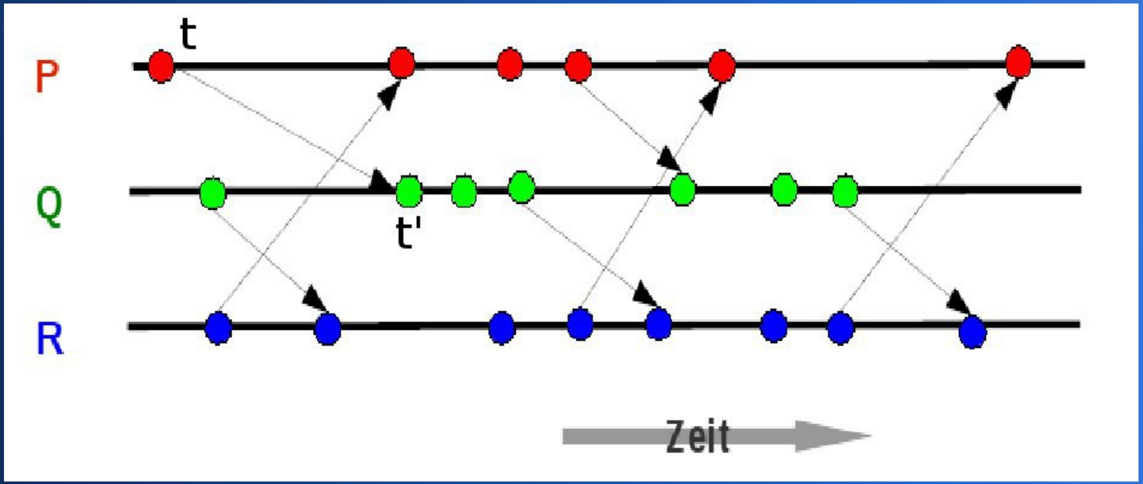
Formale Definitionen zum Zeitbegriff

- Zeitliche Ordnung
  - Verallgemeinerung der üblichen Zeit
  - übliche Zeit:
    - UTC: Sekunden, Nachbildung Erdrotation, UT1
    - TAI: Sekunden ab 1.1.1958 (Cs-Atome)
    - Datenformat: Integerwert (64 Bit)
    - „totale Ordnung“, ohne logische Abhängigkeiten
  - logische Zeit:
    - Ereignisse in verteilten Systemen
    - **gleichzeitig = unabhängig**

Definition: Logische Zeit

- partiell geordnete Menge M (hier: Zeitstempel)
- Relation „≤“ (happened-before)
- Ereignisse in selbem Prozess:
  „happened before“ klar

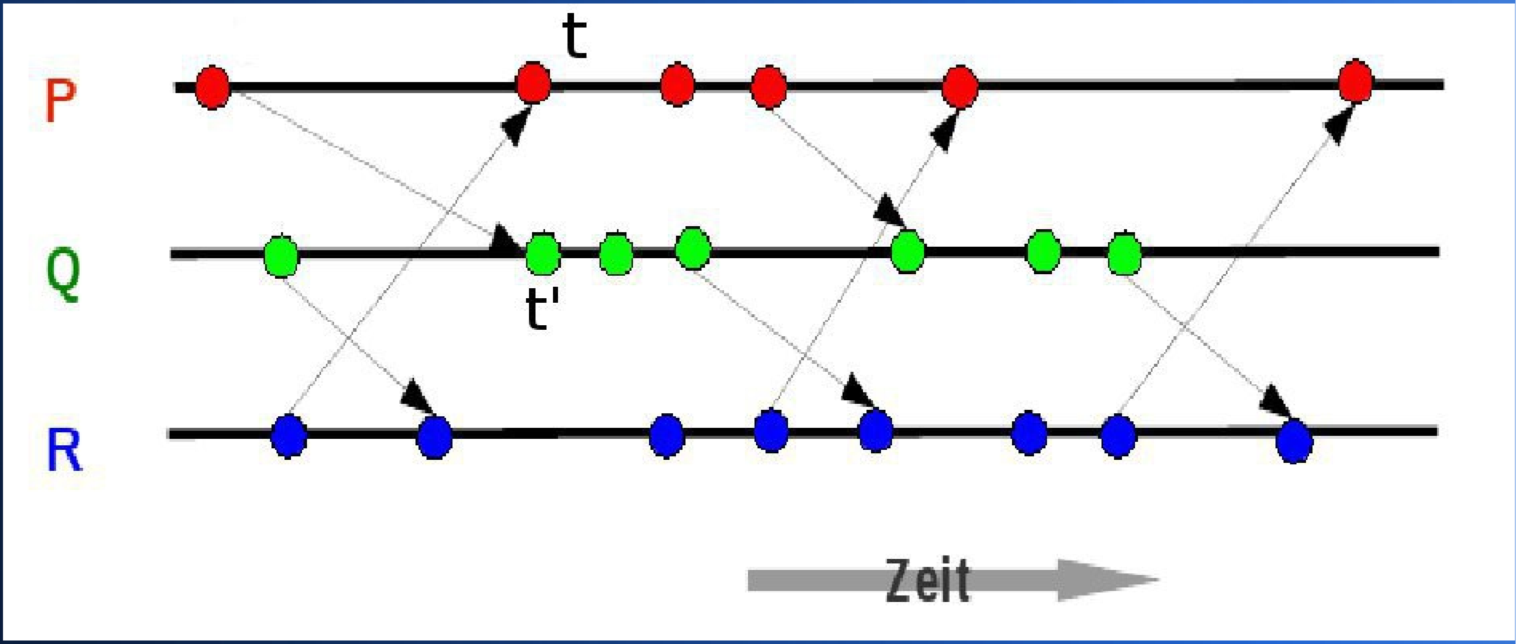
  Senden „happened before“ *Empfangen*
- t≤t', Relation ist
  - reflexiv
  - transitiv
  - antisymmetrisch



