Rechnerarchitektur SoSe 2020

Netzwerk

Jian-Jia Chen

TU Dortmund

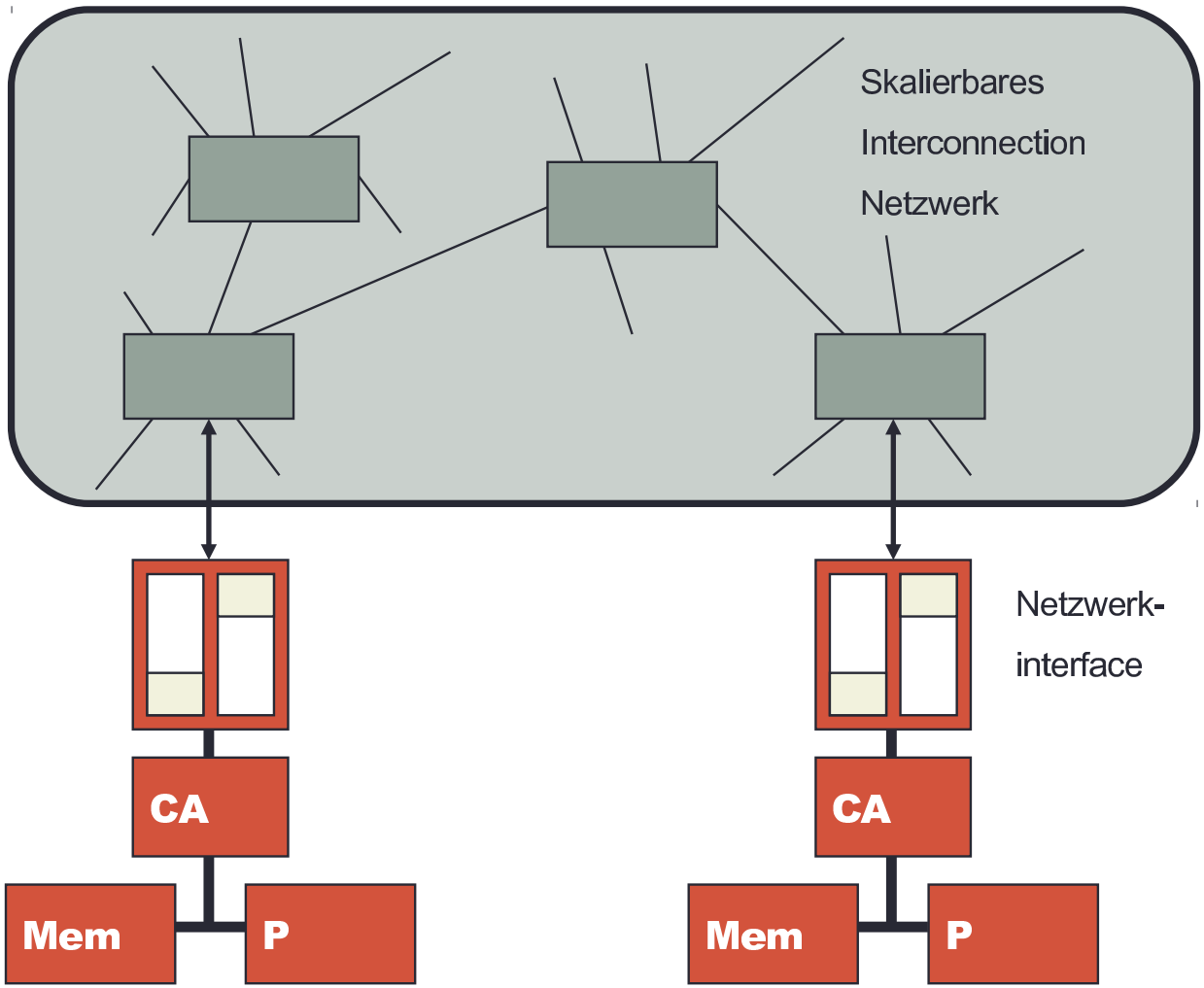
Teilweise basierend auf Material von Michael Engel, Gernot A. Fink und R. Yahyapour

Was ist die Verwendung von Interconnects?

Systemkomponenten sollen zur Kommunikation von Daten miteinander verbunden werden

- *Prozessoren werden miteinander verbunden*
- Prozessoren und Speicherbänke
- Prozessoren und Caches
- Caches und Caches
- I/O Geräte

Allgemeine Verbindungsarchitektur für MP-System



Quelle: R. Yahyapour, ehem. TU Dortmund, nach Culler/Singh/Gupta, 1999

Motivation

Warum sind Interconnects so wichtig?

- Interconnects beeinflussen maßgeblich die Skalierbarkeit des Systems:
  - Wie groß kann das System werden?
  - Wie einfach ist es dem System mehr Prozessoren hinzuzufügen?
- Interconnects beeinflussen die Performance und Energieeffizienz:
  - Wie schnell können Prozessoren, Caches, und Speicher miteinander kommunizieren?
  - Wie groß sind die Latenzen bei Speicherzugriffen?
  - Wieviel Energie wird bei der Kommunikation verwendet?

