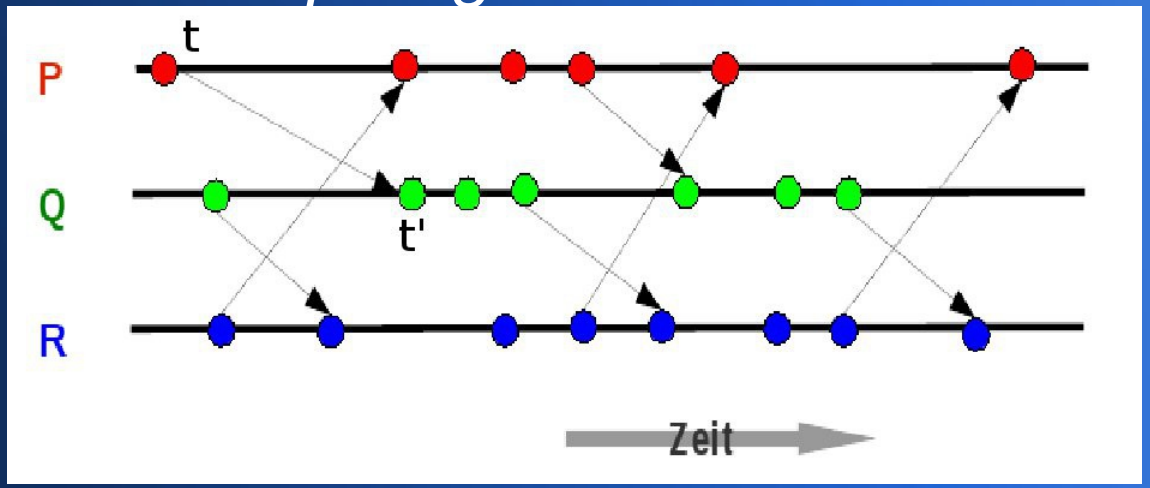
Formale Definitionen zum Zeitbegriff

- 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

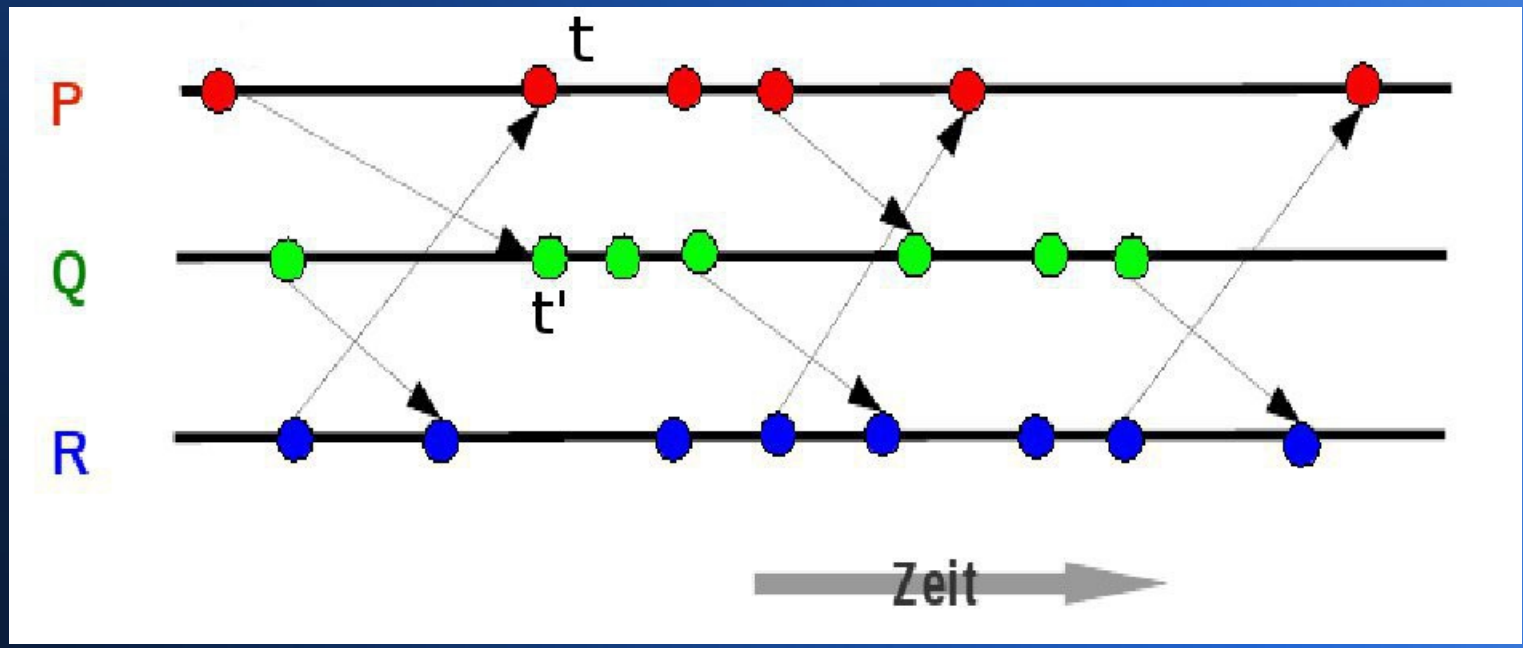
- partiell geordnete Menge M
- Relation „<“ (happened-before)
- Ereignisse in selbem Prozess: „happened before“ klar
- *Senden* „happened before“ *Empfangen*
- t<t', Relation ist
  - reflexiv
  - transitiv
  - antisymmetrisch