„≤“ beschreibt *kausale Abhängigkeit*

Logische Zeit: Gleichzeitigkeit

- t,t' gleichzeitig (kausal unabhängig), wenn keines der folgenden gilt
  - $t \leq t'$
  - $t' \leq t$
  - $t' = t$



Implementierung?

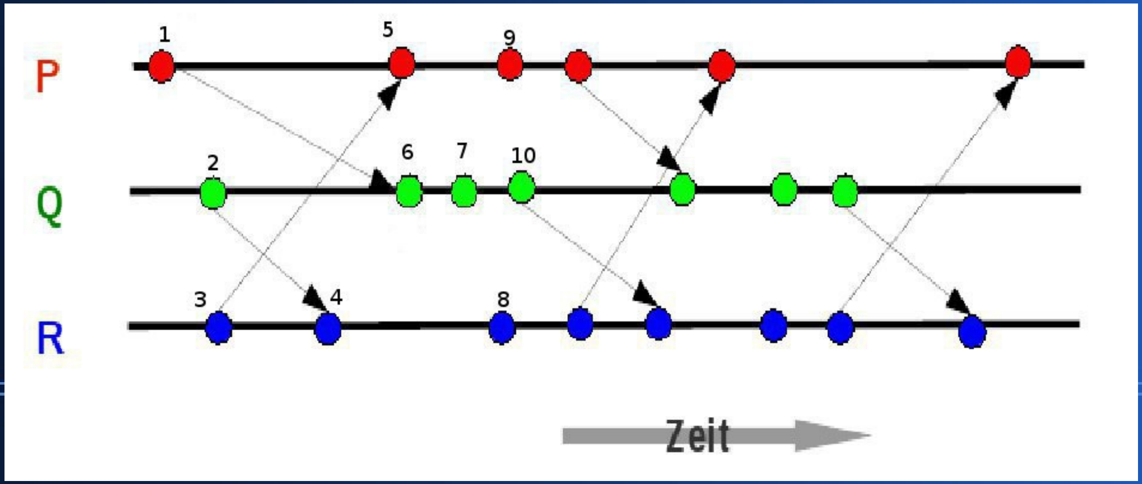
Aus welchem Bereich

soll t stammen

???

Implementierung!

- Lamport 1978:
- Lamport-Uhr bzw. Lamport-Zeit
- Vorstellung eines idealen Beobachters
  - Inkrementiert Zeit bei jedem Ereignis



Lamport-Zeit

Jeder hat seinen (lokalen) Zeitstempel t.

Senden: t = t+1
send(message,t)

