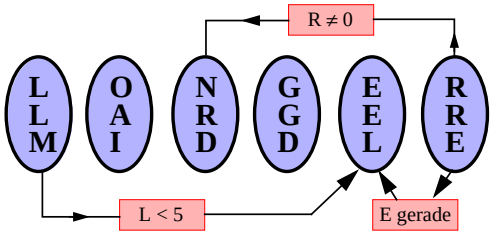
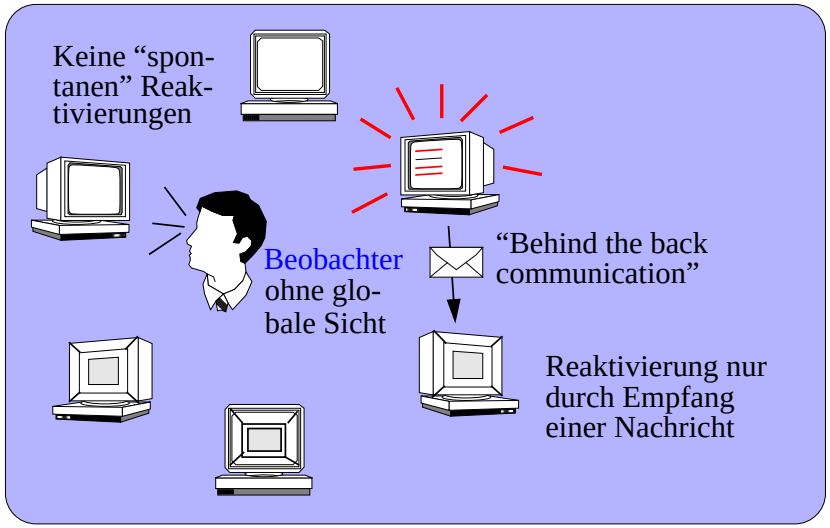
Das Problem der Terminierung

- Bsp: Zahlenrätsel (oder ggT) auf einem PC-Cluster



- pro Spalte (bzw. "Philosoph") jeweils ein Display
- dort jeweiligen Zustand und neue Wertemengen anzeigen

aktiv oder *passiv*

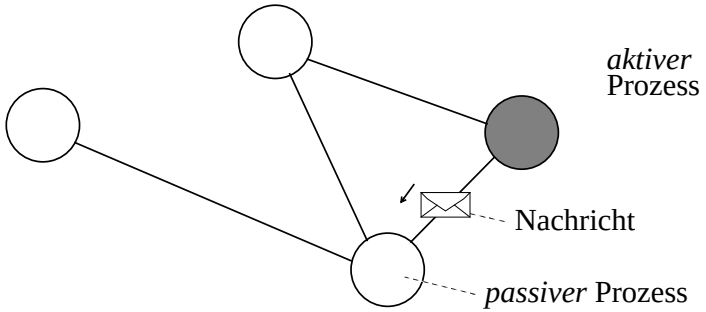


Erkennung der verteilten Terminierung?

alle passiv und keine Nachricht unterwegs

Terminierungserkennung

- Dem Zahlenrätsel- und dem ggT-Beispiel gemeinsam ist ein etwas abstrakteres Modell:

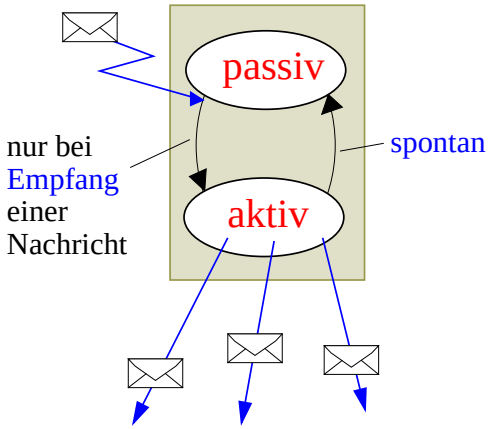


- Es gibt aktive und passive Prozesse sowie Nachrichten
  - dabei kann eine ankommende Nachricht einen passiven Prozess wieder reaktivieren
- Man möchte "irgendwie" erkennen, ob alle Prozesse passiv sind und keine Nachricht mehr unterwegs ist
  - Stagnationszustand beim verteilten Approximationsparadigma

