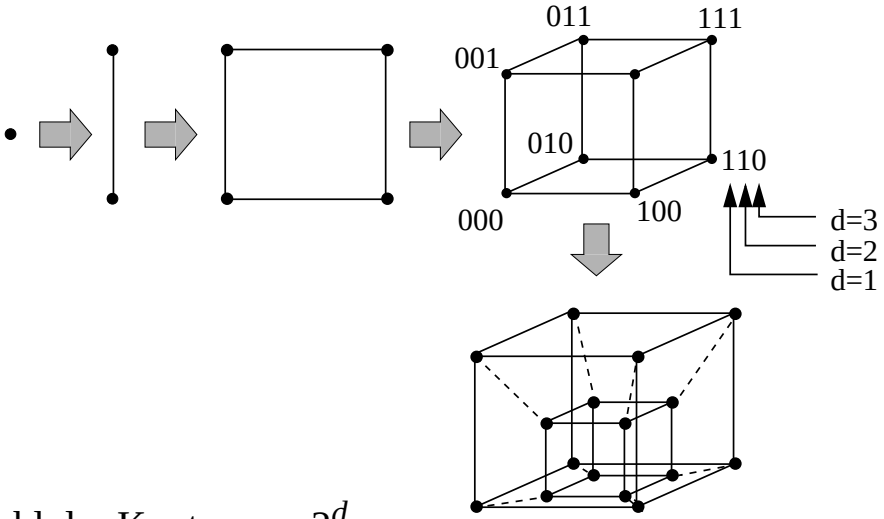
Hypercubes

- 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“



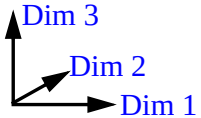
- Anzahl der Knoten $n = 2^d$
- Anzahl der Kanten = $d\ 2^{d-1}$ (Ordnung O(n log n))
  - viele Wegalternativen (Fehlertoleranz, Parallelität!)
  - maximale Weglänge: d = log n
  - mittlere Weglänge: d/2 (Beweis als Denkübung!)

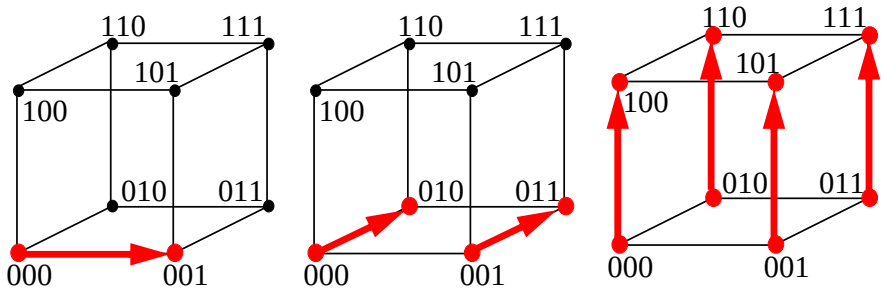
wieviele verschiedene Wege der Länge k gibt es insgesamt?

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

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

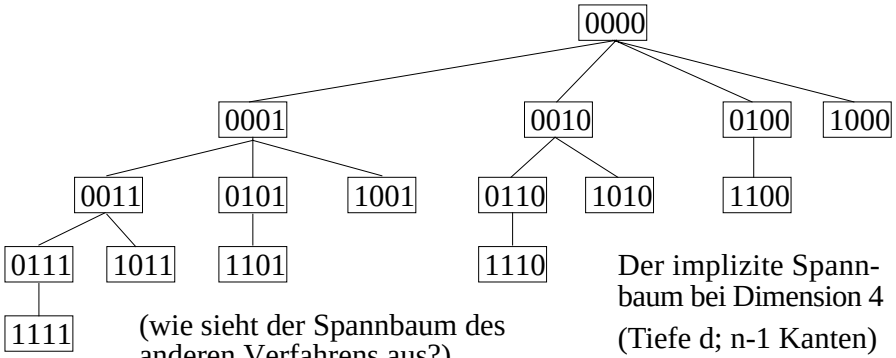
  am besten gleichzeitig, wenn dies technisch geht!

