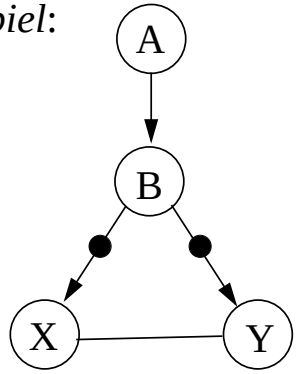
Verbesserung des Echo-Algorithmus?

[Helary et. al.]

- *Idee*: vermeide Besuch von Knoten, von denen man weiss, dass sie von anderen Explorern besucht werden

Beispiel:



Nachricht von B an X enthält Information, dass Y nicht besucht zu werden braucht

- *Voraussetzung*: Identitäten der Nachbarn bekannt

Schema (Modifikation gegenüber PIF-Echo):

Menge von "Tabu-Knoten"

```
receive <..., z>
...
y := neighbors \ z
send <..., z ∪ y> to all y
(* Registrieren, über welche Kanäle
Echos oder Explorer eintreffen müssen *)
```

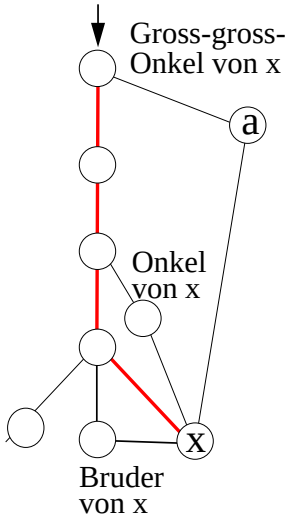
Statt neighbors ohne Vorgänger im Original

- Initiator i: send <..., neighbors ∪ {i} >

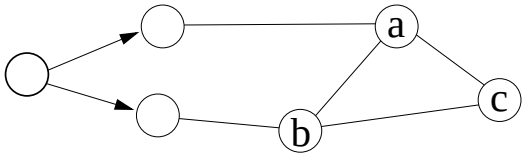
Was wird gespart?

- Keine Nachricht an einen Vorgänger oder einen direkten Nachbarn eines Vorgängers (z.B. "Brüder" und "Onkel")

- Allerdings: Obwohl x nicht an a sendet, sendet a an x! Über diese Kante fliesst dann auch ein Echo zurück --> nichts gespart, da 2 Nachrichten über die Kante!



- Auch in diesem Fall spart man nicht (immer?) etwas:



Knoten a und b werden "gleichzeitig" erreicht: wissen nichts voneinander: --> senden beide gegenseitig und an c

- Wieviel wird bei vollständigen Graphen gespart? Und bei Bäumen? Spielt der "Vermaschungsgrad" eine Rolle?

- Ersparnis nicht ganz klar
  - interessante Extremfälle, z.B. Baum oder vollständiger Graph
- Ersparnis an Nachrichten durch Nachteile erkauft
  - lange Nachrichten (O(n))
  - Nachbaridentitäten müssen bekannt sein

Das Märchen von der verteilten Terminierung

[F. Mattern - Informatik-Spektrum 8:6, pp. 342-343, 1985]

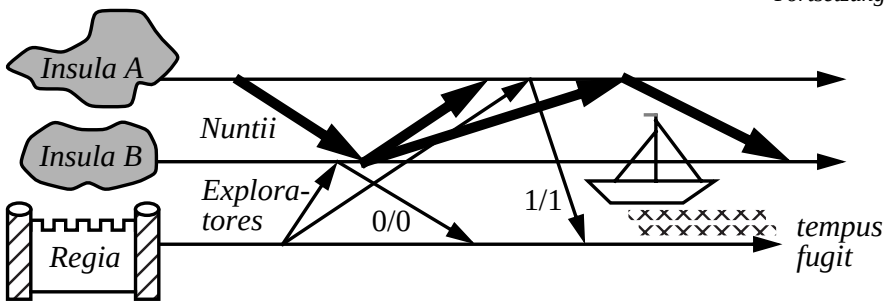
Als einst der König von Polymikronien merkte, dass die Zeit gekommen war, sein aus unzähligen Inseln bestehendes Reich gerecht unter seinen Enkelkindern aufzuteilen, sandte er Botschaften an die weit über das Land verteilt lebenden Weisen aus, auf dass diese ihm einen klugen Vorschlag unterbreiten mögen.

Wohl wusste der König um den eigenwilligen aber steten Lebenswandel seiner Ratgeber, welche den ganzen Tag nichts taten, als zu essen und zu denken: War einer von ihnen bei Speis und Trank, so konnte alleine eine königliche Botschaft oder eine Nachricht eines anderen Weisen ihn zum Denken anregen - er griff meist sogleich zu Feder, Papier, Tinte und Siegel, um einigen übrigen Mitgliedern des so weit verteilten königlichen Konsiliums eine neue Weisheit zuzusenden. Hungrig vom Denken wandt er sich alsbald wieder der stets fürstlich gedeckten Tafel zu.

Als indes die Jahre vergingen und der König immer älter wurde, ohne dass er von den Weisen einen Rat erhalten hätte, seufzte er, schickte nach seinem geheimen Hofrat und sprach zu ihm: "Ich weiss wohl, wie schwierig ein gerechter Plan zur Aufteilung meines Erbes ist, und ich kenne die Regeln meines Konsiliums, wonach man mir erst kundtut, wenn die königliche Sache so erschöpfend beraten wurde, dass ein jeder der Weisen zufrieden ist und keine Botschaft mehr unterwegs ist. Alleine die königliche Post bereitet mir Sorge - ist etwa ein Schiff in den Stürmen der Meere gesunken oder hat sich ein Bote in der Weite des Reiches verirrt?"

"O königliche Hoheit", entgegnete der geheime Hofrat und sprach weiter: "Unermesslich gross ist Euer Reich, und gar lange brauchen die Segler, um von einem Eiland zu einem anderen zu gelangen. Rein niemand vermag die Zeit vorher abzuschätzen, und es wird sogar berichtet, dass in der ein oder anderen Nacht ein Postboot ein anderes überholt. Aber die Segelkünste der Seefahrer, die hohe Schule der Schiffsbaumeister und die pflichtbewusste Ergebenheit Eurer Diener sorgen dafür, dass nicht eine einzige Botschaft verloren gehen kann. Lasset uns also Kundschafter aussenden, mein König, um jeden Ratgeber zu befragen, wieviele Botschaften er empfangen und versandt hat. So können wir leicht Gewissheit darüber erlangen, ob summa summarum so viele Nachrichten versandt wie empfangen wurden und das Konsilium des Königs Sache abschliessend beraten hat."