Verteilte Terminierung: Modell und Problemdefinition

Nachrichtengesteuertes Modell einer vert. Berechnung:

- Prozesse sind *aktiv* oder *passiv*
- Nur aktive Prozesse versenden Nachrichten
- Prozess kann "spontan" passiv werden
- Prozess wird durch ankommende Nachricht reaktiviert



Problem:

Feststellen, ob (zu *einem* Zeitpunkt)
- alle Prozesse passiv sind
- keine Nachricht unterwegs ist

- "globales Prädikat"
- "stabiler Zustand"

Die Aktionen der Basisberechnung im nachrichtengesteuerten Modell

Prozess p:

"guard": Prädikat über dem lokalen Zustand

S_p: {Zustand = *aktiv*}
send message <M> **to** ...

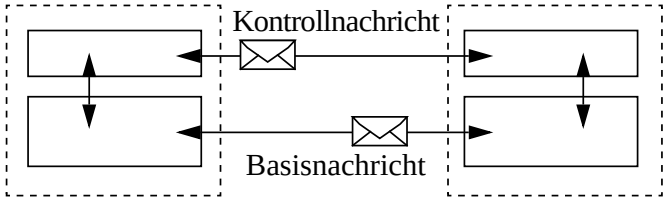
R_p: {Eine Nachricht ist angekommen}
receive <M> ; Zustand := *aktiv*

I_p: {Zustand = aktiv}
Zustand := *passiv*

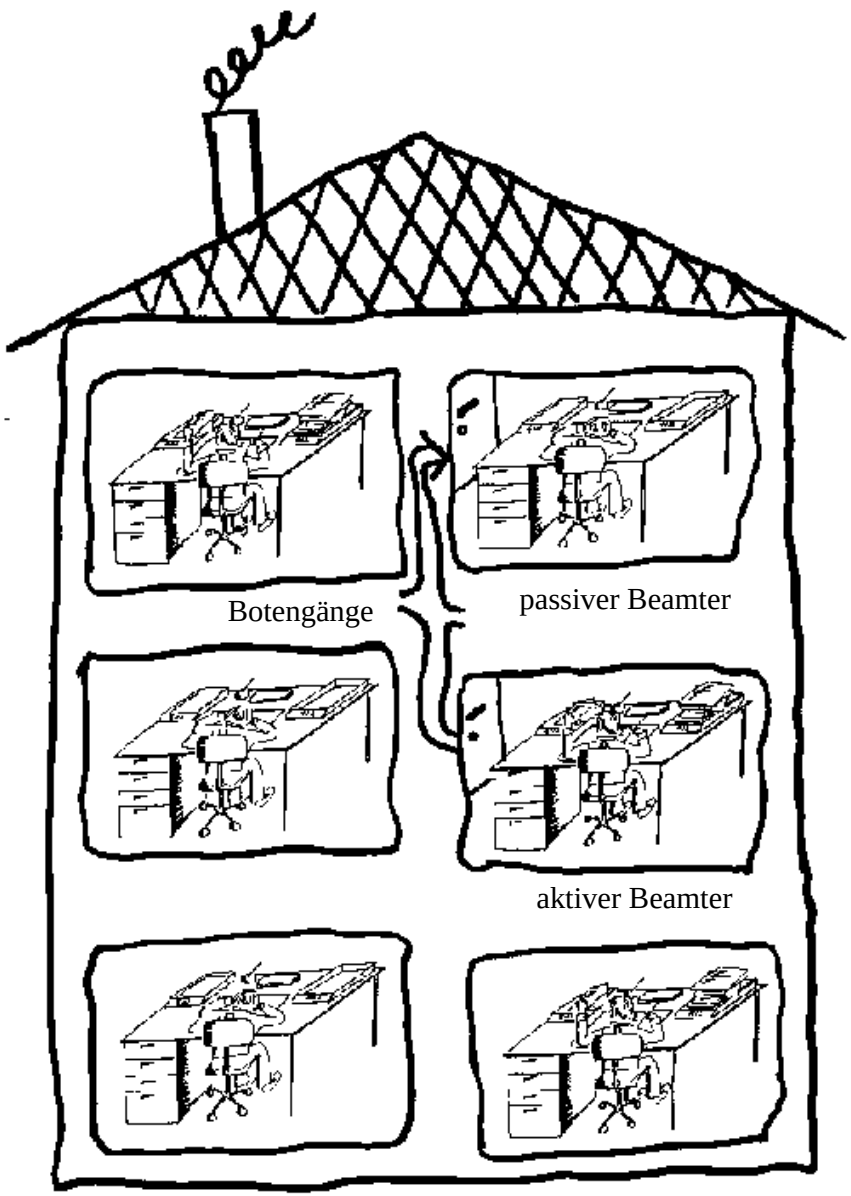
Typischerweise werden die Aktionen als "atomar" betrachtet