Wesentliche Charakteristika von Interconnects

Topologie

- Legt fest wie die Switches miteinander verbunden sind
- Beeinflusst das Routing, Realibilität, Durchsatz, Latenz und Fertigungskomplexität

Routing

- Wie wird eine *message* von der Quelle zum Ziel über Switches und Kanäle geleitet
- Statisch oder adaptive Verfahren

Bufferung and Flusskontrolle (Flow Control)

- Was wird im Netzwerk gespeichert?
- Ganze Pakete, Teile von Paketen, etc?
- Wie wird der Fluss bei überbelegung geregelt?
- Fragestellung ist eng mit dem Routing verbunden

Verbindungsstukturen von Interconnects

- Das Verbindungsnetz kann definiert werden als Graph bestehend aus
  - *Rechenknoten*,
  - *Kommunikationsknoten* (= Switches) und
  - *Kommunikationskanälen*.
- Netzwerk-Topologie beschreibt Verbindungsstruktur zwischen Knoten mit Kommunikationsfähigkeiten (d.h. i.d.R. *Kommunikationsknoten*)
- *Kommunikationsknoten* i.d.R. als Switches ausgelegt

Bewertungskriterien für Netzwerke

- Hardware Komplexität (Kosten)
- Latenz in der Anzahl an Hops oder Nanoseconds
- Maß an Übertragungskonflikten (Contention)
- Energieverbrauch
- Bandbreite
- ...

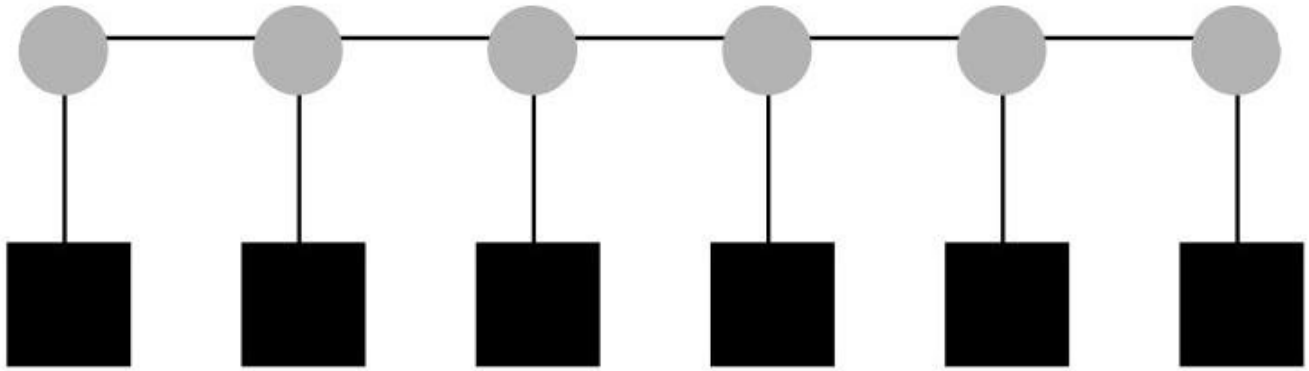
Bewertungskriterien für Netzwerke

- *Knoten-Grad*:
  Anzahl der mit einem Knoten oder Schalter verbundene Kanäle
  (betrachtet werden hier i.d.R. nur Kommunikationsknoten!)
- *Grad eines Netzwerkes*:
  Der maximale Knotengrad im Netzwerk
- *Durchmesser*:
  Länge des maximalen kürzesten Pfades in dem Netzwerk zwischen zwei beliebigen Knoten
- *Halbierungsbandbreite* (*bisection bandwidth*):
  - Die minimale Anzahl der gerichteten Kanten in einem Netzwerk zwischen zwei gleichen Hälften des Netzwerkes bei beliebiger Unterteilung, bzw.
  - Anzahl paralleler Verbindungen zwischen zwei Netzwerkhälften

Topologien

Topologien

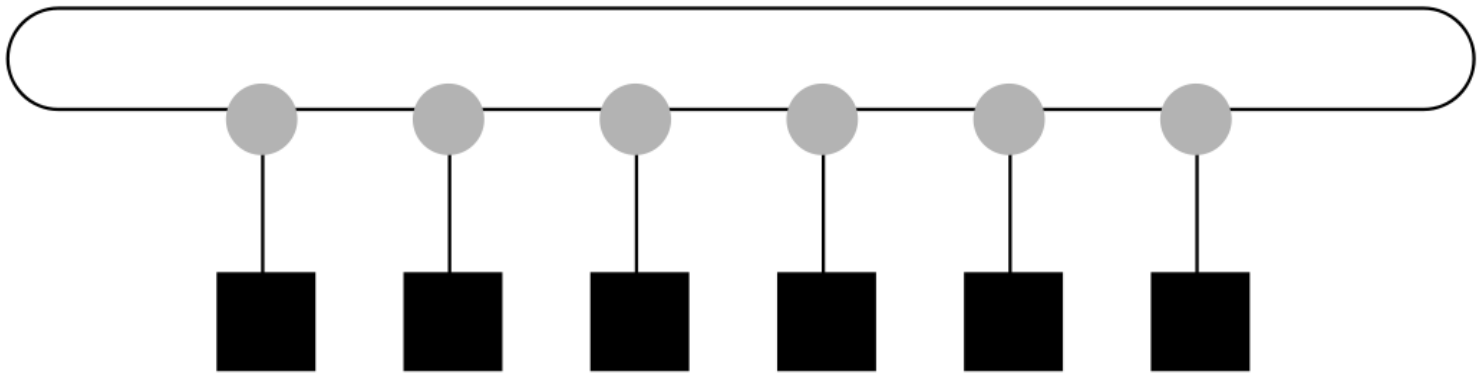
Lineare Felder / Arrays



Grad der Knoten:	2 für alle inneren Knoten
	1 für Randknoten
Durchmesser:	$N - 1$
Halbierungsbandbreite:	1 (i.S. Anzahl d. Links)
Skalierbarkeit:	keine Hardwarebeschränkung