Fortsetzung--->



Übungen (2) zur Vorlesung "Verteilte Algorithmen"...

a) Man beweise die Korrektheit des im Märchen beschriebenen "Doppelzählverfahrens" zur Feststellung der verteilten Terminierung.

b) Man beweise die Korrektheit des Echo-Algorithmus:
- der Initiator terminiert erst, wenn alle Knoten informiert wurden ("safety"),
- nach endlicher Zeit terminiert der Initiator ("liveness").

Überlegen Sie sich, was für Beweistechniken Sie einsetzen können (Invarianten, Induktion...) und wie genau / formal die Spezifikation des Algorithmus sein sollte, damit Sie im Beweis formal argumentieren können. Geben Sie ggf. eine formalere Spezifikation des Algorithmus an.

---> Fortsetzung Märchen

Der König war hoch erfreut über diese weisen Worte, schöpfte neue Zuversicht und beauftragte sogleich den Hofmathematicus, einen Plan auszuarbeiten. Dieser erschien alsbald mit einer grossen Leinwand und sprach: "Majestät, auf diesem Szenario seht Ihr, dass des Hofrats Plan versagen kann - die zu den Eilanden A und B gesandten Kundschafter berichten, dass so viele Botschaften empfangen wie versandt wurden - summa summarum nur eine Botschaft. Nichtsdestotrotz sind noch Nachrichten unterwegs. Die Sache ist wohl so, dass die Kundschafter sehr klug vorgehen müssen, um sich nicht täuschen zu lassen und des Königs Zählung zu verfälschen, alldieweil wir primo die Uhr noch nicht erfinden konnten und somit auch keine einheitliche Reichszeit haben, secundo wir den Rundfunk noch nicht kennen und tertio die ehrwürdigen Sitten es verbieten, dass die Weisen an einem gemeinsamen Ort zusammenkommen." Der König war erstaunt über die gar wundersamen Worte und meinte: "Nun, das weiss ich wohl, denn ich bin der König. Was also rät Er mir ?" "Hoheit, mein Plan sieht vor, die Kundschafter erneut auszusenden, sobald der letzte bei Hofe eingetroffen ist. Wird alsodann das Ergebnis genau bestätigt und sind die Summen gleich, so ist keine Nachricht mehr unterwegs." "Vortrefflich", meinte der König, der nichts verstanden hatte. "Kümmere Er sich nur sogleich um die Instruktion der Kundschafter!"

Der Hofmathematicus tat wie befohlen, verbesserte seinen Plan noch verschiedentlich, und bald waren die Kundschafter mit allen königlichen Vollmachten versehen auf den besten Seglern des Reiches unterwegs zu den Weisen.

Als endlich der Schluss der Weisen bei Hofe eintraf, war der König so voll Glück, dass er den Hofmathematicus bald darauf zu seinem ersten Hofinformaticus ernannte. Und dieser forscht, wenn er nicht gestorben ist, noch heute in einem stillen Turm des königlichen Palastes... an einer noch besseren Lösung zum Problem der verteilten Terminierung!

Papier war kostbar - so verzichtete der Hofinformaticus leider darauf, einen Korrektheitsbeweis seines Planes niederzuschreiben. Einer Randbemerkung seiner Schriften entnehmen wir, dass er später ein Verfahren ersann, bei dem nicht in jedem Fall die Inseln zwei- oder mehrfach von den Kundschaftern aufgesucht werden müssen. Desweitern gelang es ihm offenbar, die Zahl der notwendigen Kundschafterfahrten durch eine einfache Funktion der Zahl der Inseln und Botschaften zu beschränken. Leider reichte wiedereinmal der Rand zur Darstellung der Methode nicht aus. Wer hilft mit bei der Rekonstruktion und Verifikation der Verfahren ?

Zeitkomplexität

Beachte: Algorithmen i.a. nichtdeterministisch --> *mehrere* mögl. Berechnungen!

Variable Zeitkomplexität eines vert. Algorithmus = max. "Zeit" aller Berechnungen des Algo unter:

Z1: Lokale Berechnungen erfolgen in Nullzeit
Z2: Eine Nachricht benötigt *maximal* 1 Zeiteinheit

Einheitszeitkomplexität eines vert. Algorithmus = max. "Zeit" aller Berechnungen des Algo unter:

E1: Lokale Berechnungen erfolgen in Nullzeit
E2: Eine Nachricht benötigt *exakt* 1 Zeiteinheit

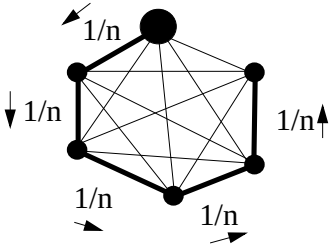
Behauptung:
Es gilt *nicht* immer variable Zeitkplx ≤ Einheitszeitkplx.

- Grund: Einheitszeitkplx erlaubt nicht alle Berechnungen!
- Frage: Gilt Umkehrung?

Bsp. Echo-Algorithmus auf vollständigem Graph

(1) Einheitszeitkomplexität = 3 ← Phase 1: Alle werden rot
Phase 2: Alle bis auf Initiator werden grün
Phase 3: Initiator wird grün

(2) Variable Zeitkomplexität ≥ n



- Explorer "aussen": 1/n Zeiteinheiten
- Jede sonstige Nachricht 1 Zeiteinheit
- Entarteter Baum Tiefe n-1 nach einer Zeiteinheit aufgebaut
- Echo beim Initiator nach n Einheiten

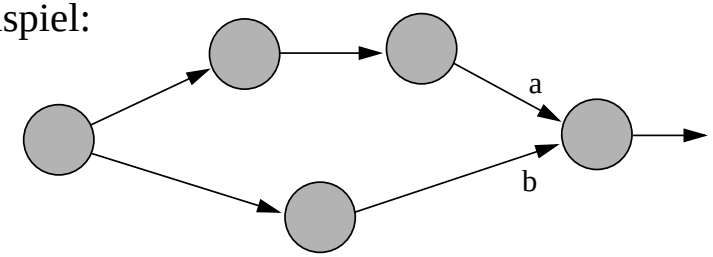
Zeitkomplexität: Welche Definition?

- *Einheitszeitkomplexität*: Einige Berechnungen bleiben unberücksichtigt!

Nicht bei var. Ztkplx! (Wieso?)

unwahrscheinliche?

