

Time and Order

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)

Causal and Temporal Order

Causal Order

- Deduced from “causal dependency” between events
- Reichenbach: *“If event e_1 is a cause of event e_2 , then a small variation (a mark) in e_1 is associated with a small variation in e_2 , whereas small variations in e_2 are not necessarily associated with small variations in e_1 .”*
- 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 and Temporal Order (2)

Example

Two events

$e1$... someone enters a room

$e2$... the telephone starts to ring

Two cases

$e1$ occurs after $e2$ \rightarrow causal dependency possible

$e2$ occurs after $e1$ \rightarrow causal dependency unlikely

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

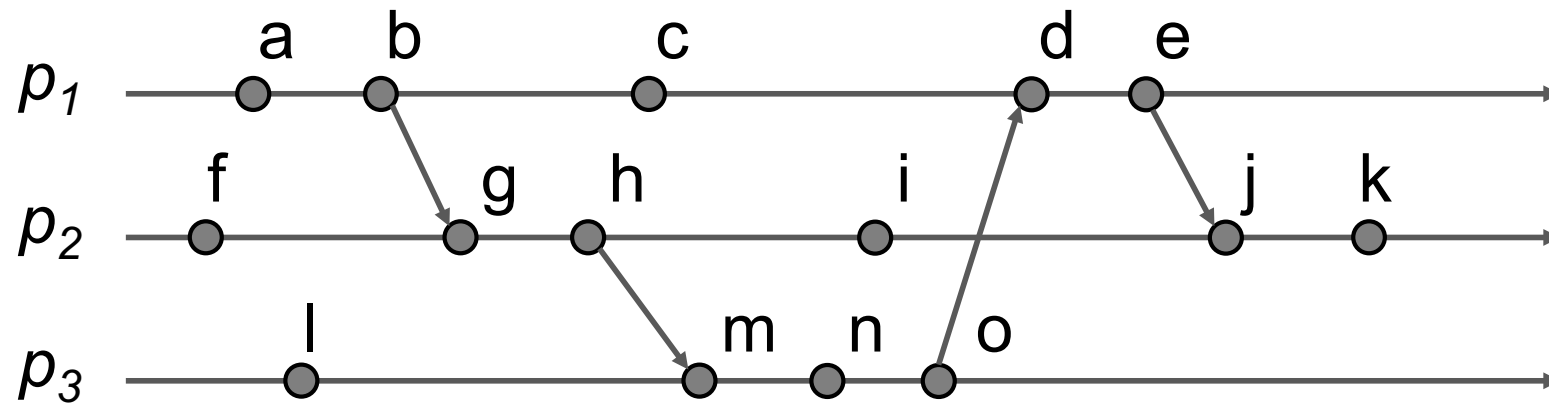
Causal Order of Computer-generated Events

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

Causal Order of Computer-generated Events (2)



Logical Clocks

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

Events a, b with $a \rightarrow b$

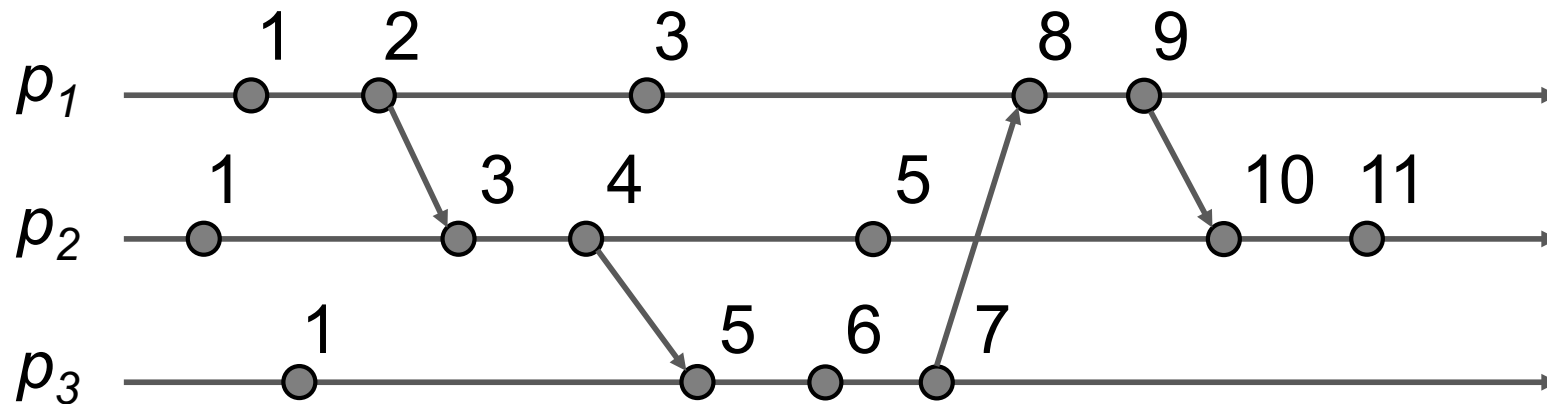
Timestamps $C(a), C(b)$

- 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) : $t \dots$ time, $i \dots$ process number
total order relation $<$ on events a, b with timestamps $(t, i), (u, j)$
 $a < b \Leftrightarrow (t < u \text{ or } (t = u \text{ and } i < j))$
- 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
- $$V_i \begin{pmatrix} 1 \\ i \\ n \end{pmatrix}$$
- Clock update rules:

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

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

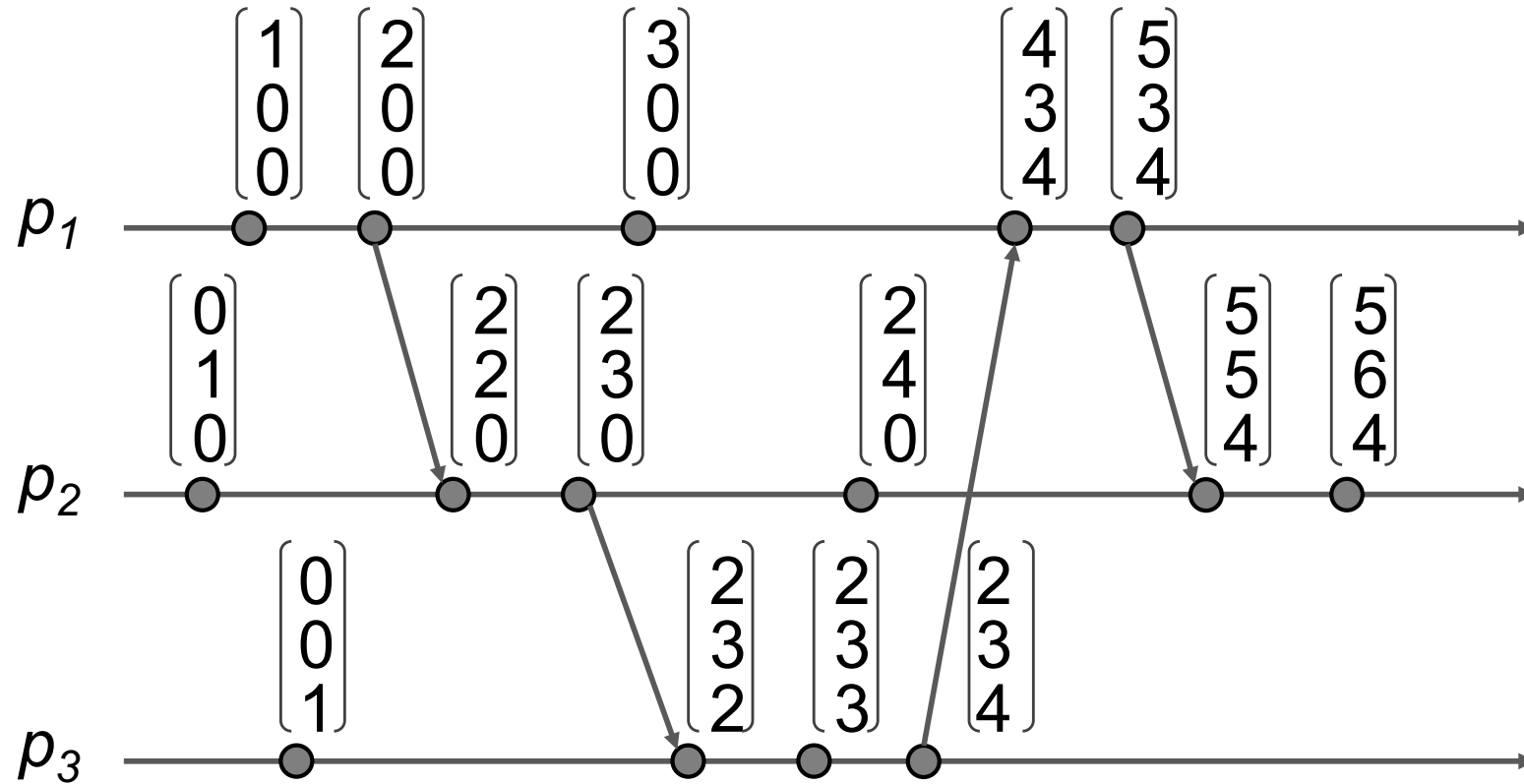
R2: each message transports sender's clock values