Abstraktes Verhalten einer verteilten Berechnung hinsichtlich Terminierung (ggf. existieren weitere Aktionen)

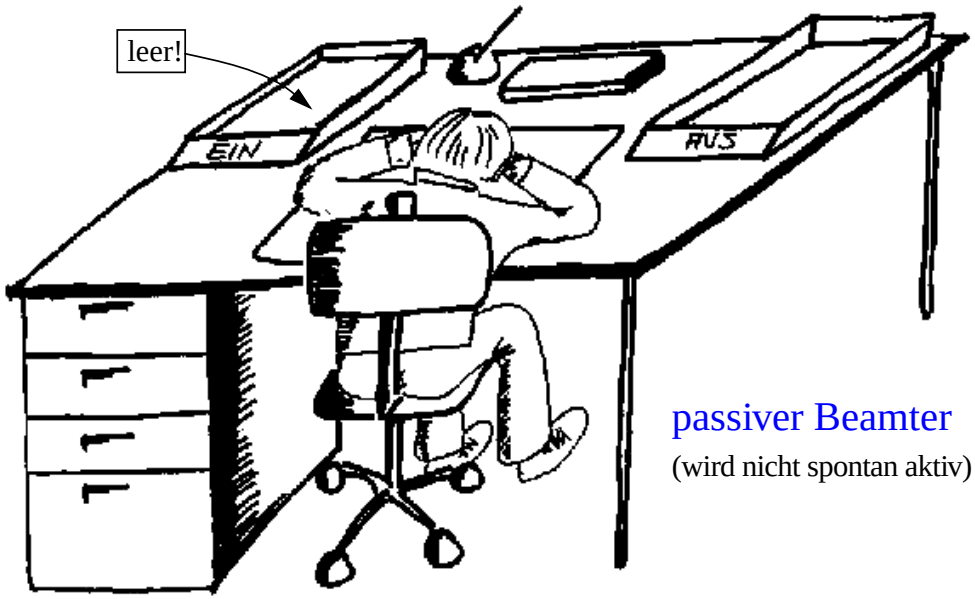
- Durch einen "überlagerten" Kontrollalgorithmus werden weitere Aktionen hinzugefügt
- "Anreichern" der Semantik der Basisberechnung für Zwecke des Kontrollalgorithmus
  - z.B. Verändern spezifischer (lokaler) Variablen
- Überlagerter Algorithmus soll Basisberechnung nicht stören
  - darf aber die Variablen, die der lokalen Kommunikation mit dem Basisalgorithmus dienen, lesen und schreiben



Eine typische Behörde



aktiver Beamter



passiver Beamter
(wird nicht spontan aktiv)

Die Funktionsweise der Behörde

(1) Publikumsverkehr nur bis 12 Uhr
(2) Schliesst erst, wenn alle Vorgänge bearbeitet
(3) Vorgänge werden von Beamten erledigt
(4) Die Bearbeitung eines Vorganges *kann* neue Vorgänge für andere Beamte auslösen
(5) Aktenaustausch per (bel. langsame) Boten
(6) Keiner hat den Gesamtüberblick
(7) Beamte sind *aktiv* oder *passiv*
(8) Ein Beamter wird nicht spontan aktiv

Terminiert, wenn alle passiv und nichts "unterwegs"

Das ist ein stabiler Zustand!