Beispiel:



Trifft a vor b ein --> sehr lange Berechnung, sonst terminiert

mag vielleicht in 10% aller Fälle der Fall sein...

aber wie oft wirklich?

systemabhängig!

- *Variable Zeitkomplexität*: Resultat wird u.U. durch extrem unwahrscheinliche Berechnungen bestimmt

- Für worst-case: variable Ztkplx? Aber: average case?

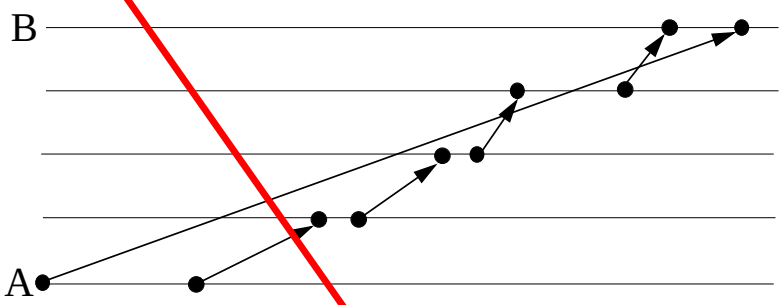
- Genauer: Wahrscheinlichkeitsverteilung --> Erwartungswert
  - systemabhängig
  - schwierig
  - jeden Tag anders...

Ein anderes Zeitkomplexitätsmass

Längste Nachrichtenkette einer Berechnung

Dann max. (oder Mittelwert?) über alle Berechnungen

Beispiel:



Sende erst direkt, dann indirekt an B

- Prozess A initiiert den Algorithmus
- Beendet, wenn B direkt oder indirekt von B hört
  (Also: Wenn B eine Nachricht empfängt)

- Einheitsztkplx. = 1
- Var. Ztkplx. = 1
  (worst case)
- Längste Kette = 4

grösser!

- Wie sinnvoll ist dieses Mass?
- Was ist das "richtige" Mass für die Zeitkomplexität?

Bem.: solche Ketten spielen im Sinne eines "critical path" bei Beschleunigungsuntersuchungen auf Parallelrechnern eine Rolle

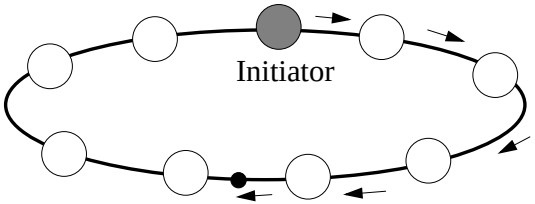
Broadcast auf speziellen Topologien

- Echo-Algorithmus realisiert einen Broadcast

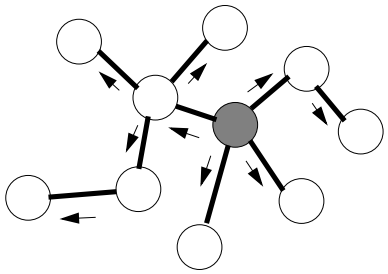
auf bel. zusammenh. Topologie

  - Verteilen von Information ausgehend von einem Initiator
  - für beliebige (zusammenhängende) Topologien
  - liefert sogar "Vollzugsmeldung" durch Echo-Nachrichten

- Auf *speziellen* Topologien lässt sich der Broadcast auch effizienter realisieren
  - Beispiel *Ring*: Ein "Token" zirkuliert mit der Information; alle sind informiert, wenn das Token wieder beim Initiator eingetroffen ist
  - ggf. kann einer anderen Topologie ein Ring überlagert werden



- Beispiel *(Spann)baum* (tatsächlich Unterschied zum Echo-Algorithmus?)



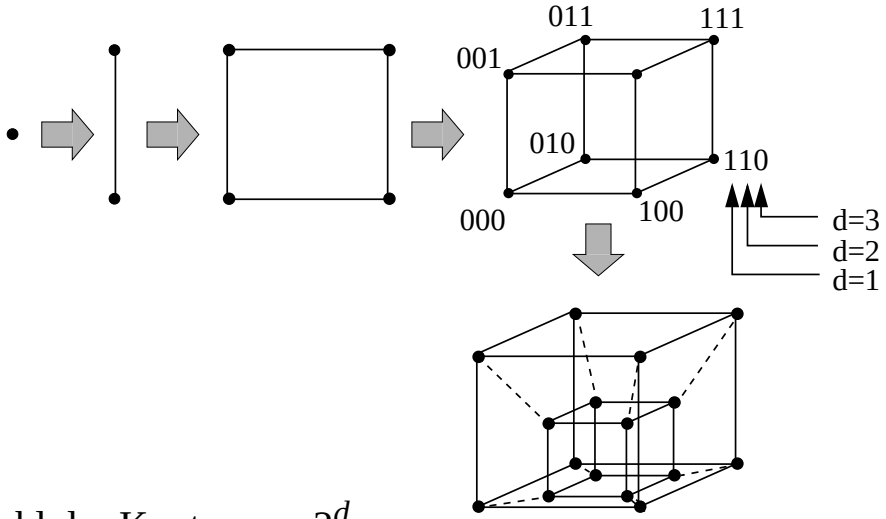
vorausgesetzt wird jeweils, dass der Algorithmus "weiss", dass eine spezifische Topologie vorliegt!

- Beispiel *vollständiger Graph* (als Denkübung)

Hypercubes

- Hypercube = "Würfel der Dimension d"
- Rekursives Konstruktionsprinzip
  - Hypercube der Dimension 0: Einzelrechner
  - Hypercube der Dimension d+1:

    „Nimm zwei Würfel der Dimension d und verbinde korrespondierende Ecken“



- Anzahl der Knoten $n = 2^d$
- Anzahl der Kanten = $d\ 2^{d-1}$ (Ordnung O(n log n))
  - viele Wegalternativen (Fehlertoleranz, Parallelität!)
  - maximale Weglänge: d = log n
  - mittlere Weglänge: d/2 (Beweis als Denkübung!)

wieviele verschiedene Wege der Länge k gibt es insgesamt?

- Knotengrad = d (nicht konstant bei Skalierung!)
- Einfaches Routing von einzelnen Nachrichten
  - xor von Absende- und Zieladresse...

Kombinatorische Aspekte (1)

- Wieviele verschiedene Wege der Länge k (ausgehend von Knoten 0) gibt es?
  - man wiederhole die Begriffe und Methoden der Kombinatorik...
  - für den Anfang: was ergibt sich für k=1, k=n?
- Ein ähnliches Problem: Wege in Manhatten...