„<“ beschreibt *kausale Abhängigkeit*

Logische Zeit: Gleichzeitigkeit

- t,t' gleichzeitig (kausal unabhängig), wenn keines der folgenden gilt
  - t<t'
  - t'<t
  - t'=t

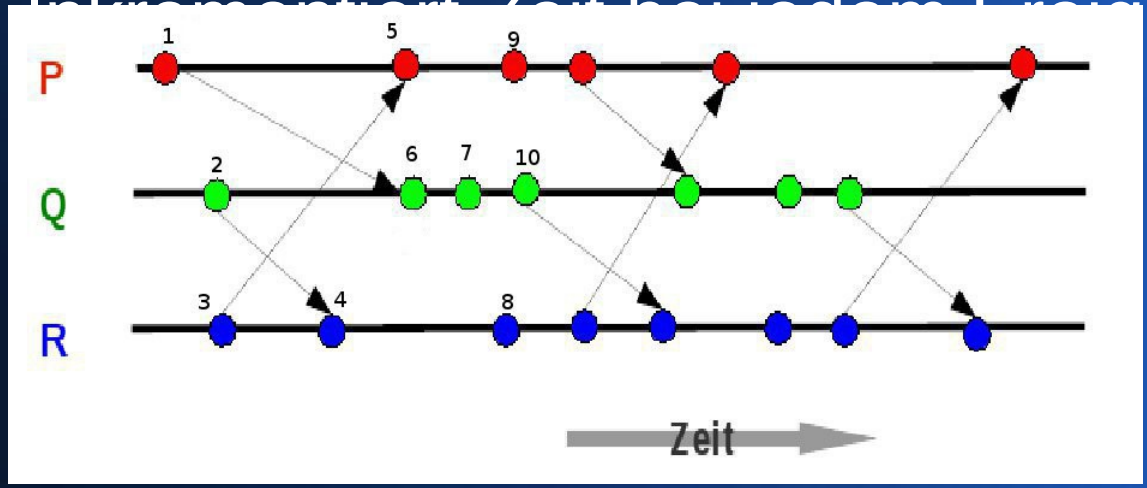


Implementierung?

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?