Variante: Beamter lässt sich während der Arbeit nicht stören (anklopfen/warten auf "herein") -->

! → Beamte *scheinen* immer passiv (Atommodell)

Speedup ist durch die Maximalzahl gleichzeitig aktiver Beamten begrenzt

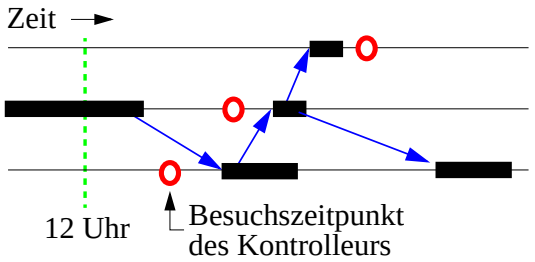
Dieser ist oft erstaunlich niedrig...

Das schiefe Bild des Kontrolleurs

- *Kontrolleur* wandert durch die Behörde, um die Terminierung feststellen zu können

- *Problem*: Wie stellt der Kontrolleur fest, ob der stabile Terminierungszustand eingetreten ist?

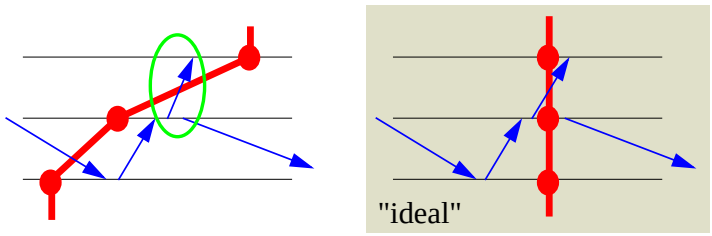
- Die *Illusion* Kontrolleurs:



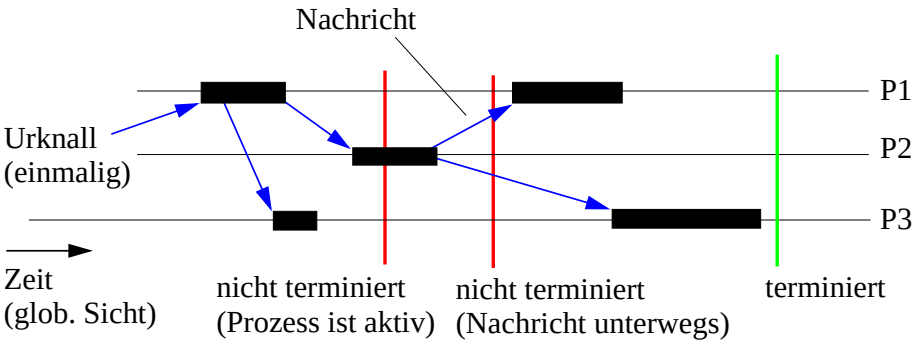
"behind the back communication"

- Alle Beamten stets passiv
- Σ Nachrichten versendet = Σ Nachrichten empfangen

- Kontrolleur macht sich ein *schiefes Bild!*

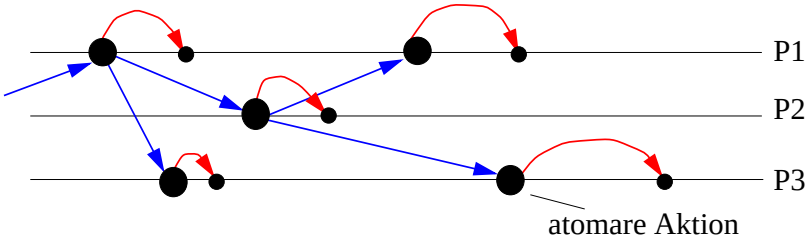


Zeitdiagramme und Atommodell



Idee: Dauer der Aktivitätsphasen "gegen Null" gehen lassen

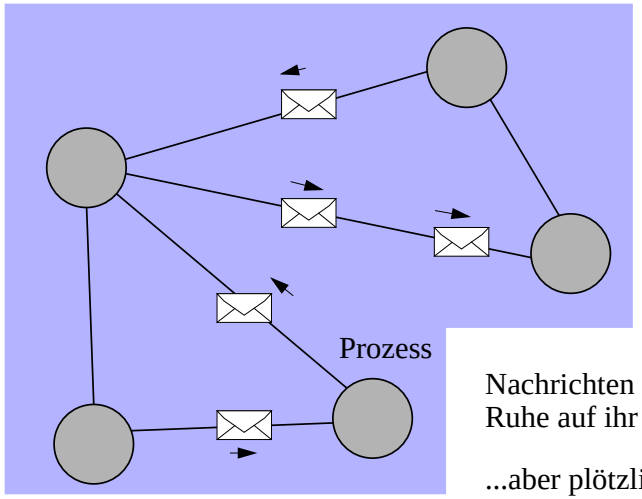
Modellierung: Prozess sendet (virtuelle) Nachricht an sich selbst, sobald er aktiv wird; ist "unterwegs", solange er aktiv ist