Brian Hayes, American Scientist, January-February 1996

A Walk in Manhattan (Auszug)

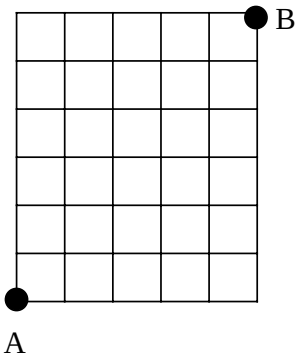
When I was a commuter in New York, I walked a diagonal route across midtown Manhattan morning and evening. I used to wonder how many different paths I could find through the grid of east-west and north-south streets without taking extra steps. Eventually I grew curious enough to calculate some solutions. For a square lattice of n-by-n blocks, any minimum-length diagonal path is necessarily 2n blocks long. How many such paths are there? For the 1-by-1 lattice the answer is 2: You can go east and then north or you can go north and then east. In a two-block square there are six paths, and in a 3-by-3 block there are 20. The sequence of values I calculated begins like this: 2, 6, 20, 70, 252, 924, 3432, 12870, 48620....

Do those numbers look familiar? They should. They are prominent members of one of the most ancient and famous families of numerical sequences. And yet I did not identify them until several years later, when I had a chance to submit them to Sloane's sequence server. The answer came back immediately: They were recognized as sequence M1645, the central binomial coefficients, the numbers that run down the middle of Pascal's triangle. My reaction was "of course! Why didn't I see it all along?" But I doubt that I would have made the connection without help.

Kombinatorische Aspekte (2)

The discovery was a productive one, which led not just to an answer but to an insight. I saw that counting the shortest diagonal paths for all rectangular lattices--not just the square ones--would fill in the rest of Pascal's triangle. Each row of the triangle consists of all the lattices with a given minimum diagonal path length. For example, the fifth row includes all the lattices with a path length of 4, namely the 0-by-4, 1-by-3, 2-by-2, 3-by-1 and 4-by-0 lattices. The corresponding counts of diagonal paths (and the corresponding entries in Pascal's triangle) are 1, 4, 6, 4 and 1.

http://www.sigmaxi.org/amsci/issues/comsci96/compsci96-01.html

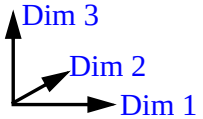


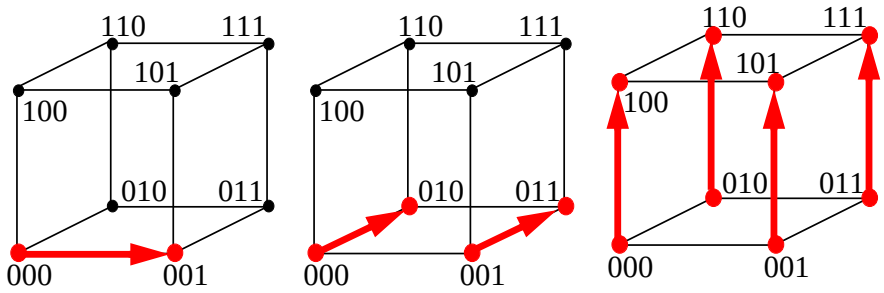
- Jeder kürzeste Weg von A nach B hat die Länge 11
- Repräsentation durch ein Wort der Länge 11
- 5 Nullen und 6 Einsen
- 0: gehe nach Osten
- 1: gehe nach Norden
- Wieviele solche Worte gibt es?

- Gegeben sei eine 11-elementige Menge
- Dem i-ten Element sei die Position i (in einem Wort) zugeordnet
- Anzahl der 6-elementigen Teilmengen = Anzahl der Worte der Länge 11 mit genau 5 Nullen und 6 Einsen
- ==> Binomialkoeffizienten

Broadcast in Hypercubes (1)

- Initiator habe die Nummer 00...00 (binär)
- Wir verzichten hier auf Vollzugsmeldung
  (also keine Acknowledgemnents oder Endeerkennung)





- Analog zum rekursiven Aufbau des Hypercube:
  - zunächst in Dimension 1 senden: Teil-Hypercube der Dimension 1 ist damit informiert
  - dann senden alle Knoten der Dimension 1 in Dimension 2
  - dann Dimension 3 etc.
- Nach d "Takten" sind alle Knoten informiert
  - Zeitkomplexität ist daher d (unter welchem Zeitmass?)
  - Nachrichtenkomplexität: $1 + 2 + 4 + ... + 2^{d-1} = 2^{d-1} - 1 + 2^{d-1} = n-1$
    (jeder Knoten, ausgenommen der Initiator, erhält genau eine Nachricht)
- Welche Komplexität hat ein *optimaler* Broadcast-Algo.?
- Geht es besser? ← was heisst überhaupt "besser"?
  - Algorithmus startet ziemlich "langsam": am Anfang geschieht wenig parallel!
  - Kann man dies durch *gleichzeitiges* Versenden "in alle Richtungen" beschleunigen?

Broadcast in Hypercubes (2)

- Ein anderes Verfahren (Vergleich als Denkübung!)
- Initiator sendet an alle seine Nachbarn:

  0...01, 0...010, 0...100, ... , 10...0

  in “kanonischer” Numerierung

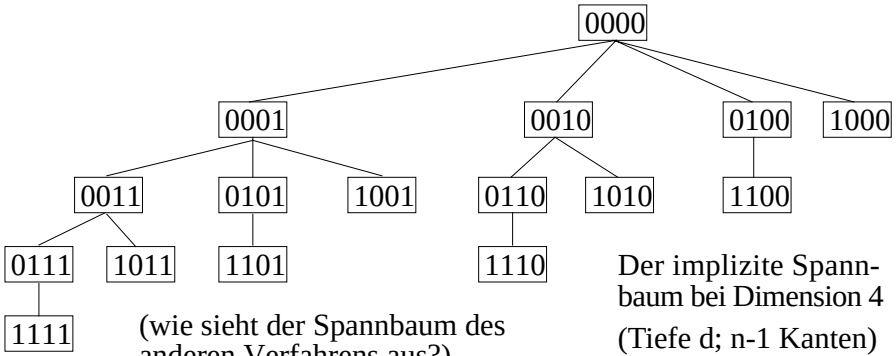
  am besten gleichzeitig, wenn dies technisch geht!

  linkeste 1 | beliebiges Restmuster

- Ein Knoten mit der Nummer 0...01x...y...z leitet die Information an alle seine “höheren” Nachbarn weiter:

  0...00**1**1x...y...z
  0...0**1**01x...y...z
  0...**1**001x...y...z
  ...
  10...001x...y...z

  Von welchem (eindeutigen) Knoten A wird Knoten B informiert?
  Setze *vorderste 1* von B auf 0
  --> = Nummer von A



(wie sieht der Spannbaum des anderen Verfahrens aus?)