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

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

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

Vector Time (2)



Vector Time (3)

Event relations

event a on p_i with timestamp Va

event b on p_k with timestamp Vb

- $a \rightarrow b \Leftrightarrow \forall i: Va[i] \leq Vb[i] \text{ and } \exists i: Va[i] < Vb[i]$
- $a \parallel b \Leftrightarrow \exists i,k: Va[i] > Vb[i] \text{ and } Va[k] < Vb[k]$
- 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)

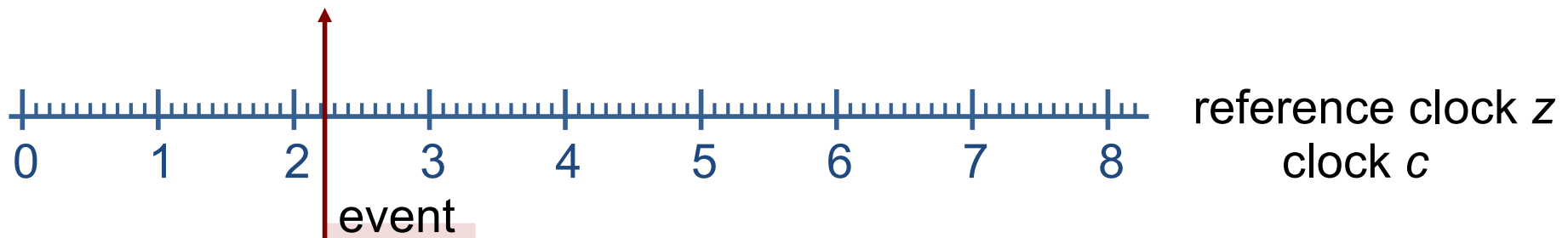
- **Granularity** of a clock c : nominal number of microticks of z between any consecutive microticks of c

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

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



Clock Drift

Real clocks deviate from the reference clock

Clock drift

$$\text{drift}^k_i = \frac{z(\text{microtick}^k_{i+1}) - z(\text{microtick}^k_i)}{g^k}$$

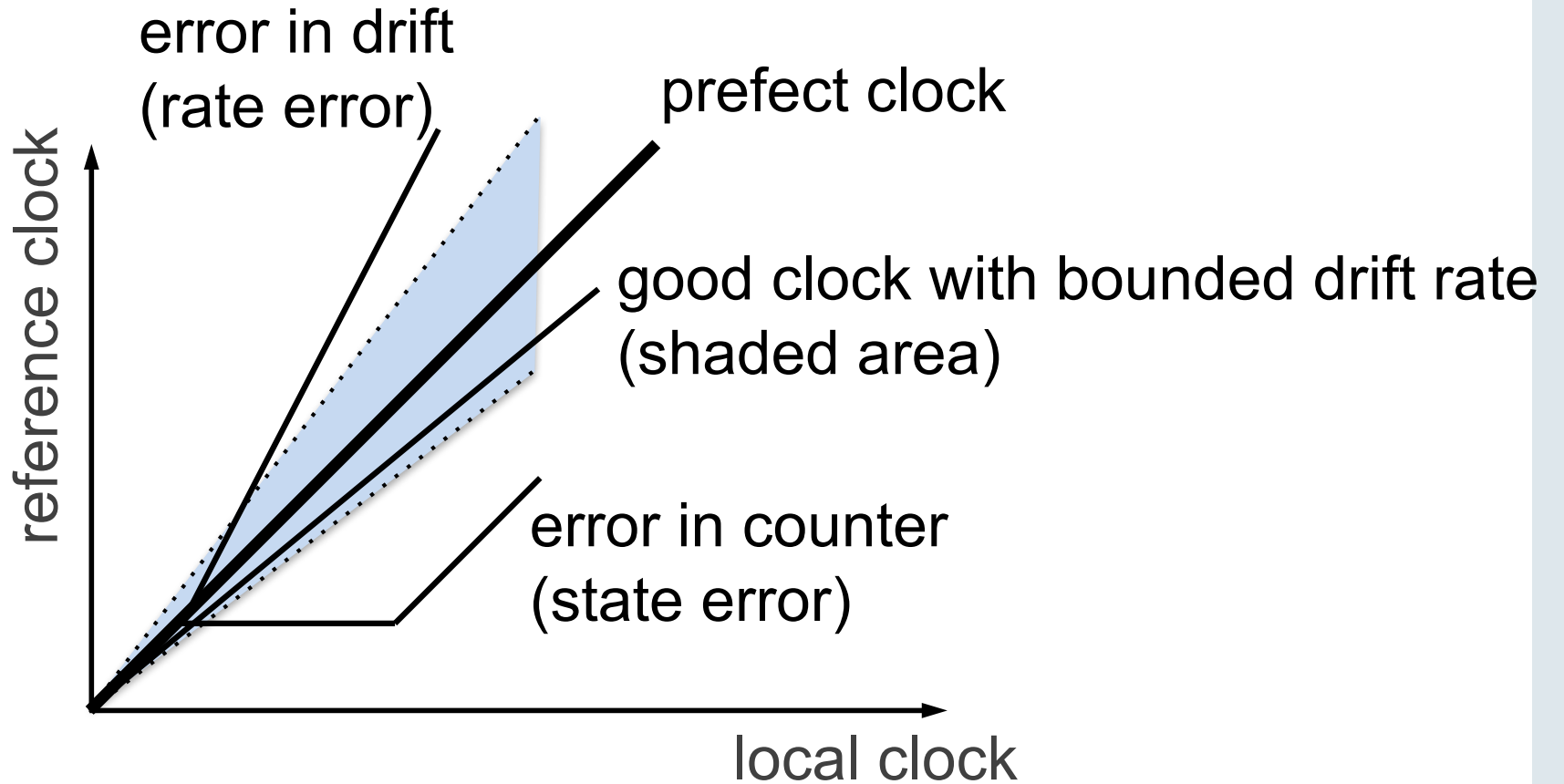
Drift rate

$$\rho^k_i = \left| \frac{z(\text{microtick}^k_{i+1}) - z(\text{microtick}^k_i)}{g^k} - 1 \right|$$

Drift rate of perfect clock: 0

Drift rate of real clocks: $10^{-8} \dots 10^{-2}$

Failure Modes of Clocks



Precision

Offset between two clocks j and k at tick i

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

Precision of an ensemble of clocks $\{1, \dots, n\}$ at macrotick i

$$\Pi_i = \max_{j, k} \{ \text{offset}^{jk}_i \}$$

Internal clock synchronization: mutual resynchronization of an ensemble of clocks in order to maintain a bounded precision

Accuracy

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

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

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

External clock synchronization: resynchronization of a clock with the reference clock

If all clocks of an ensemble are externally synchronized with accuracy A , then the ensemble is internally synchronized with a precision $\Pi \leq 2A$.

Time Standards

International Atomic Time (TAI)

- physical time standard
- defines the second as the duration of 9 192 631 770 periods of the radiation of a specified transition of the Cesium 133 atom.
- chronoscopic timescale, i.e., a timescale without discontinuities.
- defines the epoch, the origin of time measurement, as Jan. 1, 1958 at 00:00:00 hours

Time Standards (2)

Universal Time Coordinated (UTC)

- astronomical time standard, basis for the time on the "wall clock".
- duration of the second conforms to the TAI standard
- number of seconds in an hour occasionally modified by inserting a leap second into UTC to maintain synchrony between the wall-clock time and the astronomical phenomena, like day and night.