Empfangen(message,t_msg):
t = max(t_msg,t)+1

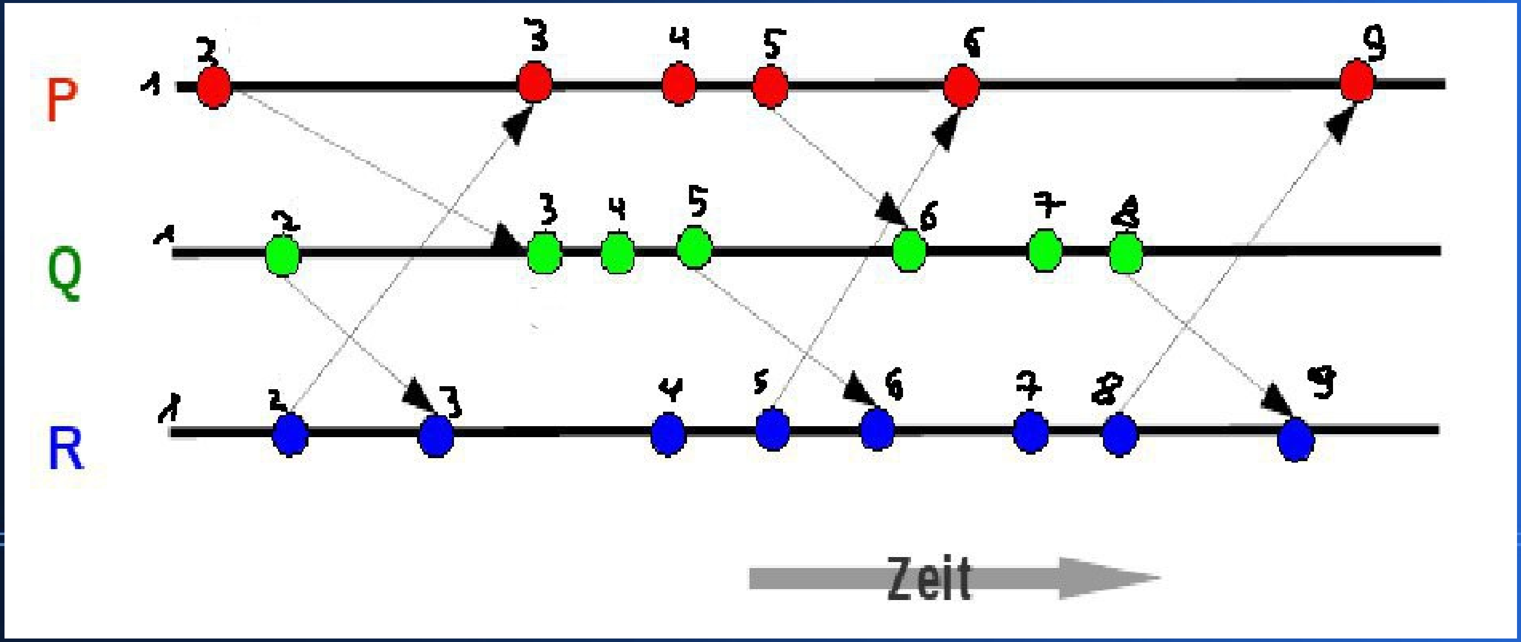
Konsequenzen?

Lamport-Zeit (Konsequenz)

- Ereignisse e, e' kausal abhängig e -> e' => $t(e) \leq t(e')$
- Gilt Äquivalenz? D.h. $t(e) \leq t(e')$ => e->e'

Lamport-Zeit (Konsequenz)