Topologien II

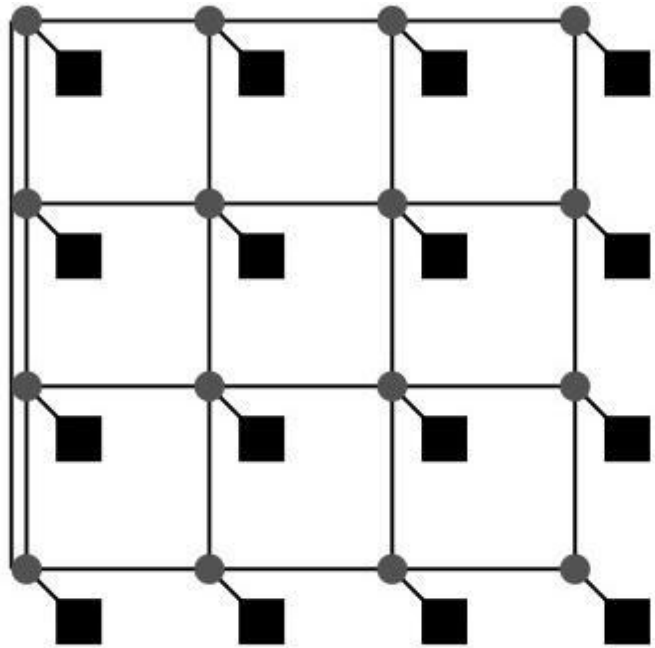
Ring-Topologien



Grad der Knoten:	2
Durchmesser:	$\lfloor N/2 \rfloor$ im bidirektionalen Fall
	$N-1$ im unidirektionalen Fall
Halbierungsbandbreite:	2 (i.S. Anzahl d. Links)
Skalierbarkeit:	keine Hardwarebeschränkung

Topologien III

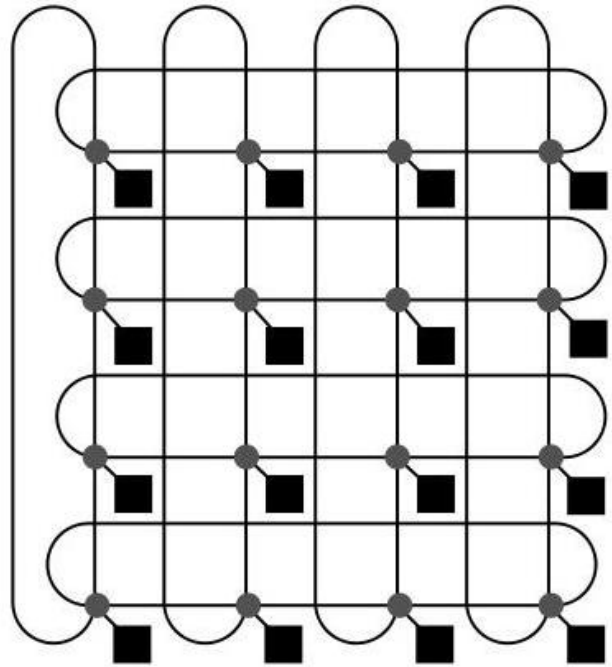
Gitter / Meshes
(in 2 und k Dimensionen)



	2-D ($N = 16$)	k-D	
Grad der Knoten:	2	k	für Eckknoten
	≤ 4	$\leq 2k$	für innere Knoten
Durchmesser:	6	$k\left(\sqrt[k]{N} - 1\right)$	
Halbierungsbandbreite:	4	$\left(\sqrt[k]{N}\right)^{k-1}$	

Topologien VI

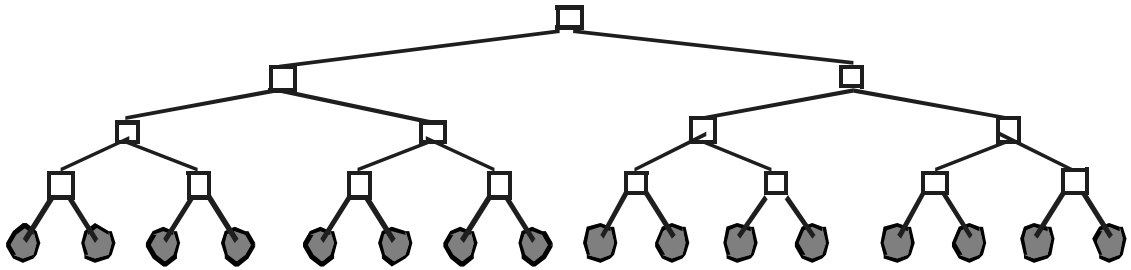
Torus

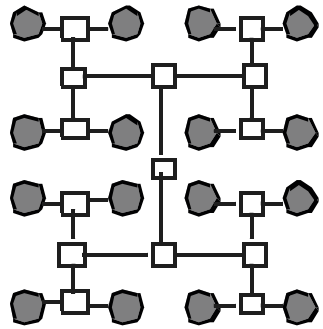


	2-D ($N = 16$)	k-D
Grad der Knoten:	4	$2k$
Durchmesser:	4	$k\lfloor \sqrt[k]{N/2} \rfloor$
Halbierungsbandbreite:	8	$2\left(\sqrt[k]{N}\right)^{k-1}$

Topologien V

Bäume (üblicherweise binär)