Terminiert (Atommodell) <==>
Keine (echte oder virtuelle) Nachricht unterwegs

Zur Lösung des Terminierungsproblems also feststellen, *ob* noch Nachrichten unterwegs sind

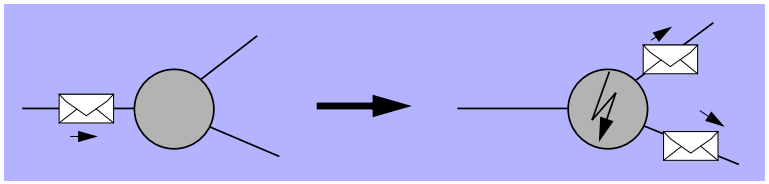
Globale Sicht ’atomarer’ Berechnungen



idealisierter Beobachter

Nachrichten fliessen in aller Ruhe auf ihr Ziel zu...

...aber plötzlich "explodiert" ein Prozess, wenn er von einer Nachricht getroffen wird!



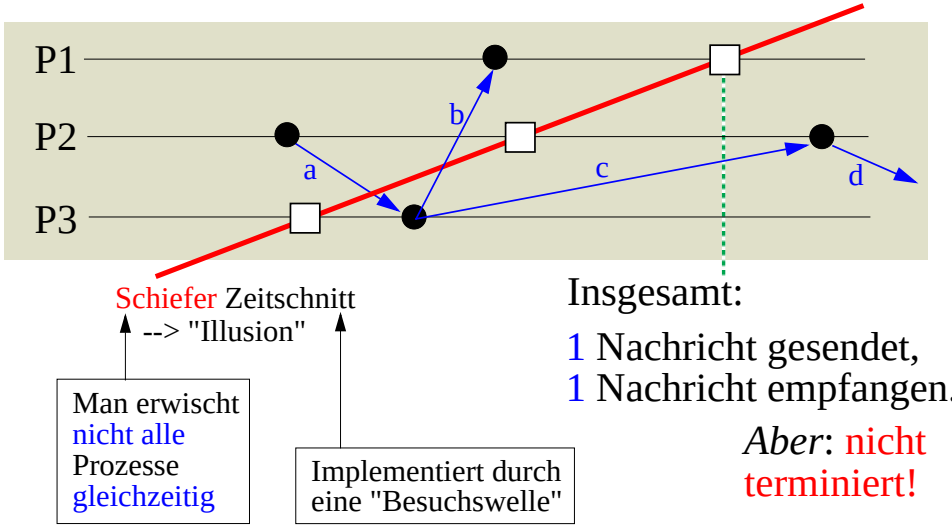
Terminiert, wenn in der globalen Sicht kein ✉ existiert

- Statt im "passiv/aktiv-Modell" genügt es offenbar, im Atommodell die Terminierungserkennung zu lösen (wieso?)
- Wie sehen die Aktionen der Basisberechnung in diesem Modell aus?

Verteilte Terminierung: Lösungen durch Zählen von Nachrichten?

- Genügt das (verteilte) Zählen von gesendeten und empfangenen Nachrichten?

- Einfaches Zählen genügt nicht, Gegenbeispiel:



Insgesamt:

1 Nachricht gesendet,
1 Nachricht empfangen.

Aber: nicht terminiert!