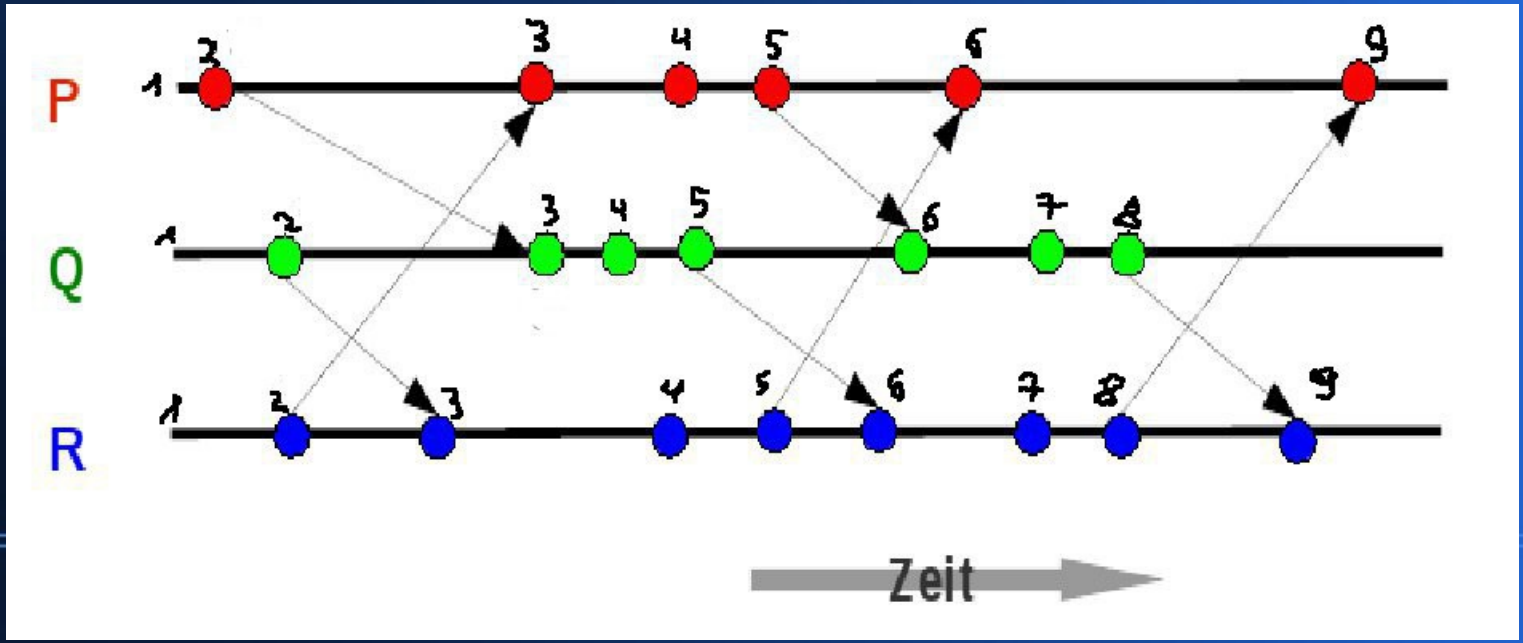
Lamport-Zeit (Konsequenz)

- Ereignisse e, e' kausal abhängig e -> e'
  => t(e) < t(e')

- Gilt Äquivalenz? D.h. t(e) < t(e') => e->e'

Lamport-Zeit (Konsequenz)