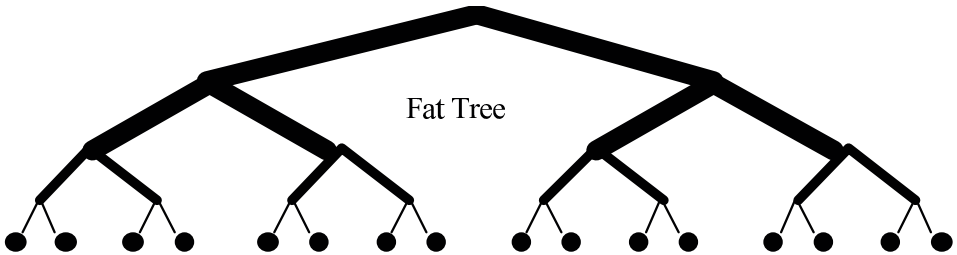


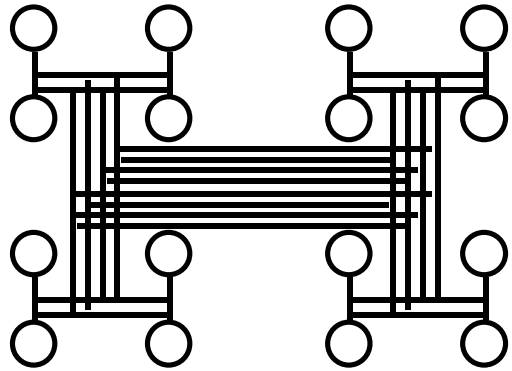
Grad der Knoten:	1 für Blatt-/Endknoten
	2 für Wurzelknoten
	3 für alle inneren Knoten
Durchmesser:	$2 \cdot \lfloor \log_2 N \rfloor$
Halbierungsbandbreite:	1

Hinweis: Bei baumartigen Verbindungsstrukturen sind Rechenknoten nur mit Blattknoten verbunden.

Topologien VI

Fat Trees (≈ "dicke" Bäume)



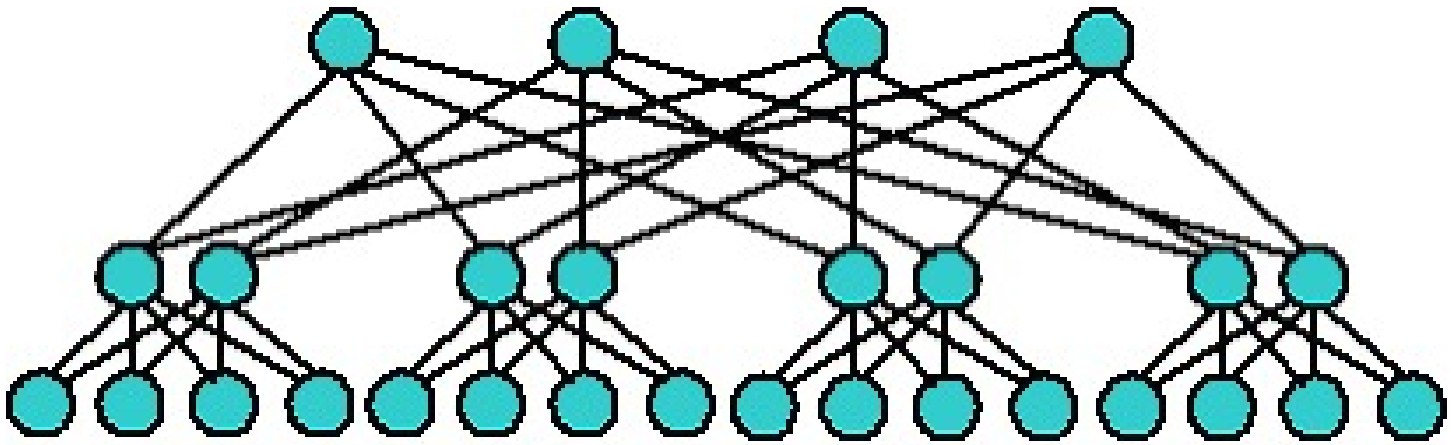


Anzahl der Verbindungen verdoppelt sich mit jeder Ebene im Baum

Grad der Knoten:	1 für Blattknoten (= Rechenknoten)
Durchmesser:	$2 \cdot \lfloor \log_2 N \rfloor$
Halbierungsbandbreite:	$\log_2 N$