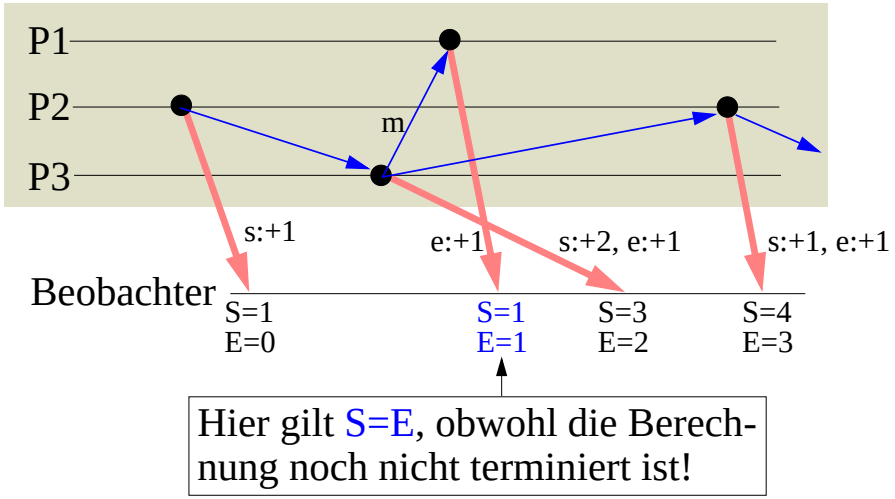
Ursache (informell):

- Nachricht aus der "Zukunft"
  - kompensiert die Zähler
- Inkonsistenter Schnitt
  - ist nicht äquivalent zu einem senkrechten Schnitt

Lösung durch "Ursachenvermeidung"? Ideen vielleicht:

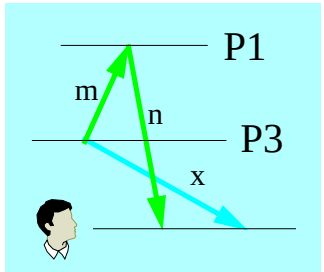
- Nachrichten aus der Zukunft *vermeiden* oder zumindest *erkennen*?
- Senkrechten Schnitt simulieren durch *Einfrieren* der Prozesse?

Beobachter über gesendete und empfangene Nachrichten informieren?



- Gleiches Szenario wie eben: Beobachter erfährt, dass m *empfangen* wurde, aber nicht, dass m *gesendet* wurde!

- Man beachte auch, dass hier eine Nachricht (x) in indirekter Weise (via m und n) "überholt" wurde!



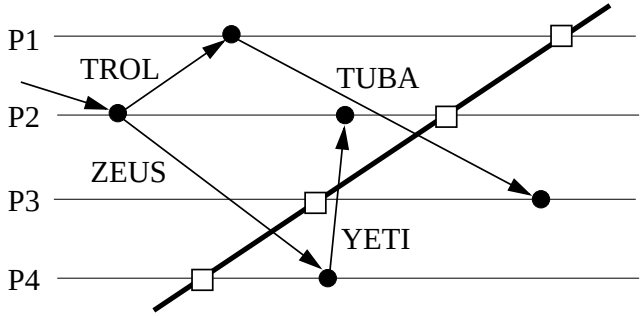
Vermutung: Wenn Informationsnachrichten nicht (indirekt) überholt werden können, dann kann das Phänomen eines "schiefen Bildes" nicht auftreten!

- worauf gründet sich die Vermutung?
- kann man solchermassen korrekte ("kausaltreue") Beobachtungen erzwingen?

Nachrichten eindeutig benennen?

Prinzip: Jede Nachricht bekommt einen (global) eindeutigen Namen:

- TROL, ZEUS, TUBA, YETI,... (?)
- Nachricht kennt ihren Namen
- Sender weiss, welche Nachrichten gesendet wurden
- Empfänger weiss, welche Nachrichten empfangen wurden



ZEUS und TUBA müssen Schnitt von links nach rechts überquert haben, YETI von rechts nach links (relativ zum Schnitt!)

- Welle akkumuliert Namen der gesendeten und Namen der empfangenen Nachrichten
- Wenn eine gesendete nicht empfangen wurde, muss sie den Schnitt überquert haben ==> Terminierung nicht melden
- Terminiert, wenn alle "bekanntermassen gesendeten" auch empfangen wurden? (Beweis?)
  - Tip: Wenn keine Nachricht den Schnitt (von links nach rechts??) überquert, ist der Lebensfaden des Systems gerissen; rechts des Schnittes kann dann keine Aktivität mehr entfacht werden (wieso?)