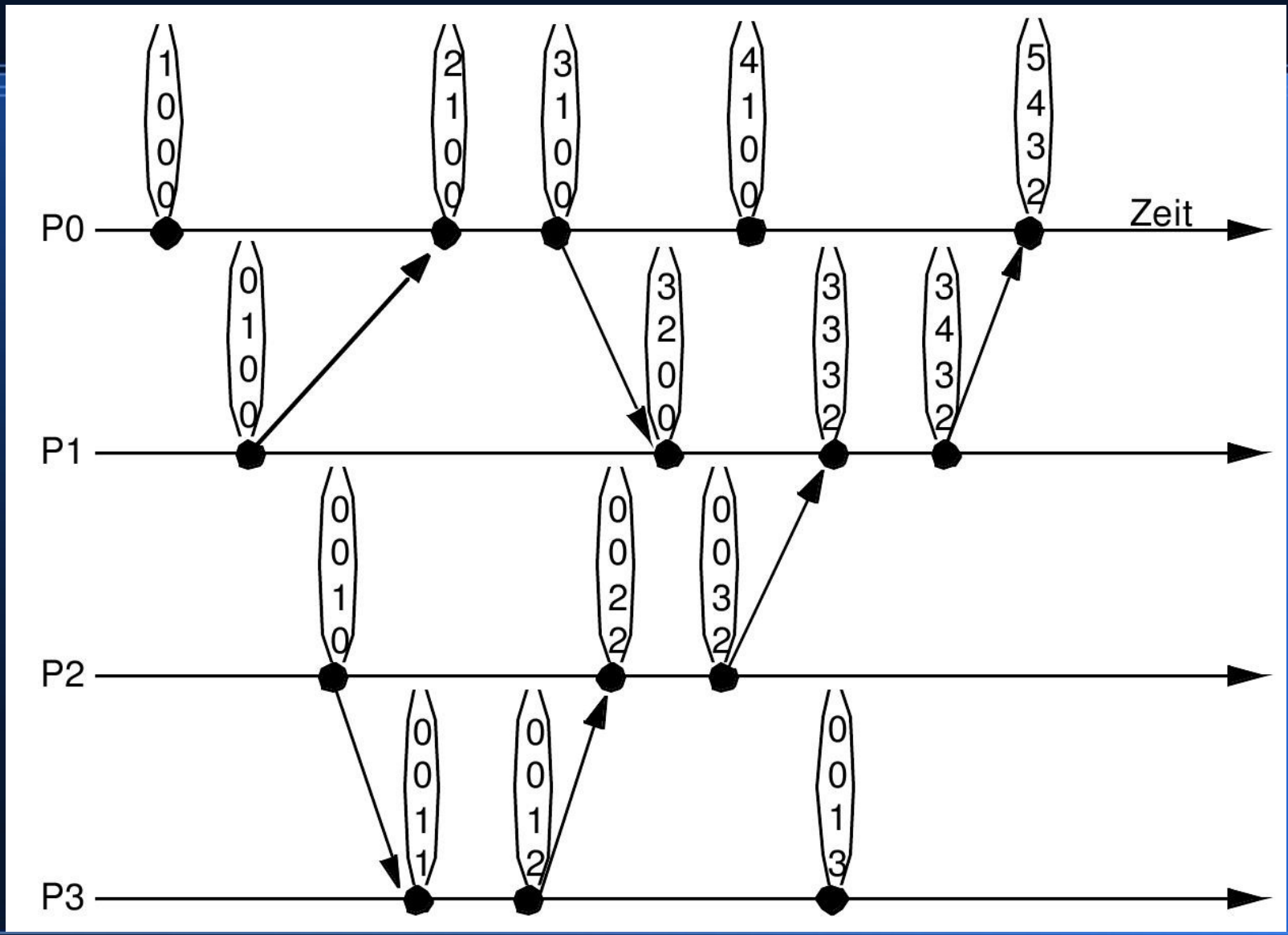
- Ereignisse e, e' kausal abhängig e -> e' => t(e) < t(e')
- Gilt Äquivalenz? D.h. t(e) < t(e') => e->e'



Verbesserung der Lamport-Zeit

- Aus t(e)<t(e') schließen, dass e<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<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) <= v(j)
- => alle Vektoren auf einem Pfad erfüllen „<“

Vektorzeit: kausale Abhängigkeit (2)

Falls v(i)<=v(j), e mit Zeit v(i), e' mit Zeit v(j)

dann e->e'

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

Aussagen, Prädikate

- 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 zu einem Zeitpunkt der verteilten Berechnung gelten und von da an immer

- Beispiele:
  - Terminierung
  - Objekt ist Garbage
  - Deadlock

Stabile Prädikate (2)

Formalisierung

monotone Funktion
f: Zustand  M

M sei eine Menge mit Ordnungsrelation

Zustand einer Berechnung =
- gesamte Berechnungshistorie
- alle Nachrichten

Stabile Prädikate (3)

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

  Zustand z, Folgezustand z'

  f(z) <= f(z')