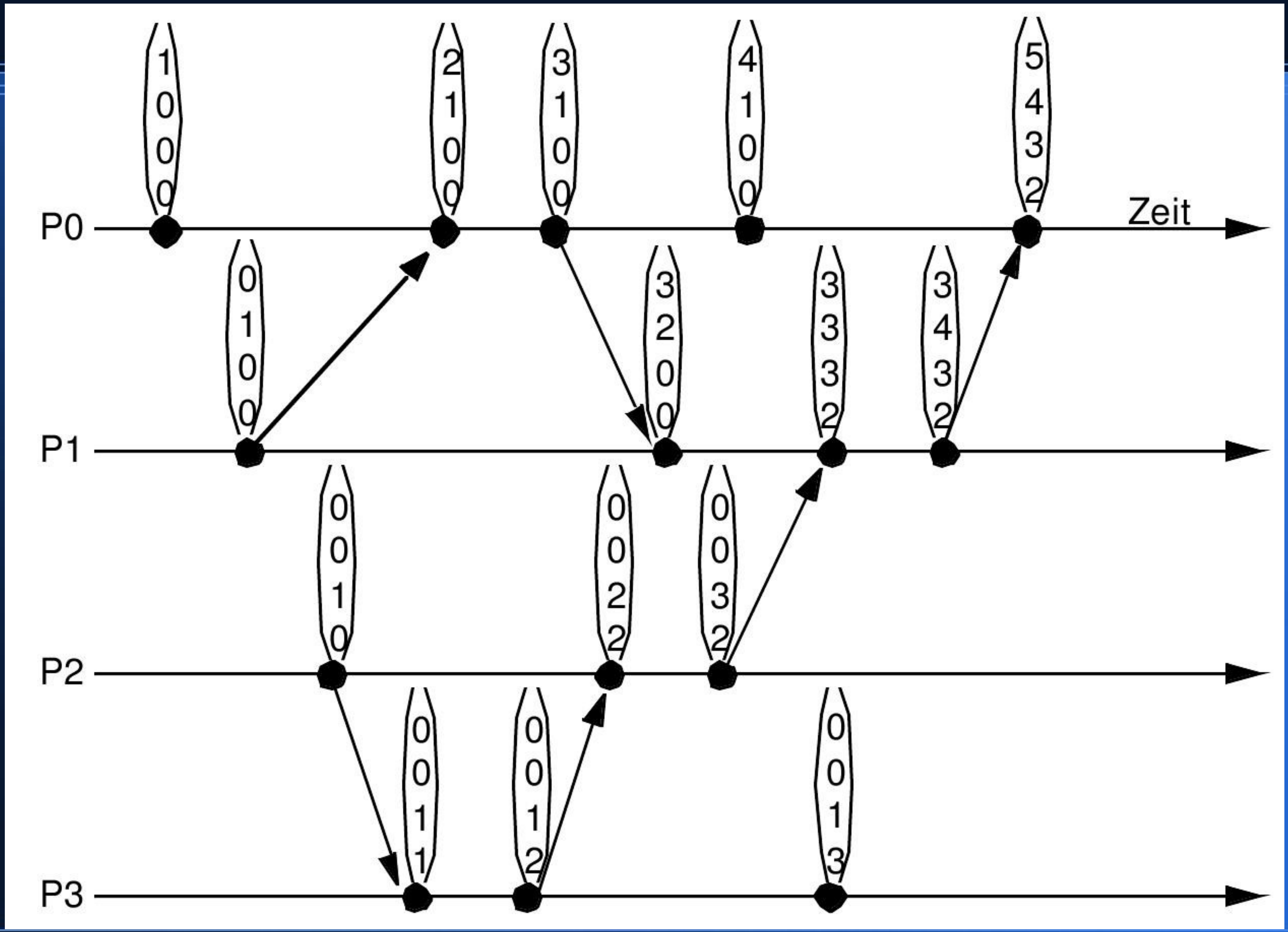
- Ereignisse e, e' kausal abhängig e -> e' => t(e) < t(e')
- Gilt Äquivalenz? D.h. t(e) < t(e') => e->e'



Verbesserung der Lamport-Zeit

- aus $t(e) \leq t(e')$ schließen, dass $e \leq e'$
- jeder approximiert die Zeit aller Teilnehmer
- Vektorkomponente *i* = Zeit von Teilnehmer *i*
- Ordnung von Vektoren? Partielle Ordnung!

Vektorzeit



Quelle: Michael Weber, Verteilte Systeme, Sommersemester 2000, Kapitel 5

Vektorzeit: kausale Abhängigkeit?

- angenommen, für zwei Ereignisse gilt e $\leq$ e'
- Sender *i*, Empfänger *j*, Vektoren *v(i), v(j)*
- Pfeil von Prozess i nach Prozess j
- $v(j,k) = \max(v(i,k), v(j,k))$ für alle k
- => $v(i) \leq v(j)$
- => alle Vektoren auf einem Pfad erfüllen „$\leq$“

Vektorzeit: kausale Abhängigkeit (2)

Forschungsresultat Mattern (1992)
„On the relativistic structure of logical time in distributed systems“

falls $v(i) \leq v(j)$,
e mit Zeit $v(i)$,
e' mit Zeit $v(j)$

dann e -> e'

Aussagen über verteilte Berechnungen: Prädikate

$$\forall n \in \mathbf{N} \quad \exists \epsilon > 0 : \qquad |x_n - \epsilon| > 0$$

Teil einer Aussage

Quantoren

Prädikate sind Aussagen über Eigenschaften von Objekten

lat. praedicare = zusprechen

Stabile Prädikate (1)

- Aussagen, die ab einem Zeitpunkt der verteilten Berechnung dauerhaft gelten

Beispiele:

- Terminierung
- Objekt ist Garbage
- Deadlock

Stabile Prädikate (2)

Formalisierung

monotone Funktion
f: Zustand $\longrightarrow$ M

M muss Ordnungsrelation haben

Prädikat *evaluiert* Zustand einer Berechnung
- gesamte Berechnungshistorie
- alle Nachrichten

Stabile Prädikate (3)

- Beispiel:
  M={*false*,*true*}   geordnet mit   *false* < *true*

  Zustand z, Folgezustand z'