Topologien VI

Redundante *Fat Trees*

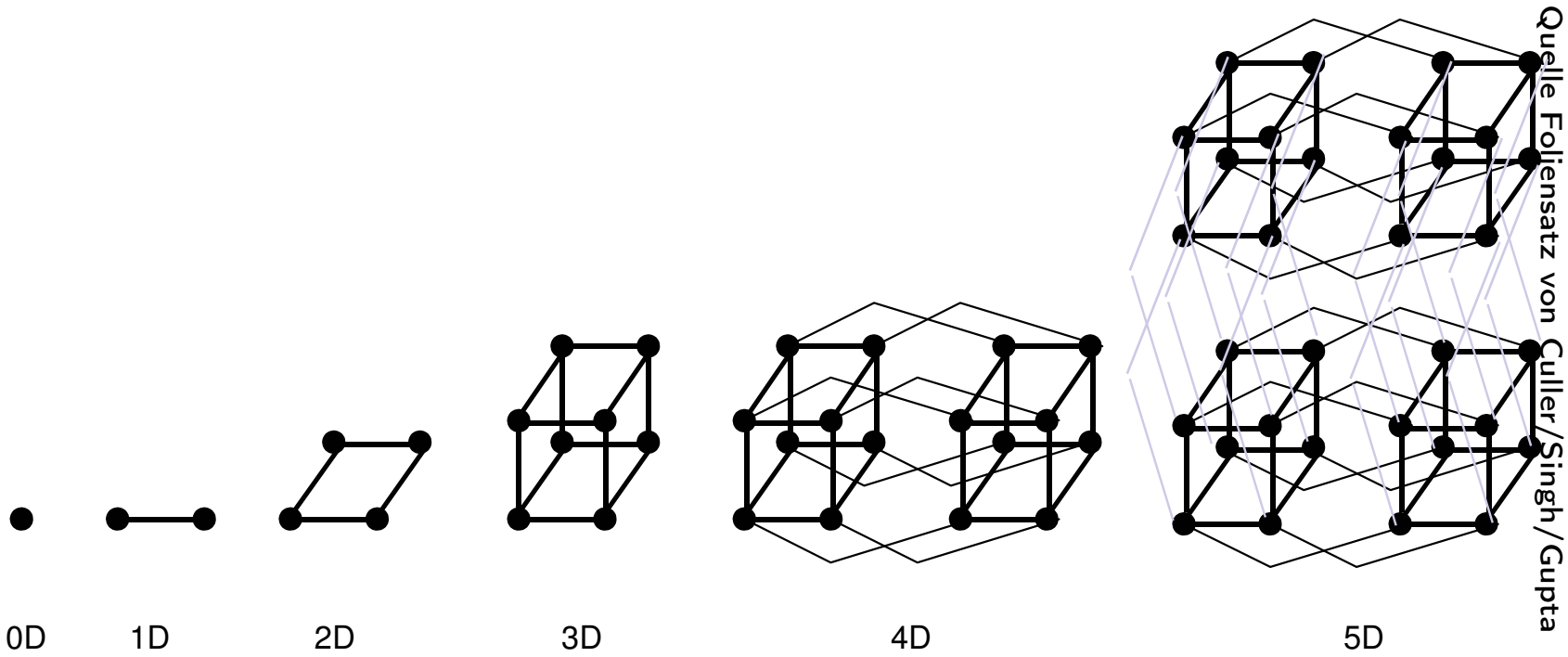


Quelle 24/7 Solutions

Weitere Verbesserung der Redundanz durch Replikation von Kommunikationsknoten im Baum und Verbindung zu *allen* Nachfolgerknoten (insbes. mehrer Wurzeln)

Topologien VII

Hyper-Kuben (hochdimensionale Würfel)



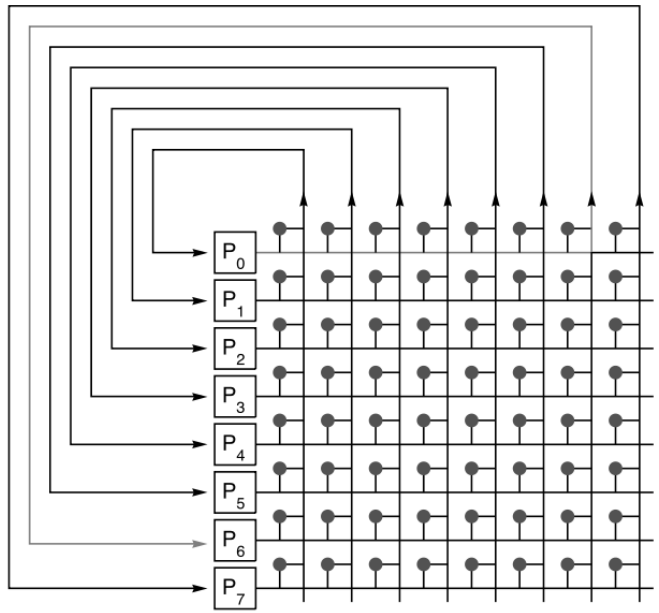
	3-D ($N = 8$)	k-D
Grad der Knoten:	3	$k = \log_2 N$
Durchmesser:	3	k
Halbierungsbandbreite:	4	2^{k-1}

Realisierung von Switches

Realisierung von Switches

Cross Bar

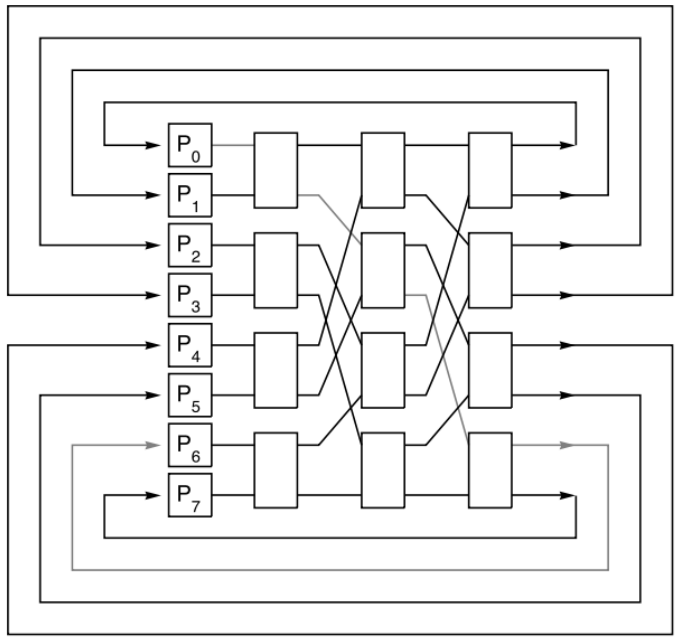
- Schaltung direkter Verbindungen aller möglichen Knotenpaare



- Vorteil: Kommunikation zwischen Knoten via Switch in einem Schritt möglich (nach Schaltung d. Verbindung)
- Nachteil: Extremer Hardware-Aufwand (N^2 Schalter)