Der implizite Spannbaum bei Dimension 4
(Tiefe d; n-1 Kanten)

- Der Algorithmus wird z.B. in Mehrprozessorsystemen (z.B. NCube) verwendet
- Wie effizient ist der Algorithmus? (Geht es besser?)
- Denkübung: Formuliere Algorithmus für einen beliebigen Initiator (schliesslich sind Hypercubes symmetrisch...)
- Denkübung: Vergleich mit Flooding bzw. Echo-Algorithmus

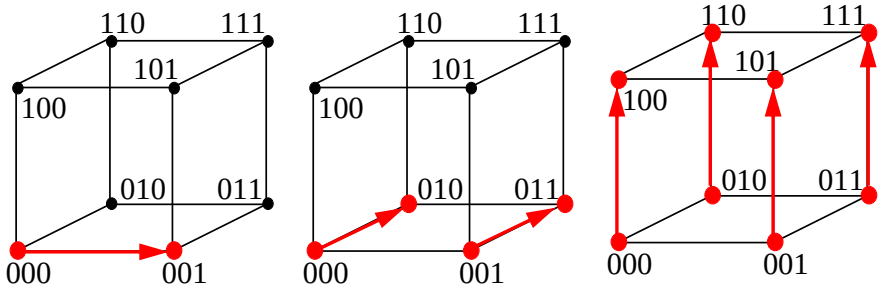
Noch ein anderer (besserer?) Algorithmus

- Beobachtungen:
  - Ein Baum verwendet im Hypercube relativ wenig Kanten --> schlechte Ausnutzung potentielle Parallelität
  - Es gibt *mehrere* Spannbäume in Hypercubes --> diese nutzen?

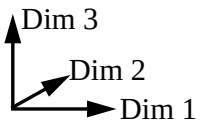
- Sender 0...0 teilt die Nachricht in d Pakete
- Sender startet für jedes Paket eine eigene “Welle”:
- *1. Paket* in *Dimension 1* senden --> 0...01
- Dann: Alle informierten Knoten (also 0...0 und 0...01) senden das Paket in Dimension 2
- Etc. Welle für Paket 1 breitet sich analog zur rekursiven Definition des Hypercubes in einer jeweils zusätzlichen Dimension aus
- Das *2. Paket* wird erst in *Dimension 2*, dann 3,..., d und erst zuletzt in Dimension 1 gesendet
- Das *3. Paket*: Dimensionsreihenfolge 3, 4, ..., d,1, 2
- Etc.: das *d.-Paket* in Dimensionsreihenfolge d, 1, 2,..., d-1

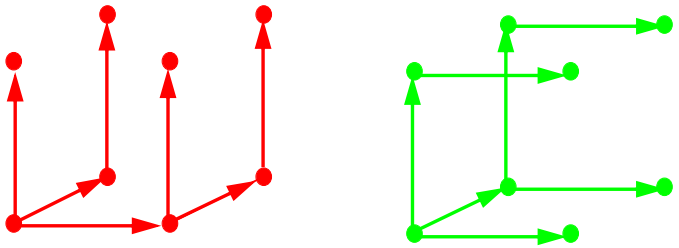
Denkübungen:
- Können so die Pakete gleichzeitig verschickt werden?
- Ist dann in jedem “Takt” pro Kante nur eine Nachricht unterwegs?
- Wieviele (kantendisjunkte ?) Spannbäume gibt es in einem Hypercube?

Veranschaulichung des Algorithmus



Die drei “Takte” der Welle von Paket 1

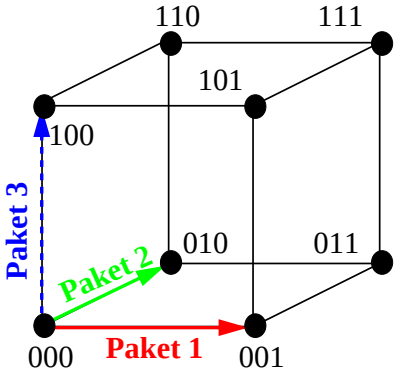




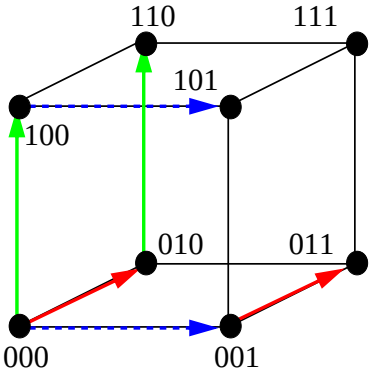
Die Spannbäume bzgl. Paket 1 und Paket 2

Parallelausführung der drei Wellen

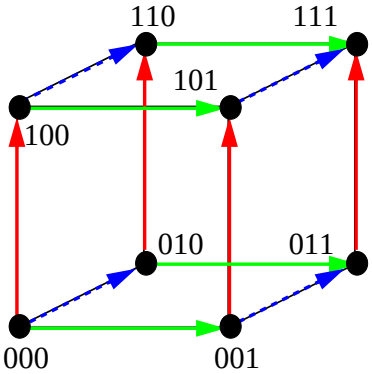
Takt 1



Takt 2



Takt 3



Takt
1 2 3

Paket 1: 1. 2. 3. Dim.

Paket 2: 2. 3. 1. Dim.

Paket 3: 3. 1. 2. Dim.

Pro Takt laufen die Pakete in jeweils unterschiedlicher Dimension!

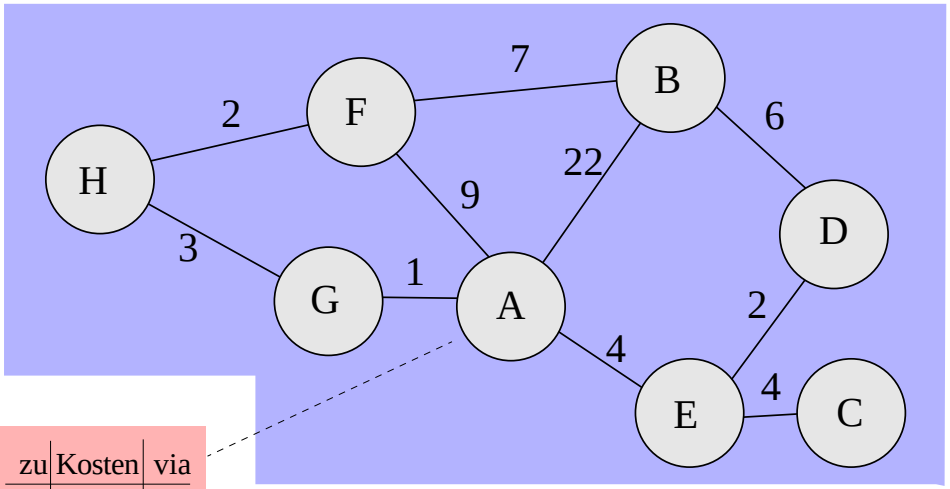
Es können also tatsächlich die drei Wellen parallel ausgeführt werden, ohne dass diese sich gegenseitig stören!