  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 X wird Knoten Y informiert?
  Setze *vorderste 1* von Y auf 0
  --> = Nummer von X



(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

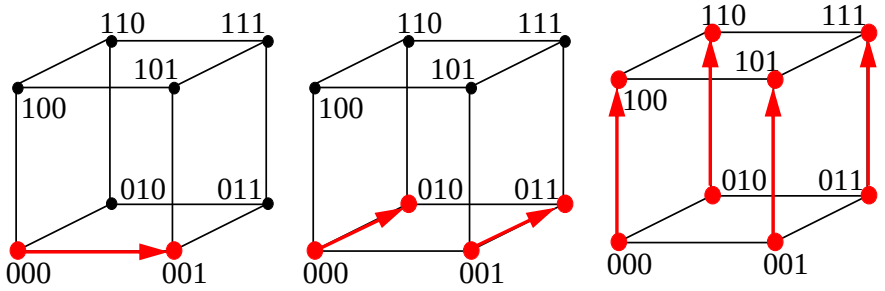
Noch ein anderer (besserer?) Algorithmus

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

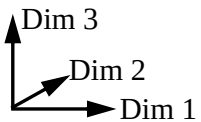
- Sender 0...0 teilt die Nachricht in d Pakete
- Sender startet für jedes Paket eine eigene “Welle”:
- *1. Paket* in *Dimension 1* senden --> 0...01
- Dann: Alle informierten Knoten (also 0...0 und 0...01) senden das Paket in Dimension 2
- Etc. Welle für Paket 1 breitet sich analog zur rekursiven Definition 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

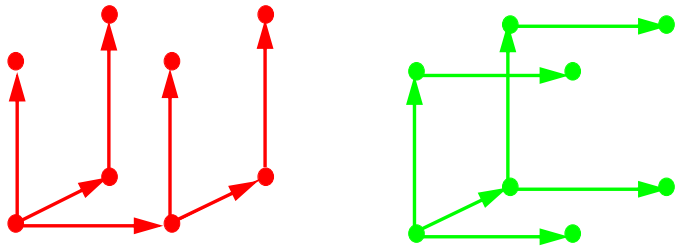
Denkübungen:
- Können so die Pakete gleichzeitig verschickt werden?
- Ist dann in jedem “Takt” pro Kante nur eine Nachricht unterwegs?
- Wieviele (kantendisjunkte ?) Spannbäume gibt es in einem Hypercube?

Veranschaulichung des Algorithmus



Die drei “Takte” der Welle von Paket 1

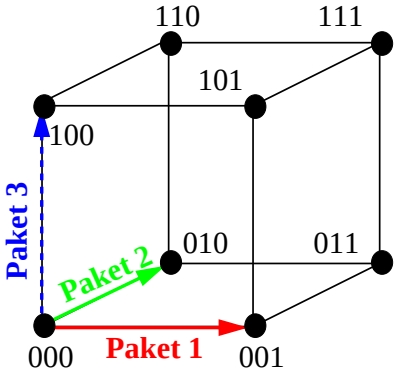




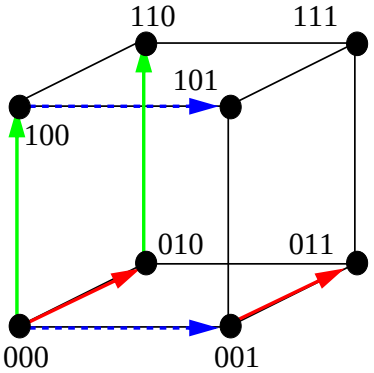
Die Spannbäume bzgl. Paket 1 und Paket 2

Parallelausführung der drei Wellen

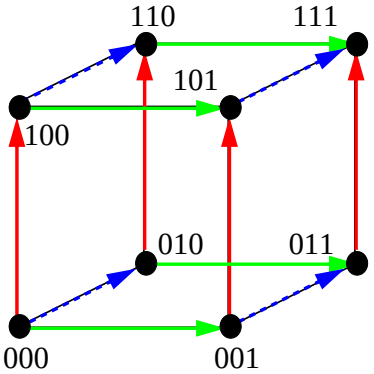
Takt 1



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!