Realisierung von Switches II

Omega-Netzwerk



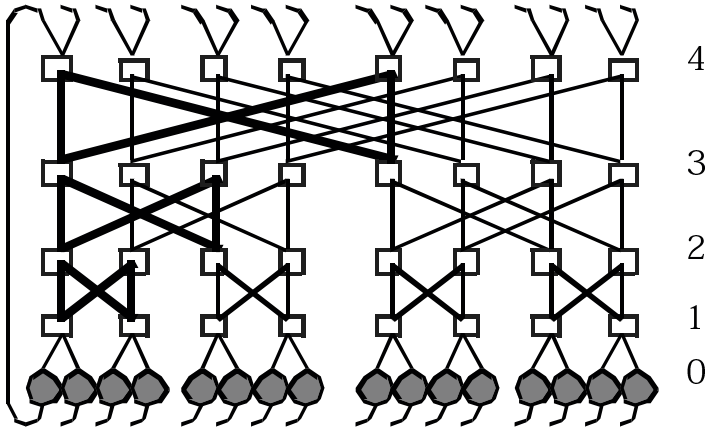
- Schaltung von Verbindungen aller möglichen Knotenpaare in mehreren Stufen

$\Rightarrow$ *multistage interconnection network (MIN)*

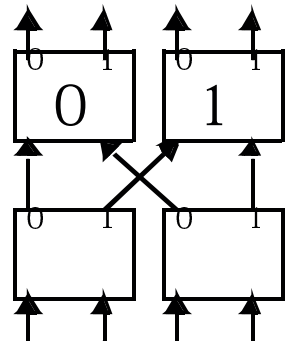
- Vorteil gegenüber Cross Bar: Geringerer Hardware-Aufwand (nur $(N/2)\log_2 N$ anstelle N^2 Switch-Elemente)
- Nachteile:
  - Weiterleitung von Daten erfordert mehrere Teilschritte
  - Gegenseitige Blockierung von Datenpfaden möglich

Realisierung von Switches III

Butterfly-Netzwerke



16 node butterfly



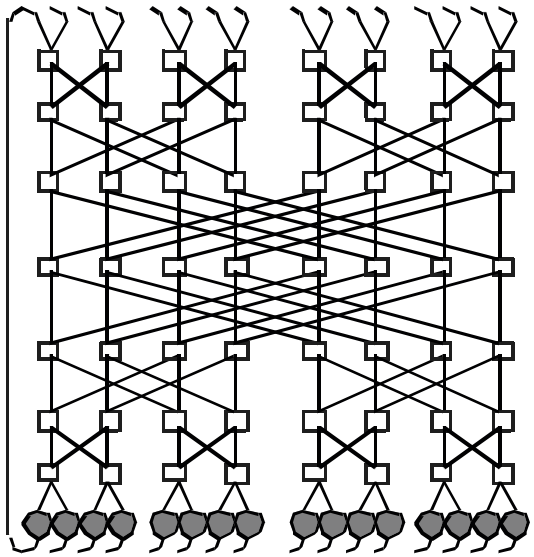
building block

- Prinzip vergleichbar mit Omega-Netzwerk
- Kommunikationswege folgen dem Prinzip des *bit reversal*
- Rekursiver Aufbau:
  Elementares Schaltelement entspricht 2×2 Butterfly
- Logarithmische Tiefe

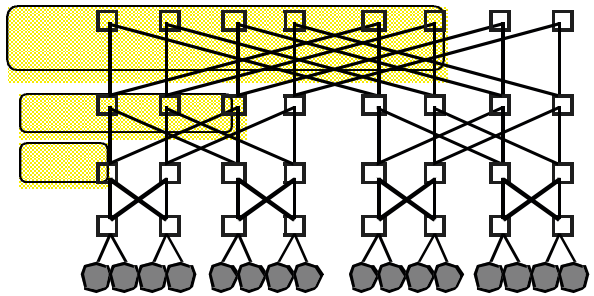
Realisierung von Switches VI

Benes Netzwerke

16-node Benes Network (Unidirectional)



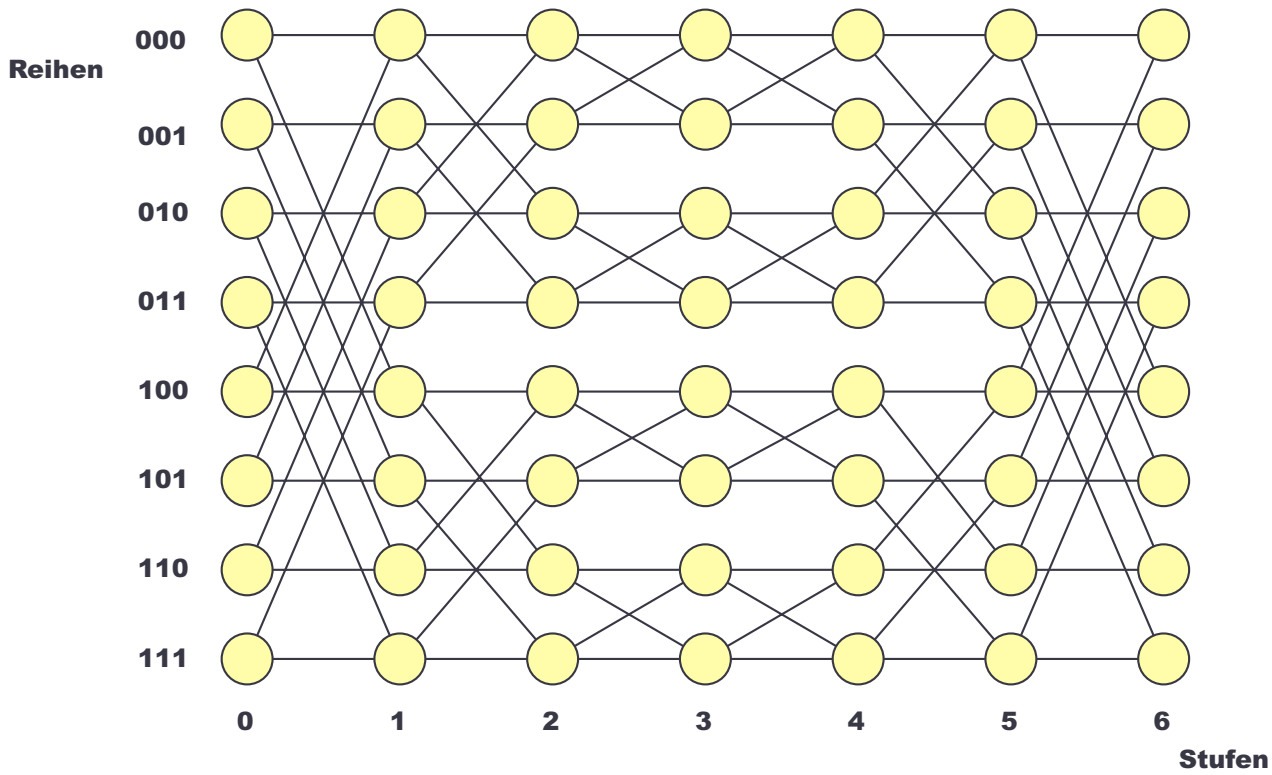
16-node 2-ary Fat-Tree (Bidirectional)