Adjusting Time can be Tricky ...

Insertion of a leap second at midnight, New Year's Eve 1995, caused a glitch that affected the time signal for the AP radio broadcast network for hours.

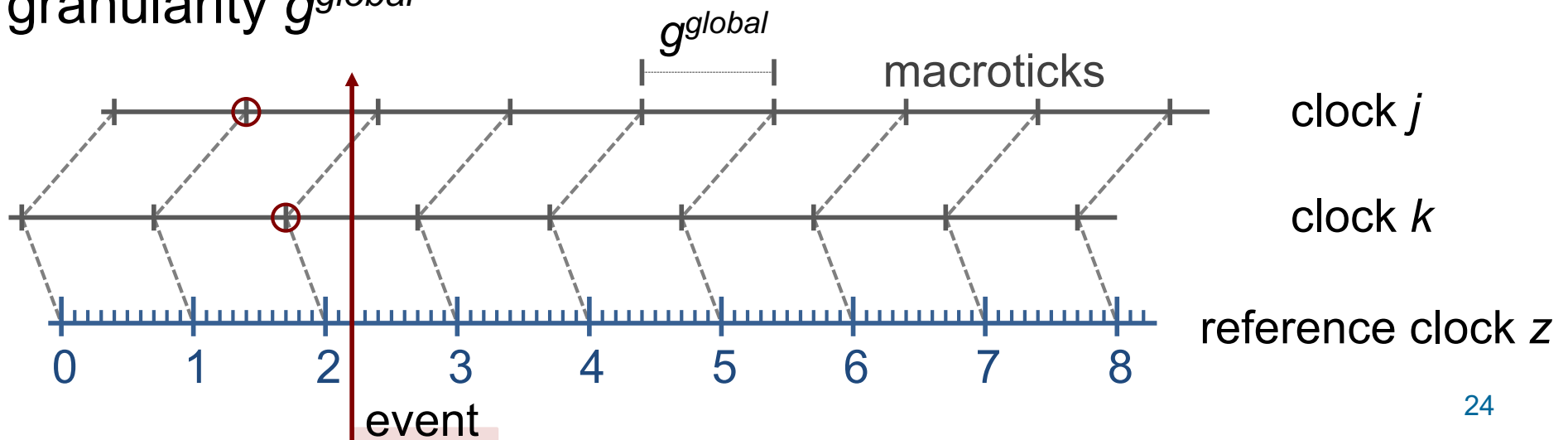
Sequence of events:

1. The day increments to January 1, 1996, 00:00:00.
2. The clock is set back one second, to 23:59:59.
3. The clock continues running.
4. The day changes again. Suddenly it is January 2, 00:00:00.

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}



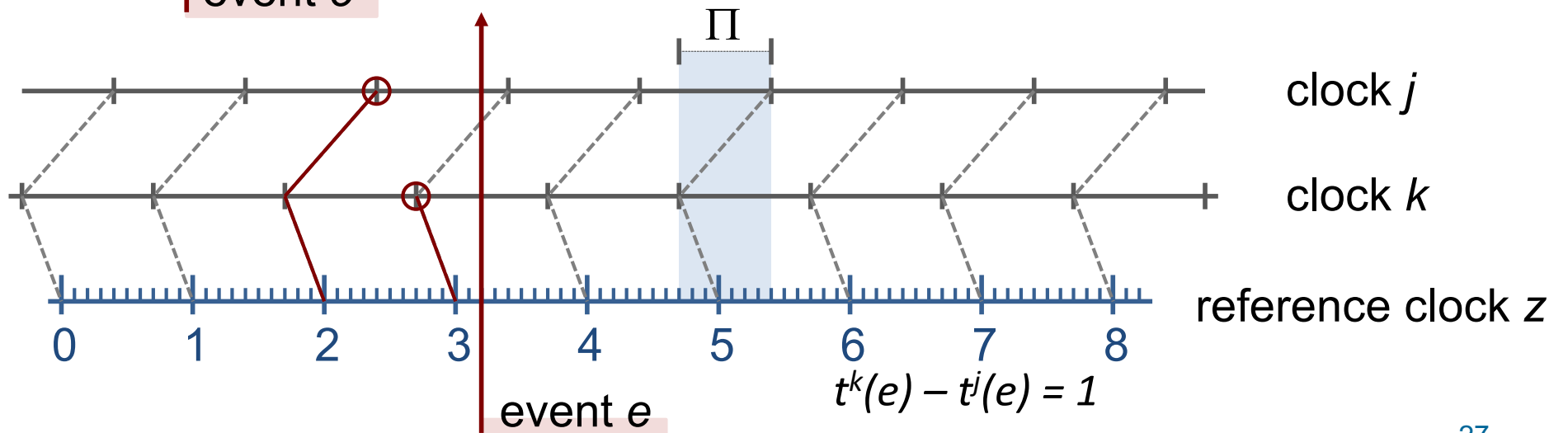
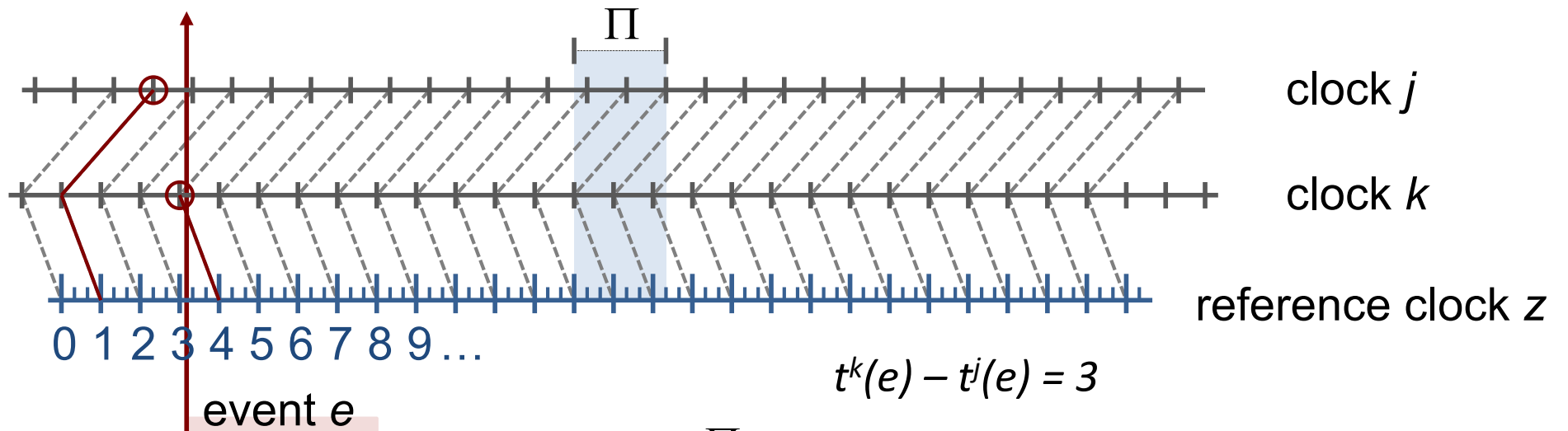
Absence of a Global Timebase

