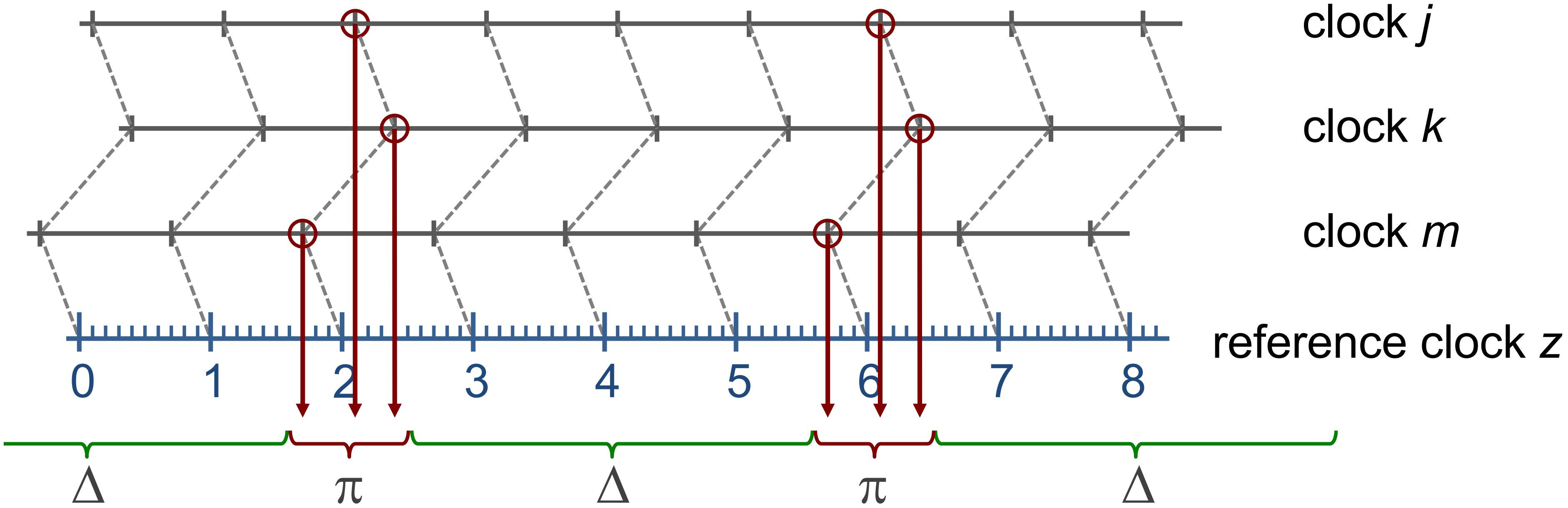
Temporal Relationship between Generated Events



Assumption: nodes generate events at clock ticks

An external observer cannot reconstruct whether local timestamps of generated events are equal or not

Dense Time and Sparse Time



Dense timebase: events are allowed to occur at any time.

Sparse timebase (π/Δ-sparse timebase):

events are only allowed to occur within the time intervals of activity π, followed by an interval of silence Δ.

Agreement on Event Order – Sparse Time

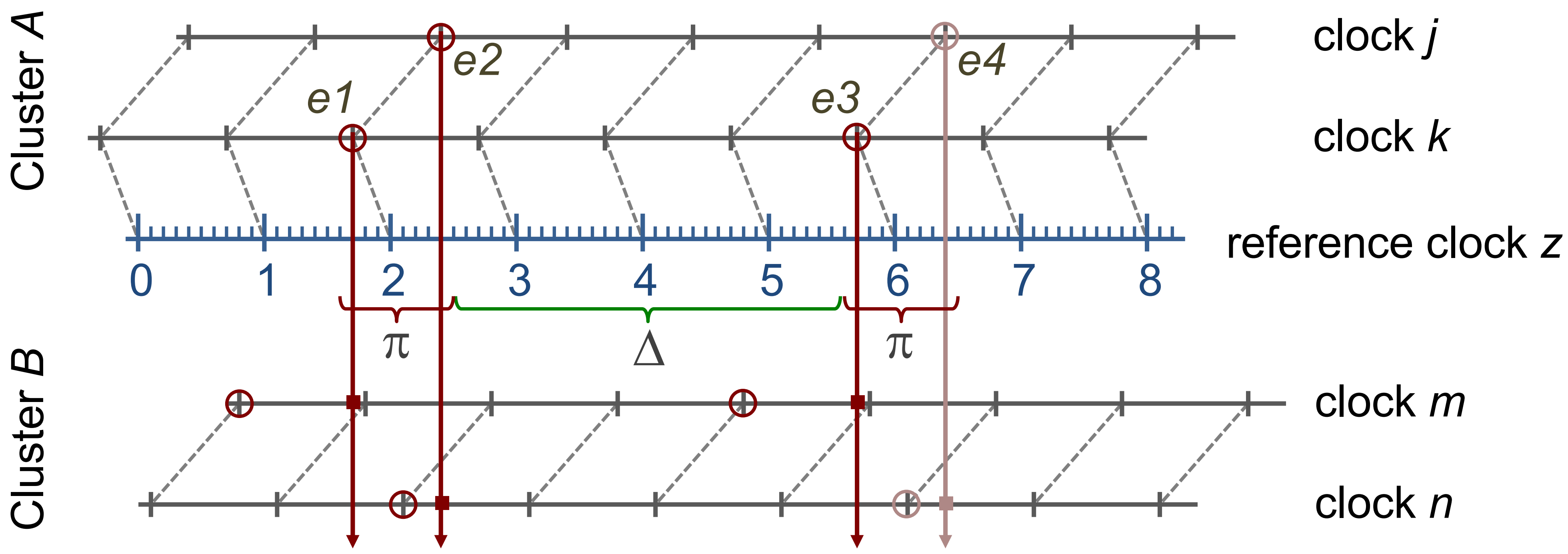
Assume: 2 computation clusters *A*, *B*

- within each cluster clocks are synchronized ($g = g^{global}$)
- no synchronization between *A* and *B*
- Cluster *A* generates events that have to be ordered by *B*:

  B must be able to determine order resp. simultaneity of all observed events

⇨ Timebase of *A* has to be 1*g*/*Ng*-sparse, with $N \geq 4$;
a 1*g*/3*g*-sparse timebase is not sufficient (see next slide)

Agreement on Event Order – Sparse Time (2)



e1, *e2* … generated in same activity interval: $t^n(e2) - t^m(e1) = 2$

e2, *e3* … gen. in different activity interval: $t^m(e3) - t^n(e2) = 2$

Lessons Learned

- Why we need time …
- Temporal and causal order
- Logical time (Lamport time, vector time)
- Physical time, event, duration
- Clocks and virtual global time
- Time stamps and temporal relations
- Sparse time