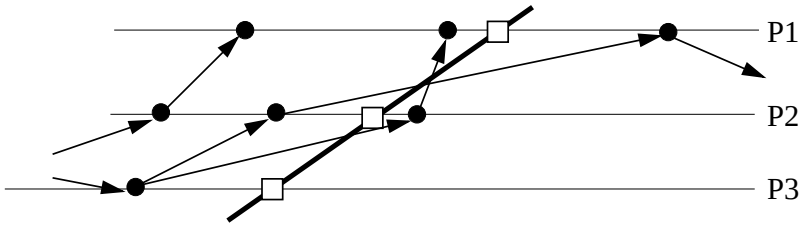
Frage: Wie geht das ganze überhaupt initial los?

Eindeutige Nachrichtennamen?

- Sender könnte Nachrichten fortlaufend numerieren und seinen eigenen eindeutigen Namen hinzufügen
  - lässt sich einfacher verwalten als beliebige (global eindeutige) Namen
- Es genügt wohl auch eine fortlaufende Numerierung pro Sender-Empfänger-Beziehung ("Kanal")
  - z.B. 17.4.239 ("239. Nachricht von Knoten 17 an Knoten 4")
  - Verwaltungsaufwand ist recht hoch (bei FIFO benötigt man keine Mengen, es genügen $O(n^2)$ Zähler)

Genügt pauschales Zählen pro Kanal?

(anstatt Nachrichten pro Kanal individuell zu betrachten)

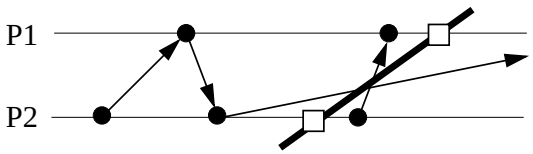


- Welle stellt folgendes fest:
  - auf Kanal P2P1 sind 2 Nachrichten gesendet und 2 Nachrichten empfangen worden
  - dennoch überquerte eine Nachricht den Schnitt von P2 nach P1!
- Denkübung: Wäre das bei FIFO-Kanälen korrekt?
  - d.h. wäre dann bei ausgeglichenen Kanalzählern keine Nachricht auf diesem Kanal unterwegs?

Behauptung (auch bei non-FIFO!):

Wenn entlang eines Schnittes *alle* Kanalzähler bzgl. send/receive ausgeglichen sind, dann überquert keine Nachricht den Schnitt

- Wieso? (intuitives Argument?)
- Beweis?



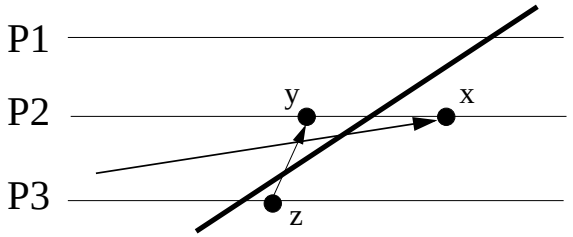
wieso ist das eigentlich kein Gegenbeispiel?

Beweisskizze für das Kanalzählerkriterium

Behauptung: Wenn entlang eines Schnittes pro Kanal gleich viele Nachrichten gesendet wie empfangen wurden, dann ist die Berechnung terminiert

Betrachte frühestes Ereignis (x) *nach* dem Schnitt:

Bei globaler (von links nach rechts fliessender) *Zeit* in der Abb. ist dies klar; wenn man ohne solche graphischen Veranschaulichungen auskommen will, muss man statt dessen die *Kausalrelation* bemühen!



Wir zeigen durch *Widerspruch*: es gibt kein frühestes Ereignis nach dem Schnitt ==> Terminierung

- Dies ist ein Ereignis mit Empfang einer Nachricht, deren Sendeereignis *links* des Schnittes liegt
- Zugehöriger Kanalzähler kann nicht getäuscht werden, da für eine *Kompensationsnachricht* gilt: Empfangen (y) vor dem Schnitt, gesendet (z) danach
- Sendeereignis der Kompensationsnachricht wäre *früheres* Ereignis *nach* dem Schnitt ==> *Widerspruch*
  - Senden ist immer früher als das Empfangen einer Nachricht!
  - z früher als y, y früher als x ==> z früher als x

Zählen pro Kanal ist aber etwas aufwendig ($O(n^2)$ Zähler); geht es nicht doch mit "ganz pauschalen" Zählern?