- 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

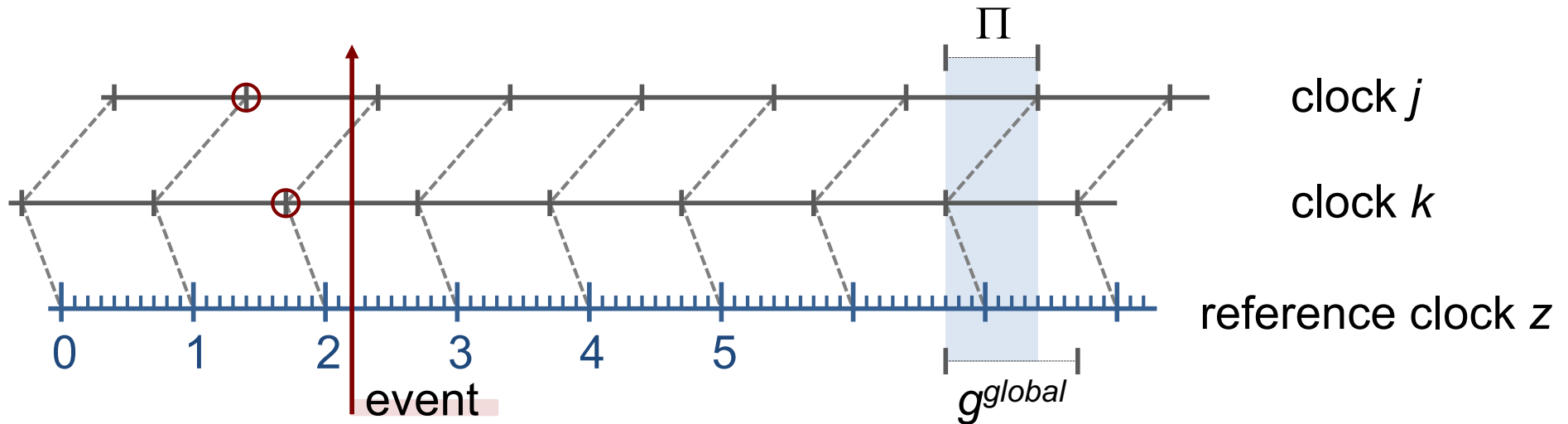
Requirements for a Global Timebase

- 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



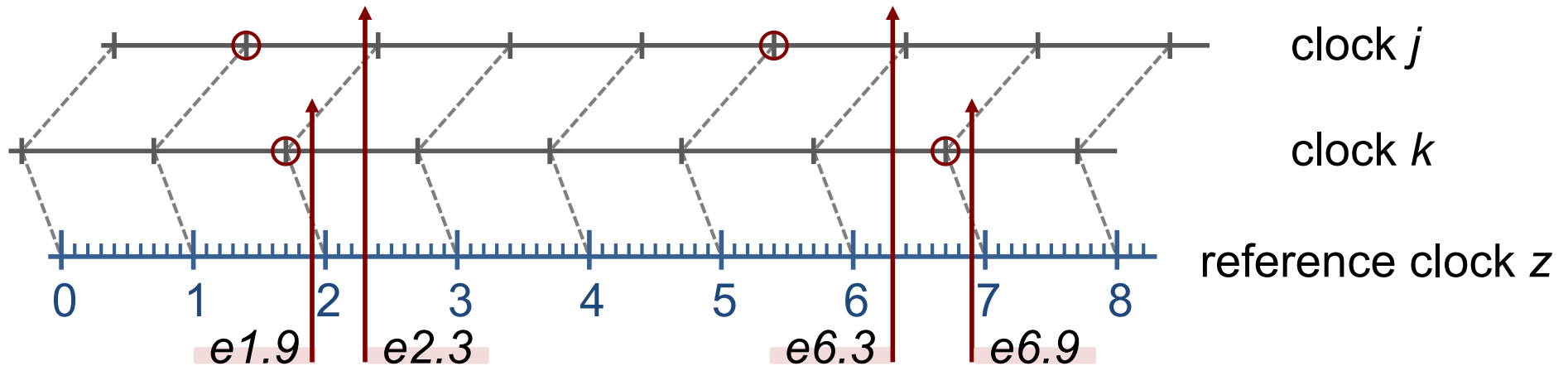
Global time t is **reasonable** if for all local implementations:

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

The reasonableness condition ensures that:

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

Timestamps and Temporal Order



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

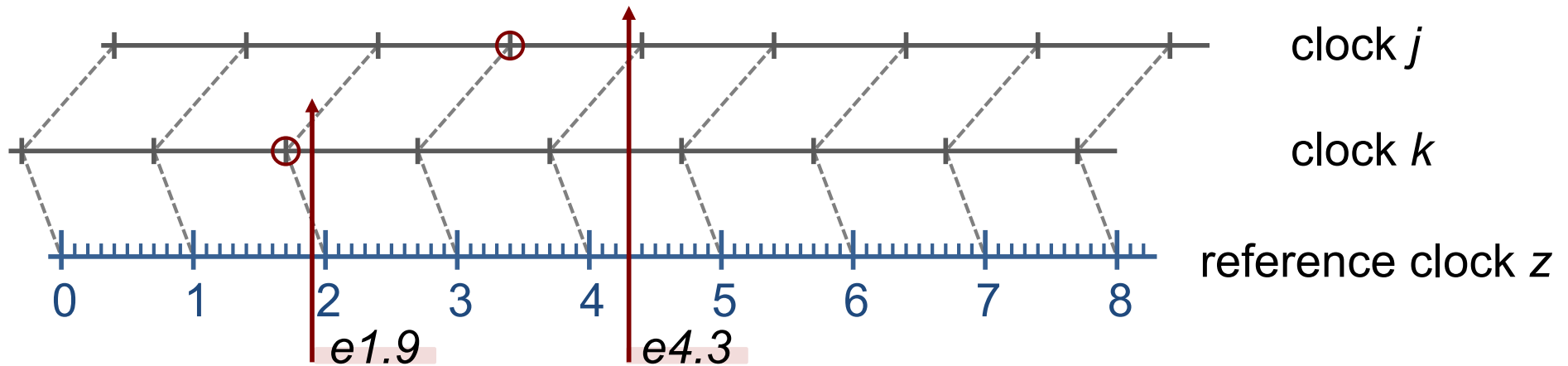
$$t^k(e1.9) > t^j(e2.3)$$

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

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

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

Timestamps and Temporal Order (2)

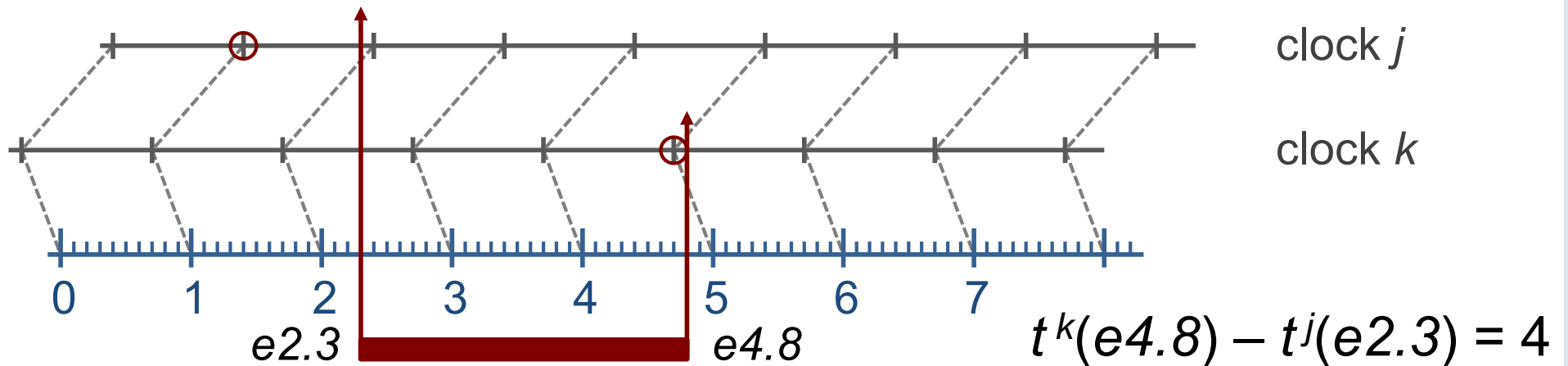
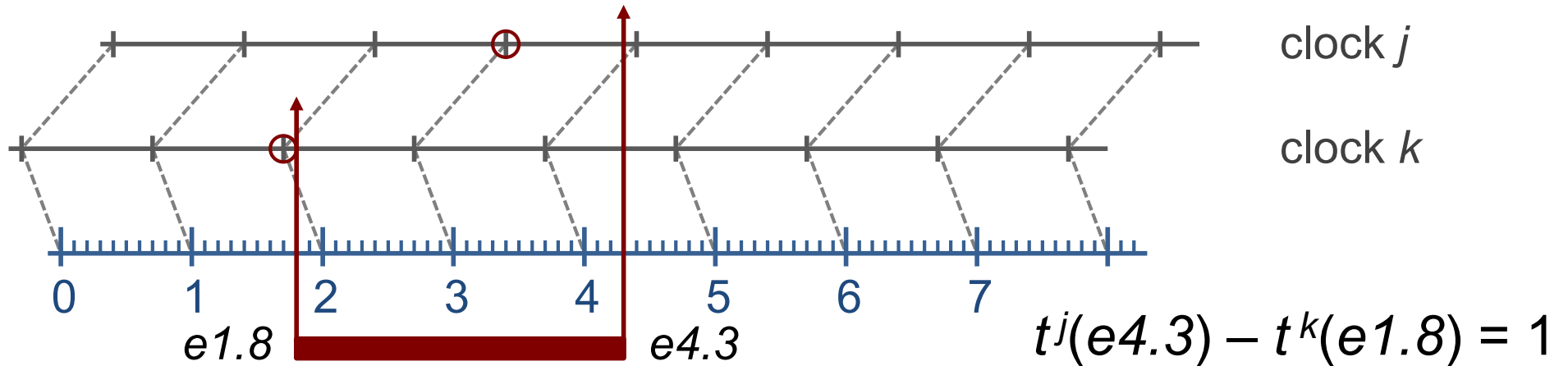