  $f(z) \leq f(z')$

- f(z) = „z ist Terminierungszustand“
- Zustände als Mengen, Halbordnung

Begründungen für Korrektheit: *Safety* und *Liveness*

- *Safety* = something bad will never happen

  Algorithmus erfüllt Invarianten

- *Liveness* = something good will eventually happen

  Algorithmus tritt in einen guten Endzustand ein

Safety und *Liveness*, Beispiele

- *Safety* = something bad will never happen
  - die Bilanz von Bank X ist immer > 0
  - ein Deadlock kann nicht auftreten
- *Liveness* = something good will eventually happen
  - Initiator erhält Kenntnis von einem Ergebnis
  - Algorithmus terminiert
  - Terminierung wird entdeckt

Beispiele trivialer Terminierungsalgorithmen

- 1. Kontrollalgorithmus
  - Meldet immer „vielleicht“

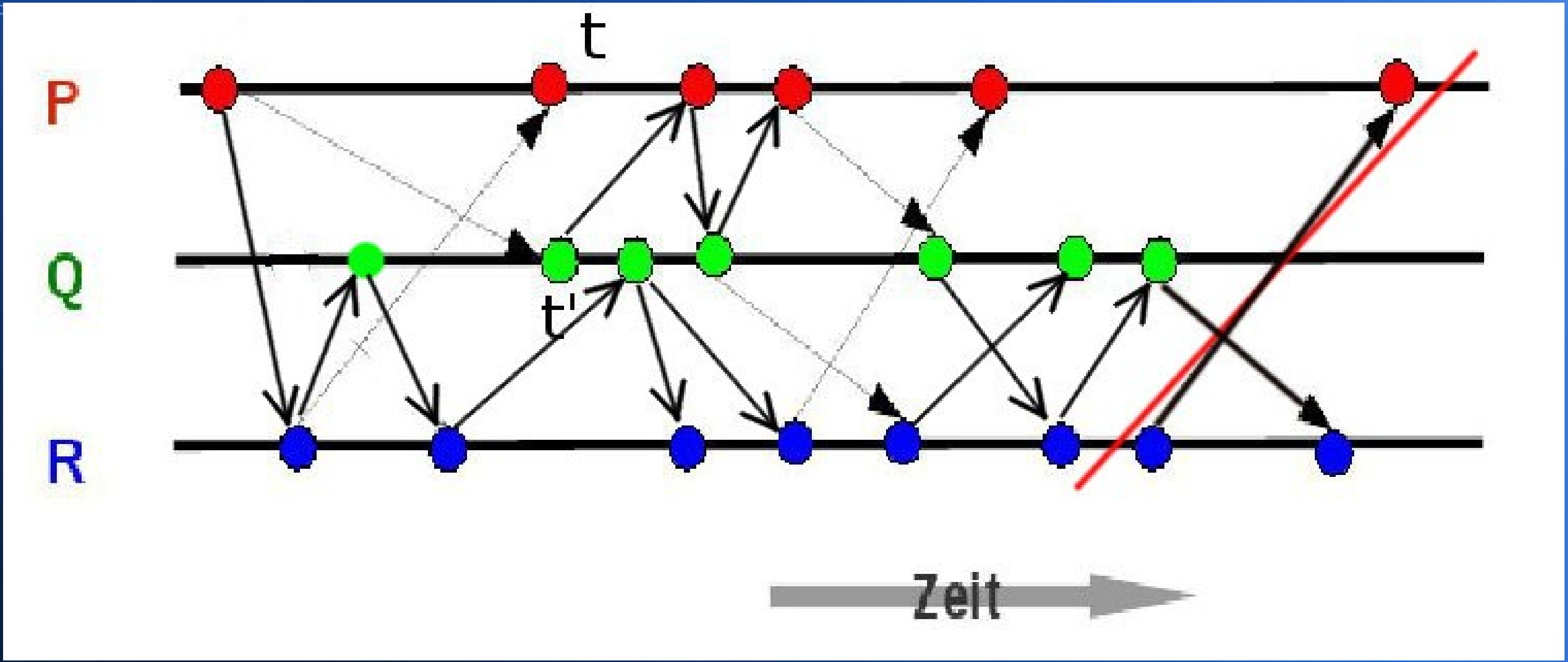
➔ Kontrollalgorithmus ist safe: sagt nichts Falsches

- 2. Kontrollalgorithmus
  - Meldet immer „ja“

➔ Kontrollalgorithmus ist live: schließlich richtig

Kunst: verteilter Algorithmus, der live und safe ist

Terminierung



Ideen ? Ziel: Beweis, dass keine Nachrichten unterwegs