- Aufbau aus zwei rückwärtig verbundenen Butterfly-Netzwerken
- Alle Permutationen von paarweisen Knotenverbindungen schaltbar (bei offline Berechnung der Routen)
- Kann durch “Falten” in redundante Baumstruktur überführt werden (dabei Bildung von “fat nodes”)

Realisierung von Switches VII

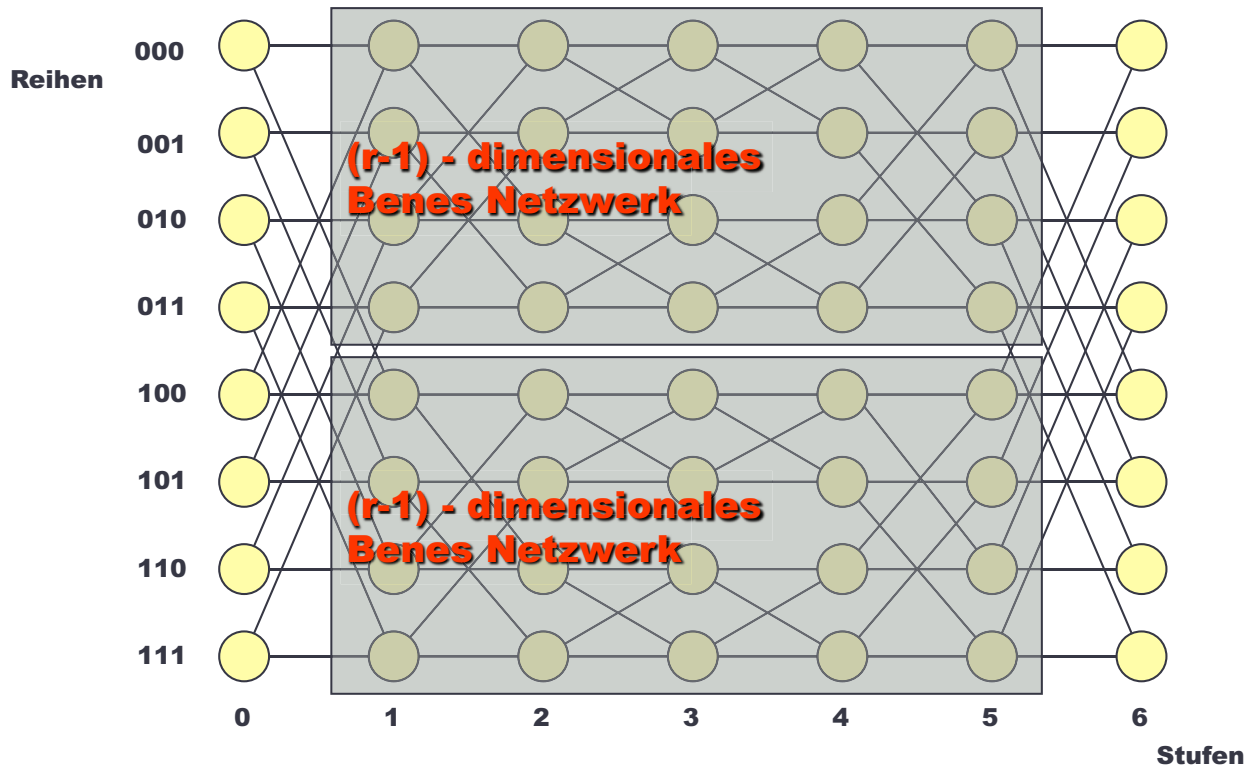
Benes Netzwerke



Quelle: R. Yahyapour, ehem. TU Dortmund

Realisierung von Switches VII

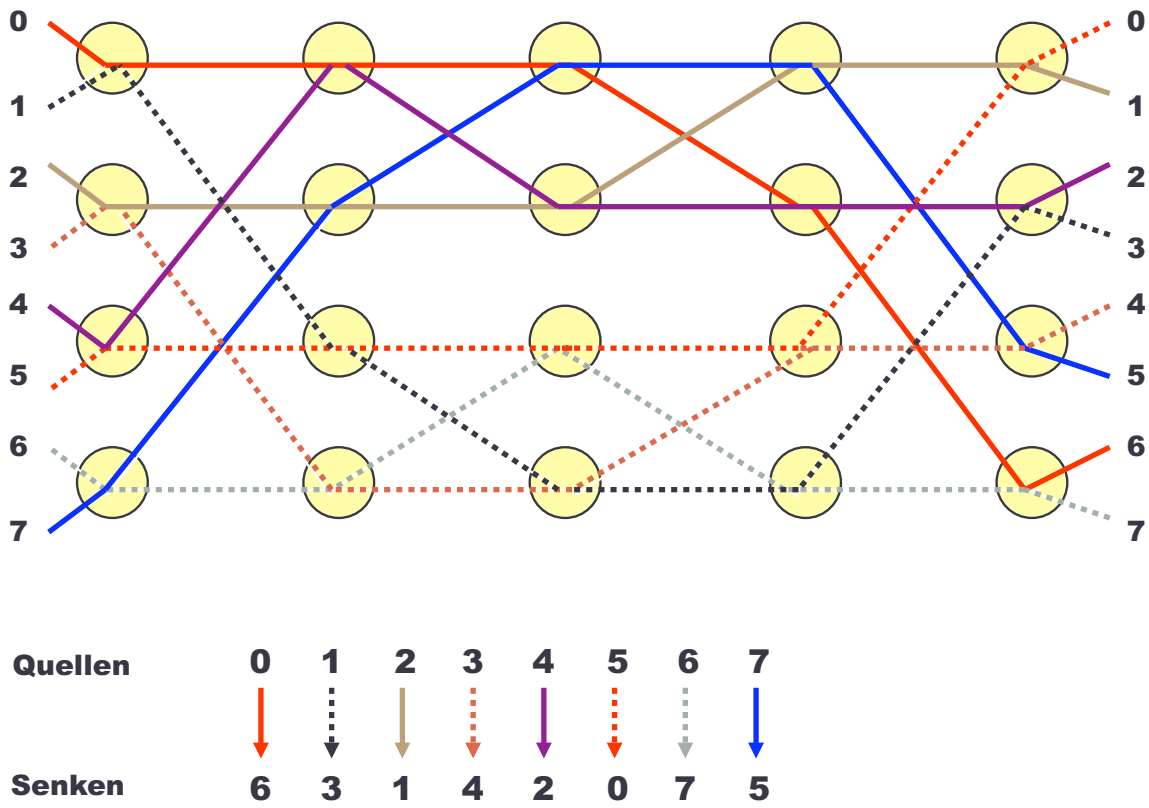
Benes Netzwerke



Quelle: R. Yahyapour, ehem. TU Dortmund

Realisierung von Switches VII

Benes Netzwerke



Quelle: R. Yahyapour, ehem. TU Dortmund

Verbindungsstrukturen: Überblick

Company	System [network] name	Max. number of nodes [× # CPUs]	Basic network topology	Injection [reception] node BW in MB/sec	# of data bits per link per direction	Raw network link BW per direction in MB/sec	Raw network bisection BW (bidirectional) in GB/sec
Intel	ASCI Red Paragon	4816 [× 2]	2D mesh 64 × 64	400 [400]	16 bits	400	51.2