$$z(e4.3) - z(e1.9) = 2.4$$

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

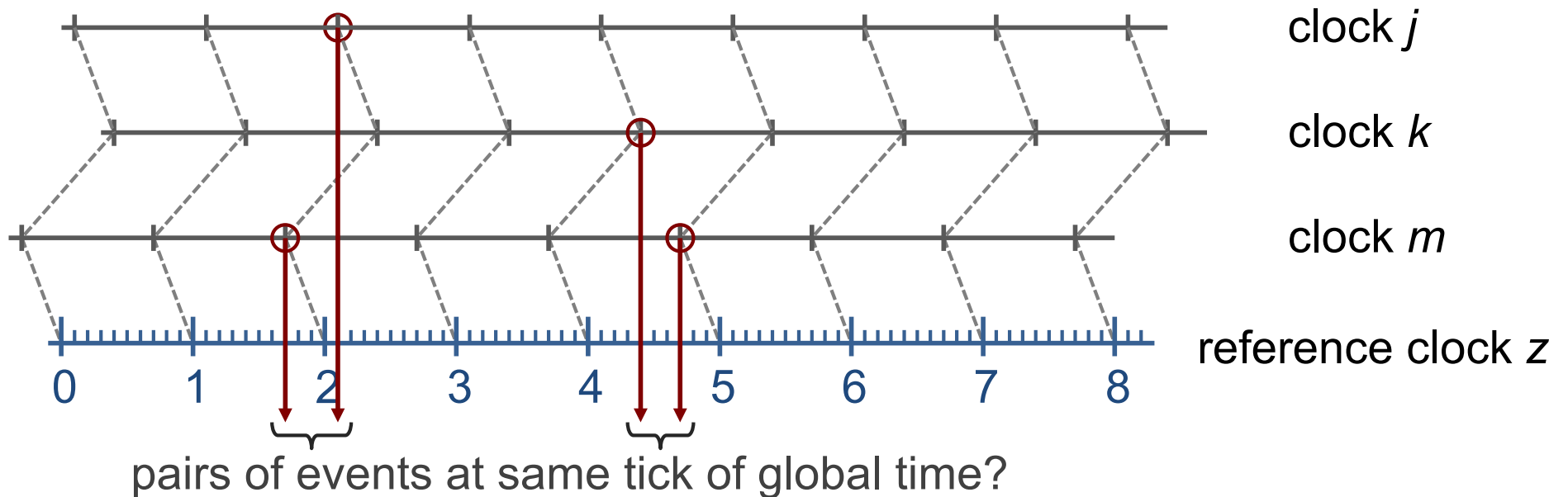
A time distance of $2g^{global}$ between two events is not sufficient to determine their temporal order (if $t^j(a) - t^k(b) = 1$).

Measurement of Durations



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

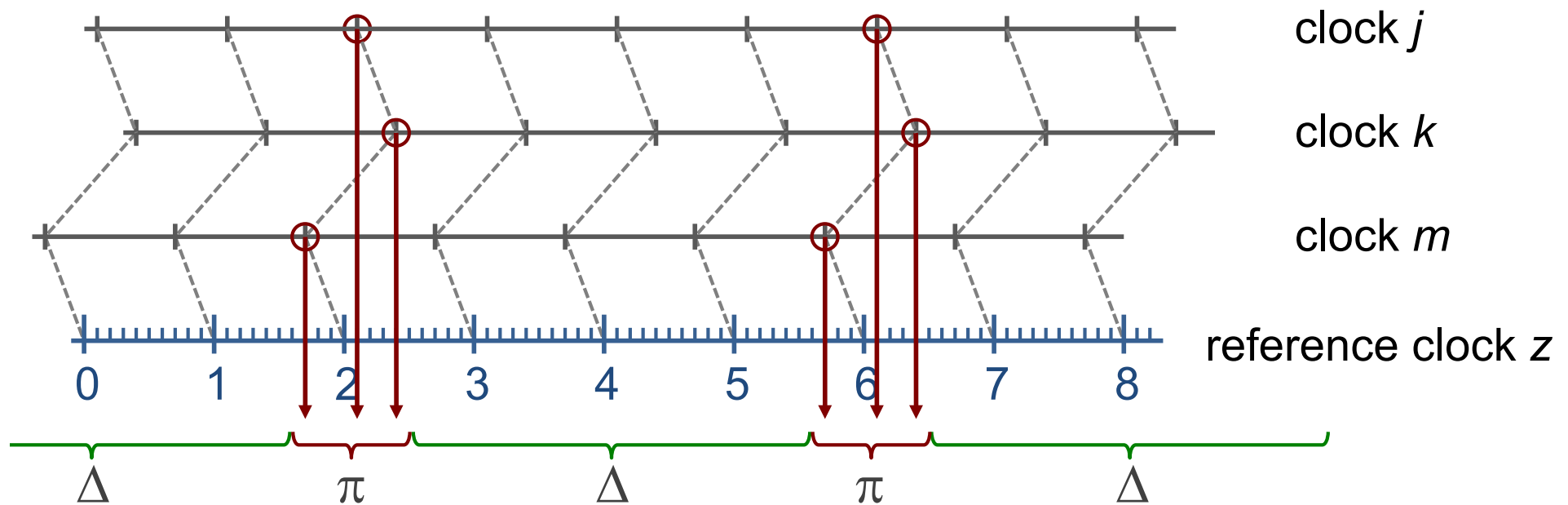
Temporal Relationship between Generated Events