--> Dies ist das (im Prinzip) schnellere Verfahren!

Beachte: Ein globaler Takt ist gar nicht nötig!

Verteilte Berechnung von Routingtabellen für kürzeste Wege

Gegeben: ungerichteter zusammenhängender Graph mit bewerteten Kanten (Kosten, Länge...)



zu	Kosten	via
A	0	-
B	22	B
C	∞	?
D	∞	?
E	4	E
F	9	F
G	1	G
H	∞	?

Anfangs-tabelle für Knoten A

- Jeder kennt anfangs die Kosten zu seinen Nachbarn
- "Spontanstart": Sende eigene Tabelle an Nachbarn
- Bei Empfang einer Tabelle über Verbindung mit Kosten g:
  Für alle Zeilen i der Tabelle:
  Falls Nachricht.Kosten[i]+g < Knoten.Kosten[i]:
  ersetze Zeile (Kosten := Kosten+g; via := Absender)
- Falls sich Tabelle verändert hat:
  Neue Tabelle an alle Nachbarn (Ausnahme: Sender)
- Wie Terminierung feststellen?

- Ist eine verteilte Version des Bellman-Ford-Algorithmus
  - ähnlich dem bekannten Dijkstra-Algorithmus für kürzeste Wege
  - " Relexationsprinzip" (Bellman 1958, Dijkstra 1959, Ford 1962)

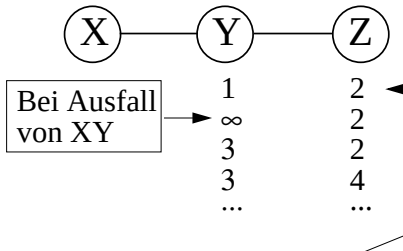
Kürzeste Wege in Rechnernetzen

- Algorithmus wird oft als dynamisches ("adaptives") Routing-Verfahren verwendet, wo in regelmässigen Zeitabständen die Tabellen neu berechnet und ausgetauscht werden
  - bzw. dann, wenn sich etwas ändert (Kosten einer Verbindung, z.B. Ausfall einer Leitung oder Änderung der Lastsituation)

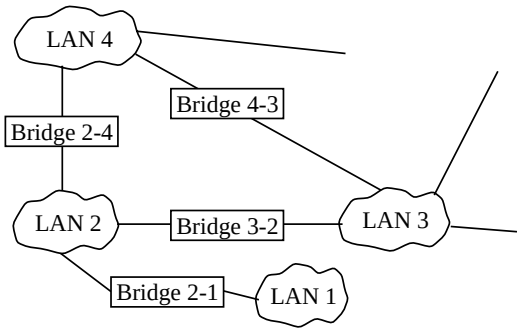
- Metrik für die Kosten z.B.:
  - (gewichtete)Anzahl der hops
  - Bitrate einer Verbindung
  - Verzögerung einer Verbindung (z.B. gemessen mit Testpaketen)
  - Länge der Warteschlange vor einer Verbindung

- "Count to infinity-Problem"

mehr zu diesen Dingen in anderen Vorlesungen ("Rechnernetze")

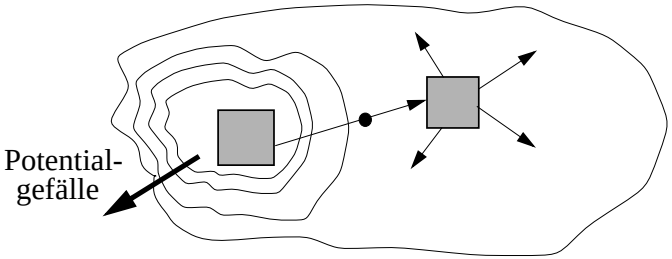


- Durch die kürzesten Wege zu einem festen Knoten ("Wurzel") ist ein kostenminimaler Baum gegeben
  - Algorithmus wird in LANs eingesetzt, um einen Spannbaum zu bestimmen (Knoten = Teil-LAN; Kante = Bridge)
  - Zyklenfreiheit ist wichtig, da kein Routing in LANs
  - Ende wird heuristisch durch Abwarten einer Zeitspanne festgestellt



Das Paradigma der vert. Approximation

Prinzip:
- Anfang: Informiere alle Nachbarn spontan
- Bei Empfang einer Nachricht:
  - berechne neue Approximation
  - falls diese "besser": informiere Nachbarn



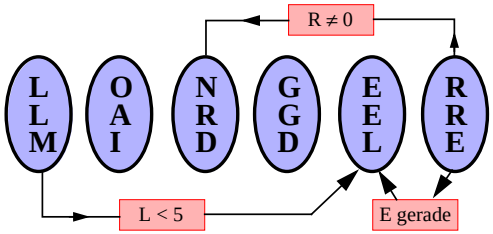
- *Nachrichtengesteuert* (aber "Spontanstart")
- Alle Prozesse arbeiten gleich, alle sind beteiligt
- *Nichtdeterministischer* Ablauf, determin. Ergebnis
- Beliebige stark zusammenhängende Topologie
- Assoziative Operatoren (min, max, $\cap$, $\cup$, +, and, or, ...)
- *Stagnation* bei globalem Gleichgewicht ("Optimum")
  --> Potentialunterschiede ausgeglichen
  --> Terminierungsproblem

Beispiele ("Instanzen der Algorithmenklasse"):
- ggT
- Zahlenrätsel
- Verteilen von Information ("Wissensausgleich")
- Routingmatrizen (inkl. Spannbaum)
- Maximale Identität ("election") } (noch)
- Lastausgleich (Approx. eines dyn. Optimums) } nicht be-
- Relaxationsverfahren (Lösen von DGL) } handelt