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

Definitionen *Safety* und L*iveness*

- *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

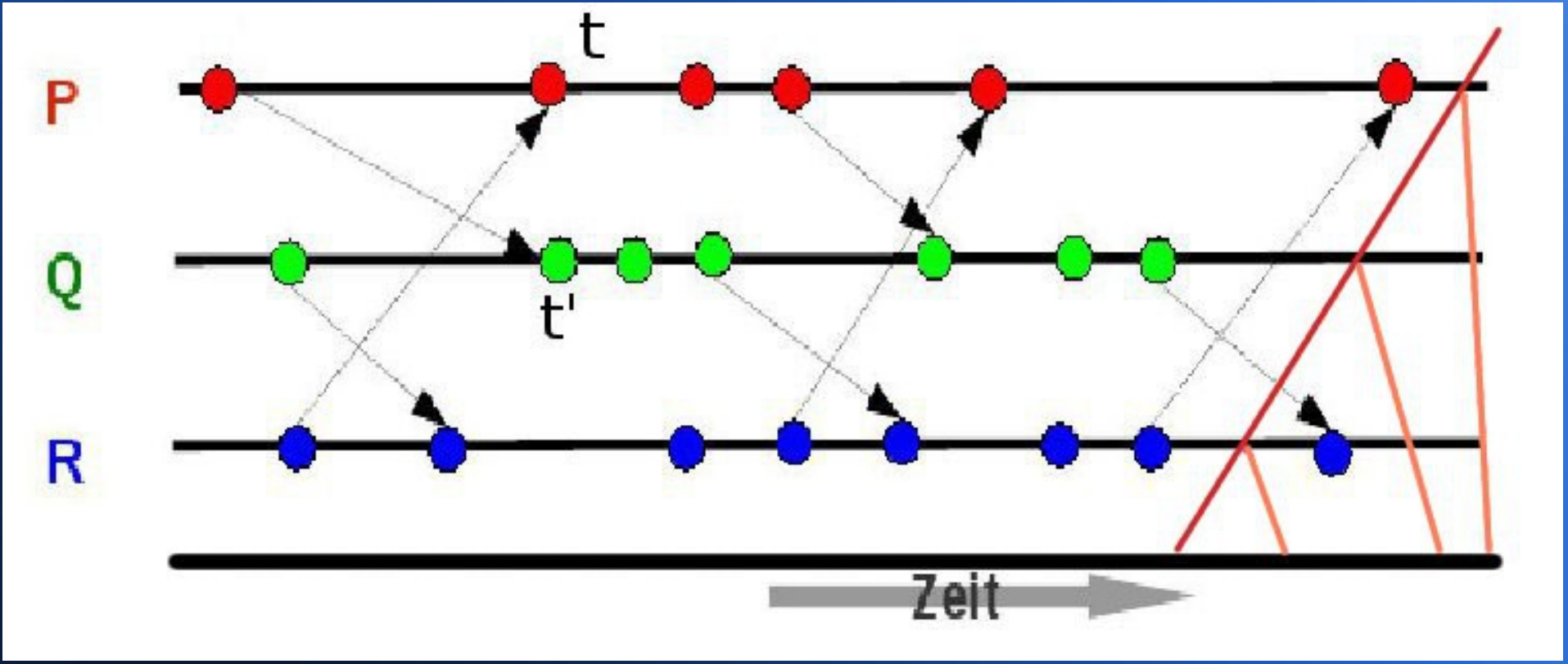
- *Safety* = something bad will never happen
  - die Bilanz von Bank X ist immer > 0
  - ein Deadlock kann nicht auftreten
- *Liveness* = something good will eventually happen
  - Algorithmus tritt in einen guten Endzustand ein
  - Algorithmus terminiert
  - Terminierung wird entdeckt

Formale Definitionen, Beispiele

- Kontrollalgorithmus zur Feststellung der Terminierung
  - Meldet immer „vielleicht“
- ➔ der Kontrollalgorithmus ist safe: sagt nichts Falsches

- Kontrollalgorithmus zur Feststellung der Terminierung
  - Meldet immer „ja“
- ➔ Der Kontrollalgorithmus ist live: schließlich richtig

Terminierung



Ideen ? Ziel: Beweis, dass keine Nachrichten unterwegs