Assumption: nodes generate events at clock ticks

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

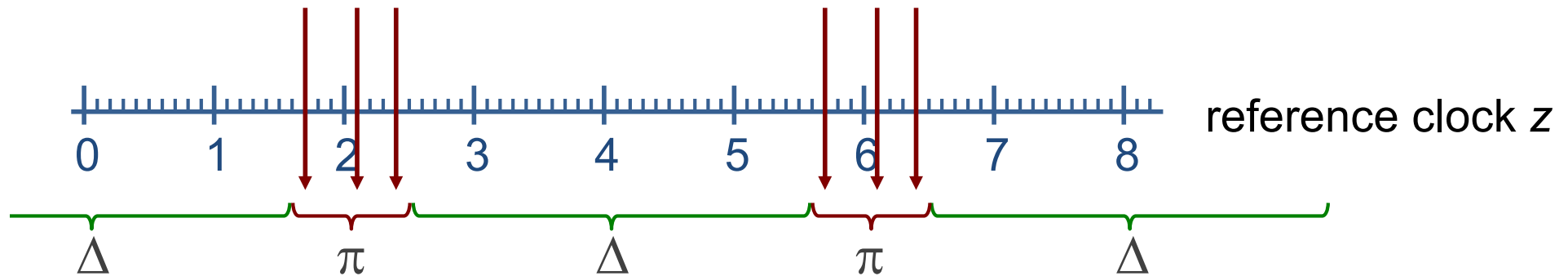
π/Δ -Precedence of Sets of Events



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

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

Dense Time and Sparse Time

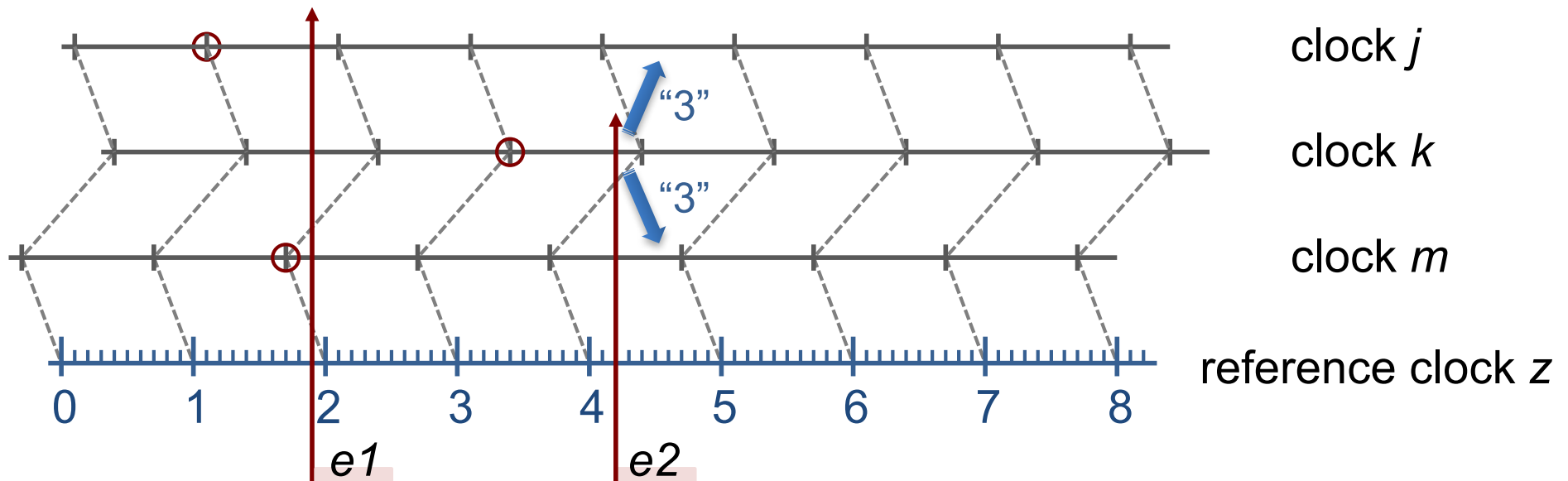


Dense timebase: events are allowed to occur at any time.

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 Observed Events – 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 Observed Events – Dense Time

Conclusions from observations:

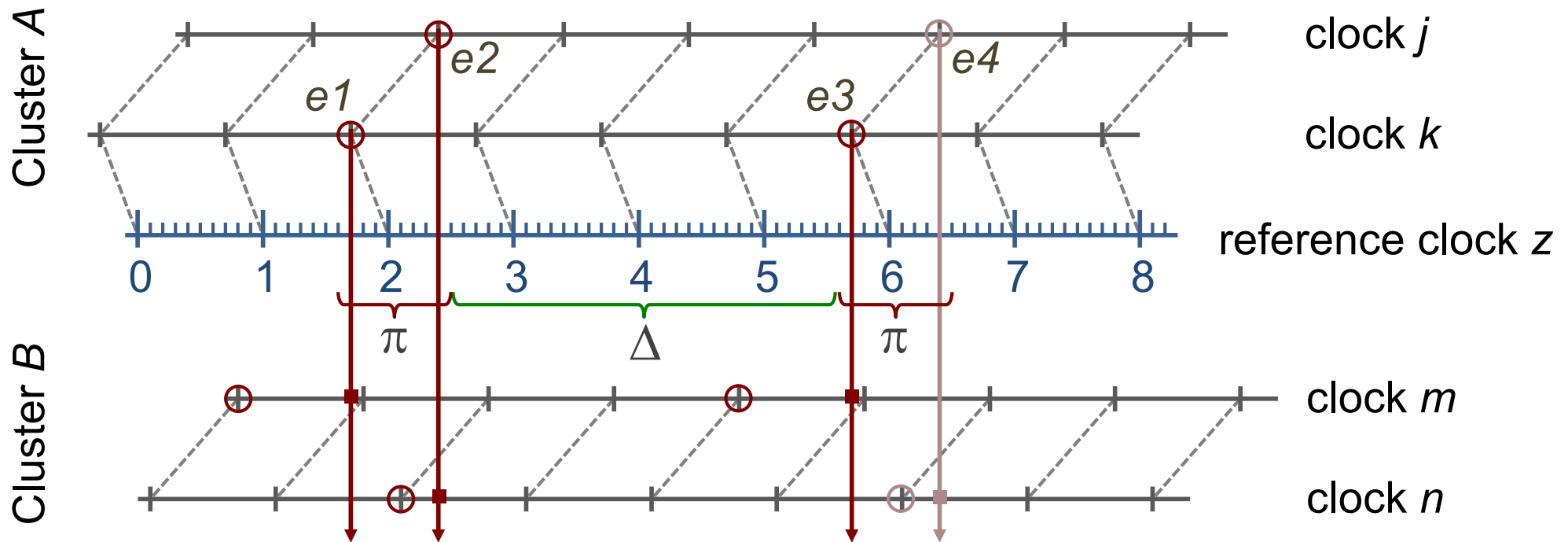
- If a single event is observed by two nodes, the local timestamps for the event may differ by one tick.
 - ⇒ an explicit **agreement protocol** (communication between the nodes) is needed to establish a consistent view about the global time of the event occurrence.
- If two events occur on a dense timeline, then it is impossible to 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 $\Delta \geq 3g^{global}$.

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 $1g/4g$ -sparse;
a $1g/3g$ -sparse timebase is not sufficient (see next slide)

Agreement on Event Order – Sparse Time (2)



$e_1, e_2 \dots$ generated in same activity interval: $t^n(e_2) - t^m(e_1) = 2$

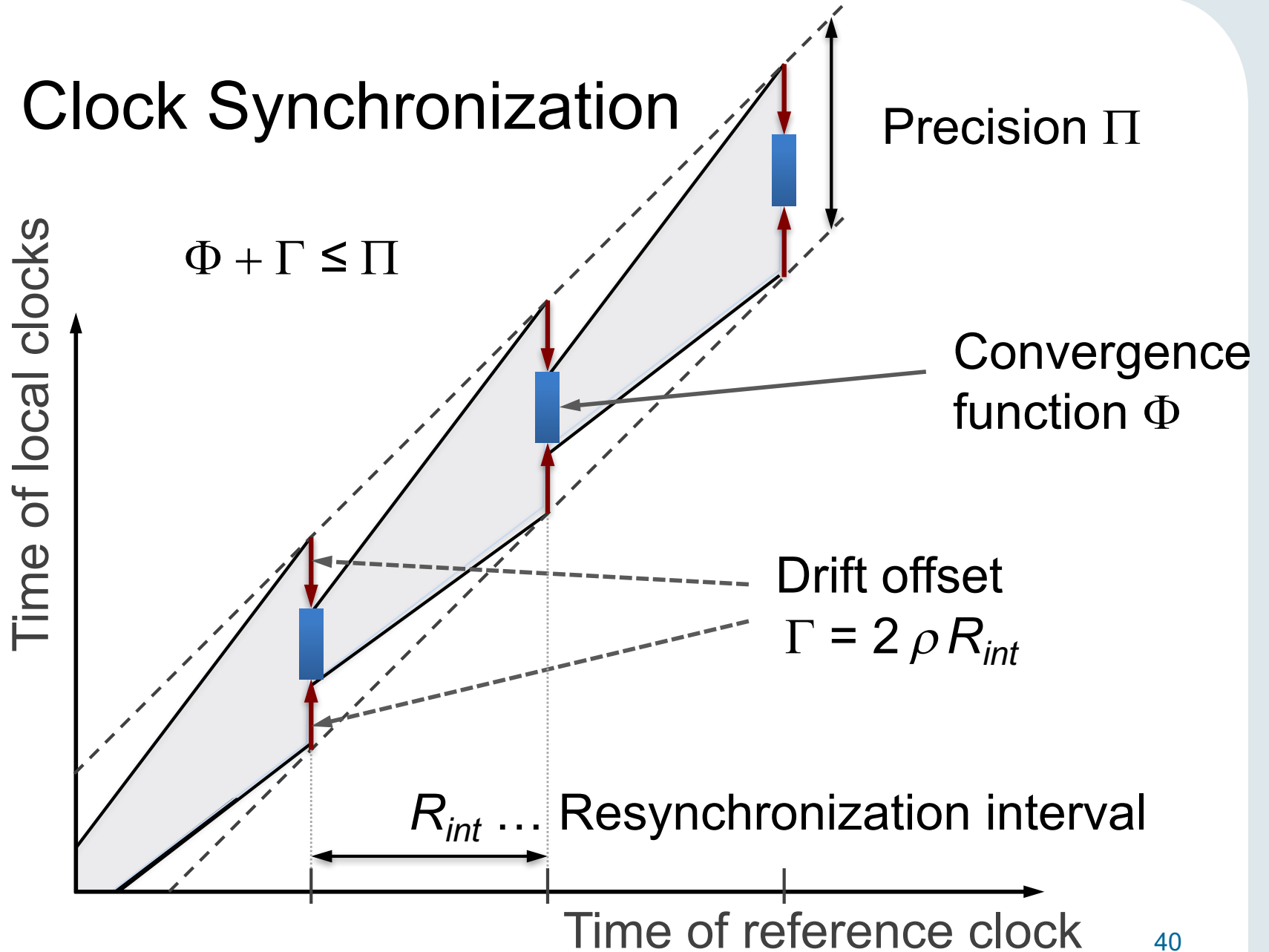
$e_2, e_3 \dots$ gen. in different activity interval: $t^m(e_3) - t^n(e_2) = 2$