Verteilte Terminierung

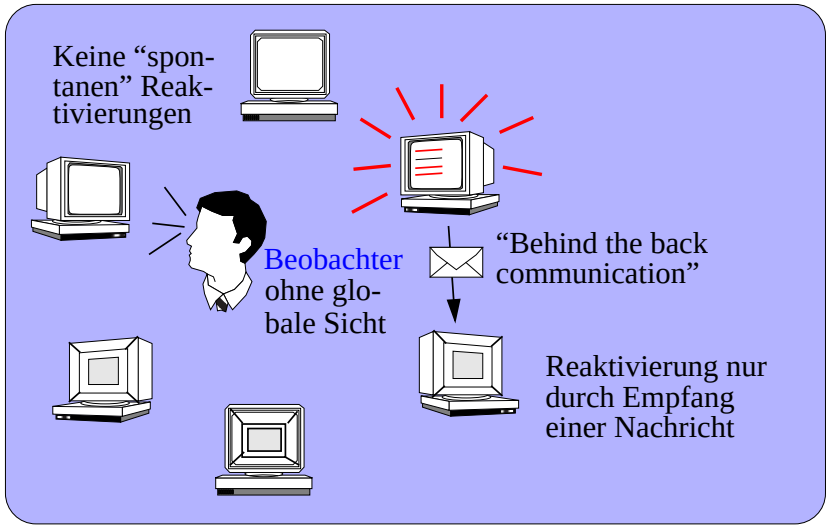
Das Problem der Terminierung

- 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*

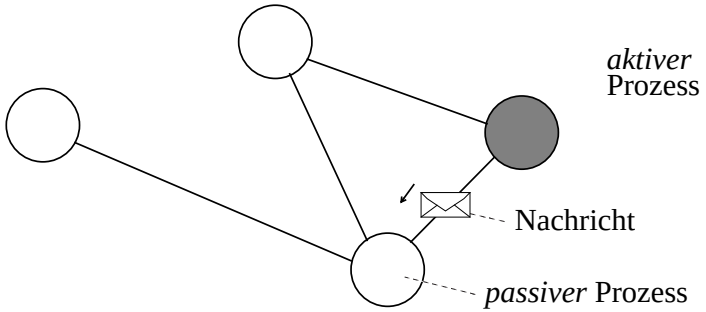


Erkennung der verteilten Terminierung?

alle passiv und keine Nachricht unterwegs

Terminierungserkennung

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

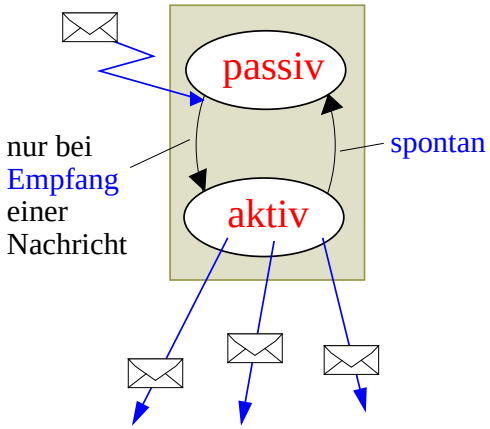


- Es gibt aktive und passive Prozesse sowie Nachrichten
  - dabei kann eine ankommende Nachricht einen passiven Prozess wieder reaktivieren
- Man möchte "irgendwie" erkennen, ob alle Prozesse passiv sind und keine Nachricht mehr unterwegs ist
  - Stagnationszustand beim verteilten Approximationsparadigma

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

"guard": Prädikat über dem lokalen Zustand

Prozess p:

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

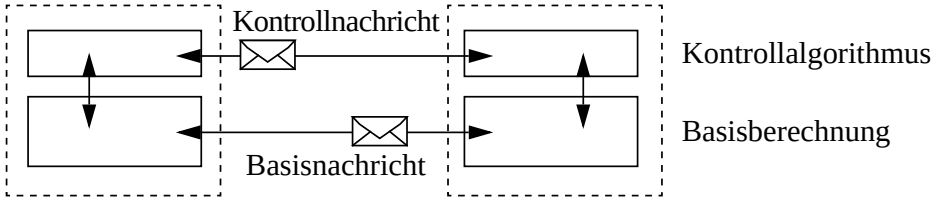
R_p: {Eine Nachricht ist angekommen}
receive <M> ; Zustand := *aktiv*

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

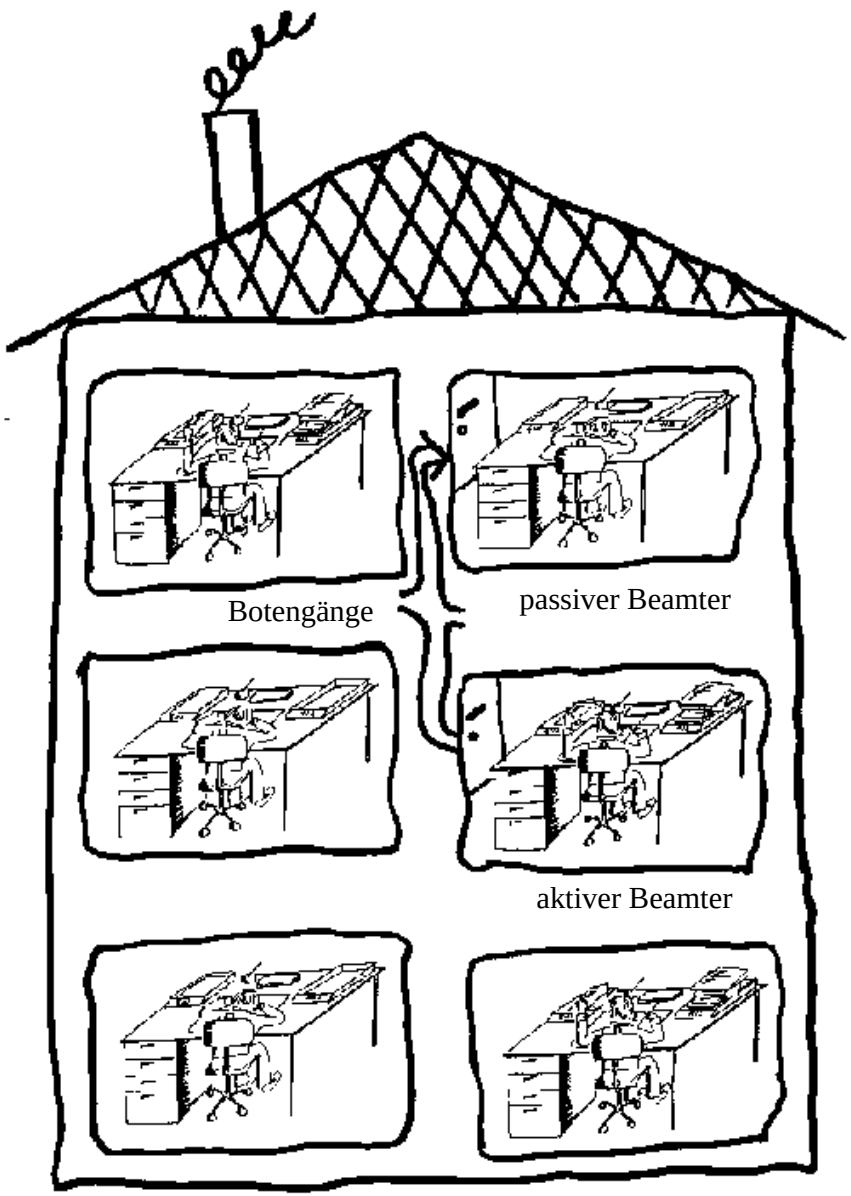
Typischerweise werden die Aktionen als "atomar" betrachtet