Fundamentals in Time Measurement

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: $d_{obs} - 2g^{global} < d_{true}^z < d_{obs} + 2g^{global}$.
- 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 $0/\Delta$ -precedent with $\Delta \geq 3g^{global}$.

Internal Clock Synchronization



Synchronization Condition

To keep the clocks internally synchronized with precision Π , the **synchronization condition** must hold:

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

Φ ... convergence function: max. offset after synchronization;
 depends on synchronization algorithm and
 message latency jitter ε (= transmission-time difference
 between fastest and slowest message, $\varepsilon = d_{max} - d_{min}$)

Γ ... drift offset: divergence of free-running clocks; $\Gamma = 2 \rho R_{int}$

R_{int} ... resynchronization interval

Central Master Algorithm

- Master node sends periodic synchronization messages, containing its local time
- Slaves adjust local clocks
 - Record local arrival time of sync. message
 - Compute difference *master clock* – *local clock*
 - Adjust this difference by latency (known, local parameter)
 - Adjust local clock
- Precision of Central Master Algorithm

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

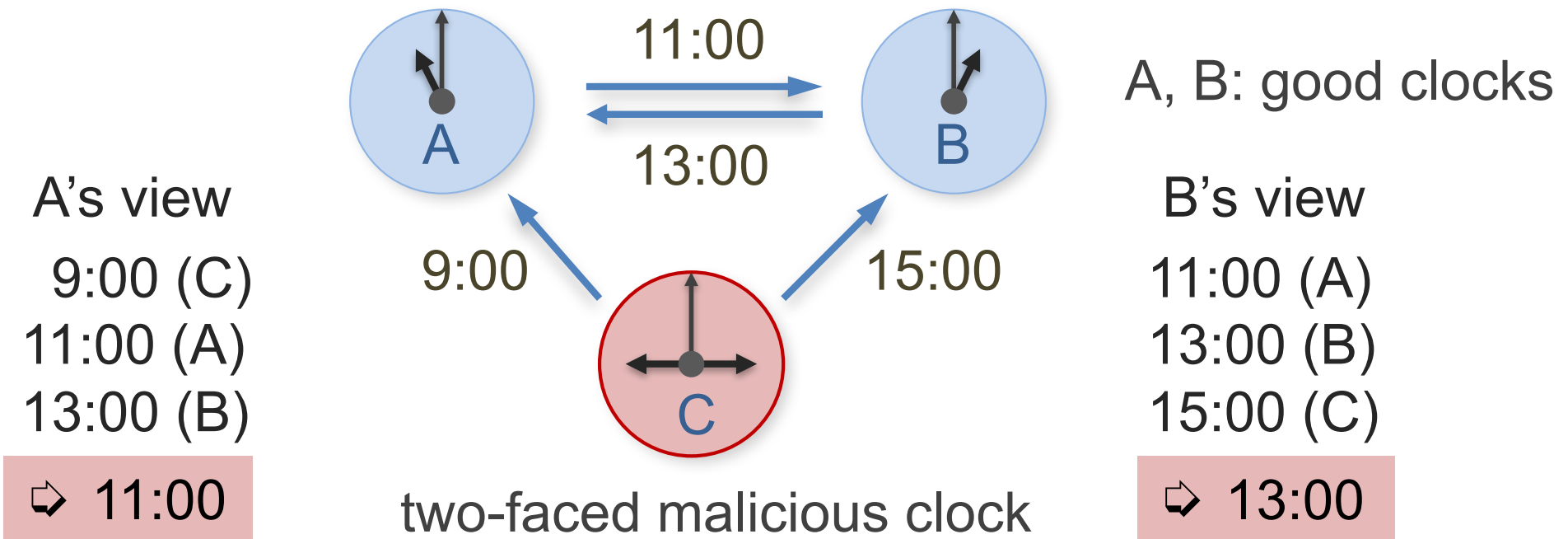
Distributed Clock Synchronization

Use of distributed algorithms to provide fault tolerance;

Typically three phases:

- Nodes exchange messages and acquire information about global-time counters at other nodes.
- Every node analyzes collected information (error detection) and executes the convergence function to compute a correction term for its local global-time counter
- Every node adjusts its local time counter by its correction term

Malicious (Byzantine) Clocks

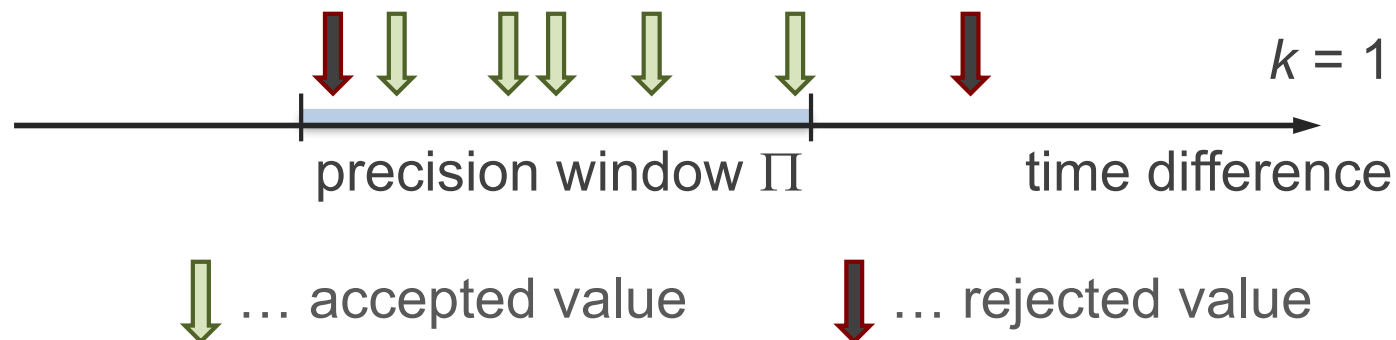


Clock synchronization: in the presence of k Byzantine clocks the number of clocks, N , must be: $N \geq 3k + 1$

Fault-Tolerant Average (FTA) Algorithm

Computation of correction term:

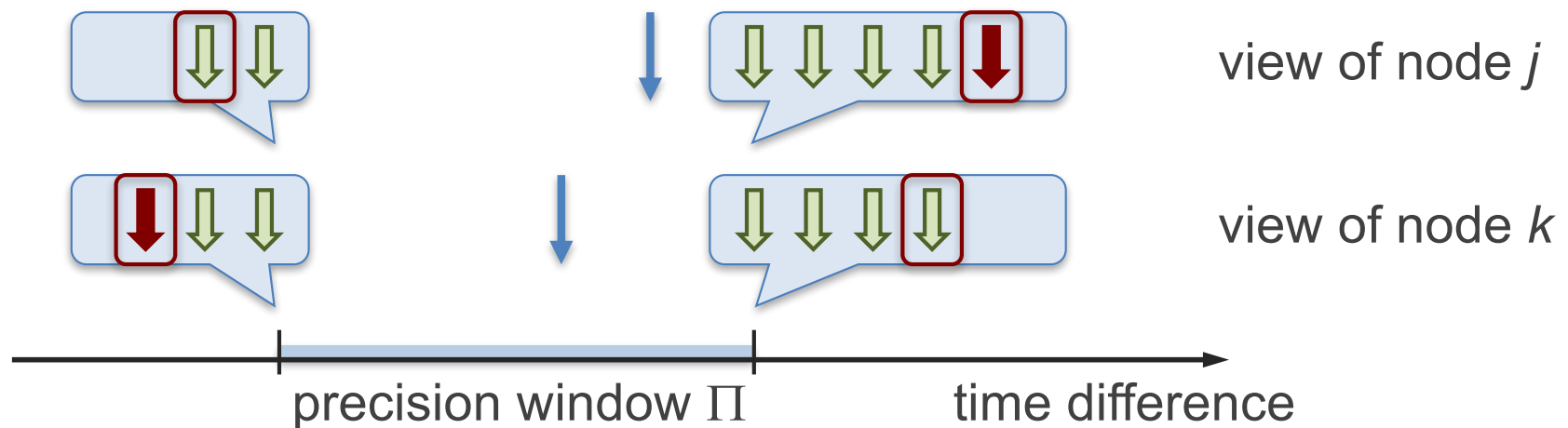
- Calculate differences between local clock and all other clocks
- Sort clock-difference values
- Eliminate k smallest and k largest values
(k ... max number of erroneous clocks)
- Correction term = average of remaining $N - 2k$ time differences
(state correction vs. rate correction)



FTA Algorithm – Effect of Byzantine Clock

Worst-case effect of a Byzantine node:

- Byzantine time values at different ends of precision window
- Error term of a Byzantine error: $E_{byz} = \Pi / (N - 2k)$



... good value
 ... malicious val.
 ... rejected val.
 ... calc. average
 46

Precision of the FTA Algorithm

Convergence Function

$$\Phi(N, k, \varepsilon) = k \Pi / (N - 2k) + \varepsilon$$

Precision

$$\Pi(N, k, \varepsilon, \Gamma) = (\varepsilon + \Gamma) \frac{N - 2k}{N - 3k} = (\varepsilon + \Gamma) \mu(N, k)$$

$\mu(N, k)$ is called the Byzantine error term

		number of nodes N						
$\mu(N, k)$		4	5	6	7	10	15	20
k	1	2	1.5	1.33	1.25	1.14	1.08	1.06
	2				3	1.5	1.22	1.14
	3					4	1.5	1.27

Interactive Consistency Algorithm

Eliminates Byzantine error term

- After collecting the time values of all other clocks, every node sends its view of the clock ensemble to all other clocks
↳ extra communication round!
- Nodes have global view; can identify Byzantine nodes
- Correction based on matrix of time vectors of all views
- $\mu(N, k) = 1$

Limit to Internal Clock Synchronization

Lundelius and Lynch show limits of clock synchronization:

The best achievable precision even with perfect clocks is

$$\Pi_{opt} = \varepsilon (1 - 1/N)$$

Clock-Synchronization Quality Parameters

- Drift offset $\Gamma = 2 \rho R_{int}$
- Delay jitter $\varepsilon = d_{max} - d_{min}$
- Byzantine failures: rare events
- Clock synchronization algorithms: effect on sync. quality is small compared to delay jitter

Keeping the Drift Offset Small

Minimize relative drift rates of clocks

- Use rate master with precise clock in each cluster
- Adjust rates of local clocks to rate of the master
- Use state correction in FTA
 - ↳ mask errors in rate correction of local clocks

Jitter of Synchronization Messages

Message jitter ε depends on where message timestamps are inserted and interpreted

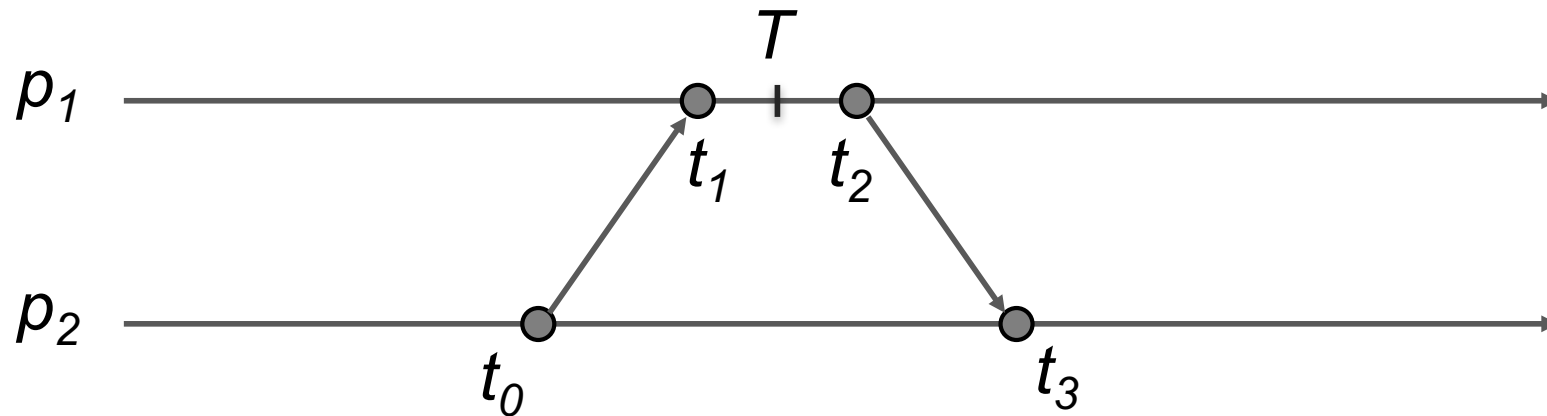
Message assembly/interpretation	appr. range of jitter
Application software level	$500 \mu s \dots 5 ms$
Operating system kernel	$10 \mu s \dots 100 \mu s$
Hardware: communication controller	$< 10 \mu s$

Quality Attributes of a Global Time Base

- Precision
- Accuracy
- Fault tolerance: number and types of faults the system of clocks can tolerate
- Blackout survivability: blackout duration that can be tolerated without losing synchronism

Cristian's Algorithm

Request time and evaluate reply



Time-request from p_2 to p_1 at t_0

Reply from p_1 arrives at t_3 : contains T , round-trip time $d = t_3 - t_0$

Clock sync: p_2 sets local time to $T + d / 2$

Clock sync. error $\leq d / 2$

Network Time Protocol (NTP)

- Built on idea of Christian's algorithm
- Hierarchy of time servers
 - Class 1: connected to atomic clocks, GPS clocks
 - Class 2: receive time from Class 1 servers, synchronize with other Class 2 devices
 - Class 3: receive time from Class 2 servers, ...
- Clock correction based on statistical analysis of $t_0 \dots t_3$ of multiple clock readings

Precision Time Protocol (PTP) builds on NTP, uses hardware support for message timestamping to keep ε small

External Clock Synchronization

Synchronize clock ensemble to an external time reference

Example: GPS, achievable accuracy below $1\mu\text{s}$

Complementary properties of internal/external synchronization:

- Internal clock synchronization:
high availability, good short-time stability
- External clock synchronization:
long-term stability, possibly lower availability

Promising combination:

gateway to external time reference = rate master for
internal synchronization

Lessons Learned

- Why we need time ...
- Temporal and causal order
- Logical time (Lamport time, vector time)
- Event, duration
- Global time and clocks
- Internal clock synchronization
- External clock synchronization