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

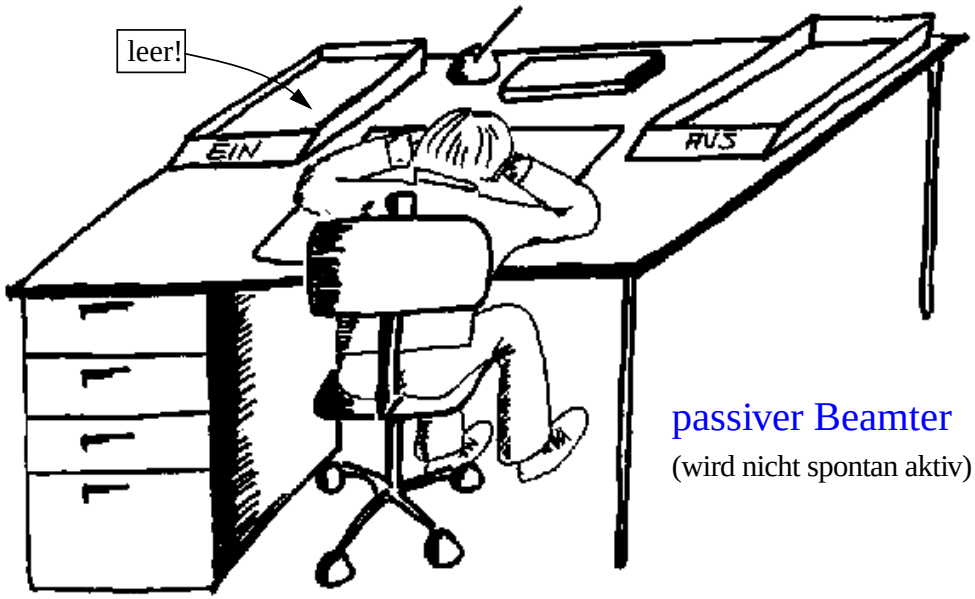
- Durch einen "überlagerten" Kontrollalgorithmus werden weitere Aktionen hinzugefügt
- "Anreichern" der Semantik der Basisberechnung für Zwecke des Kontrollalgorithmus
  - z.B. Verändern spezifischer (lokaler) Variablen
- Überlagerter Algorithmus soll Basisberechnung nicht stören
  - darf aber die Variablen, 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

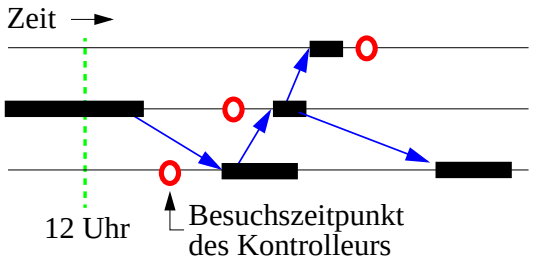
Dieser ist oft erstaunlich niedrig...

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?

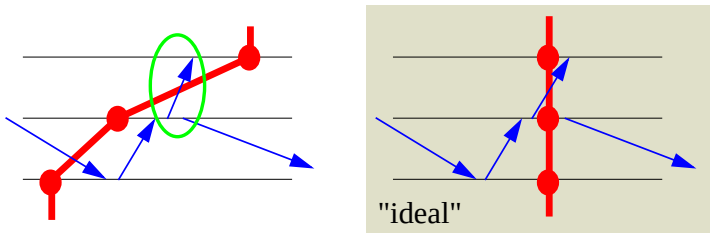
- Die *Illusion* Kontrolleurs:



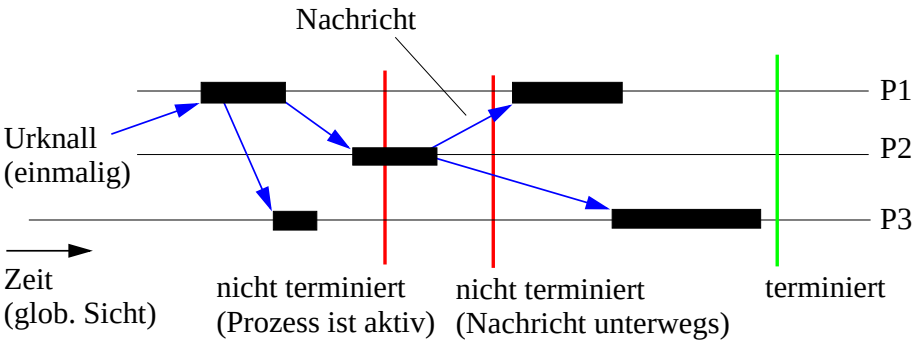
"behind the back communication"

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

- Kontrolleur macht sich ein *schiefes Bild!*

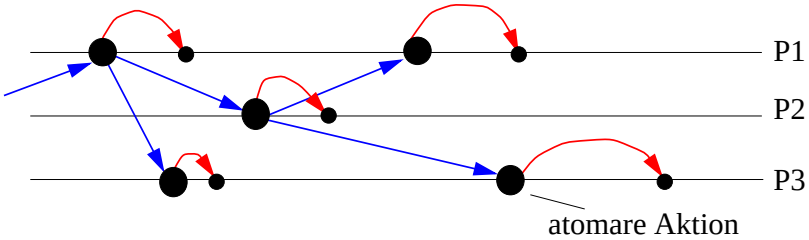


Zeitdiagramme und Atommodell



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