IBM	ASCI White SP Power3 [Colony]	512 [× 16]	Bidirectional MIN with 8-port bidirectional switches (typically a fat tree or Omega)	500 [500]	8 bits (+1 bit of control)	500	256
Intel	Thunder Itanium2 Tiger4 [QsNet^{II}]	1024 [× 4]	Fat tree with 8-port bidirectional switches	928 [928]	8 bits (+2 of control for 4b/5b encoding)	1333	1365
Cray	XT3 [SeaStar]	30,508 [× 1]	3D torus 40 × 32 × 24	3200 [3200]	12 bits	3800	5836.8
Cray	X1E	1024 [× 1]	4-way bristled 2D torus (~23 × 11) with express links	1600 [1600]	16 bits	1600	51.2
IBM	ASC Purple pSeries 575 [Federation]	>1280 [× 8]	Bidirectional MIN with 8-port bidirectional switches (typically a fat tree or Omega)	2000 [2000]	8 bits (+2 bits of control for novel 5b/6b encoding scheme)	2000	2560
IBM	Blue Gene/L eServer Sol. [Torus Net.]	65,536 [× 2]	3D torus 32 × 32 × 64	612.5 [1050]	1 bit (bit serial)	175	358.4

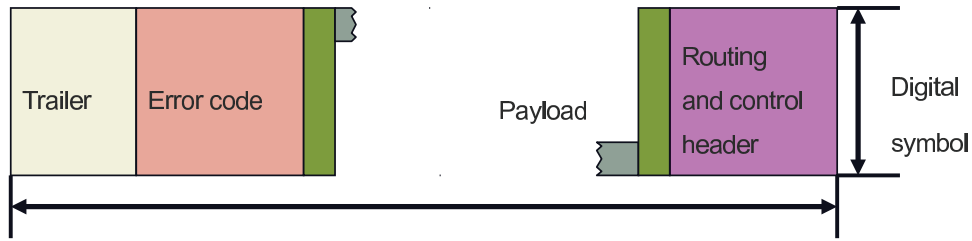
MIN = multistage interconnection network

Routing

Kommunikation in Verbindungsnetzen

- Prinzip: Datenaustausch erfolgt paketorientiert.
  Typisches Paketformat:

Quelle: R. Yahyapour, ehem. TU Dortmund

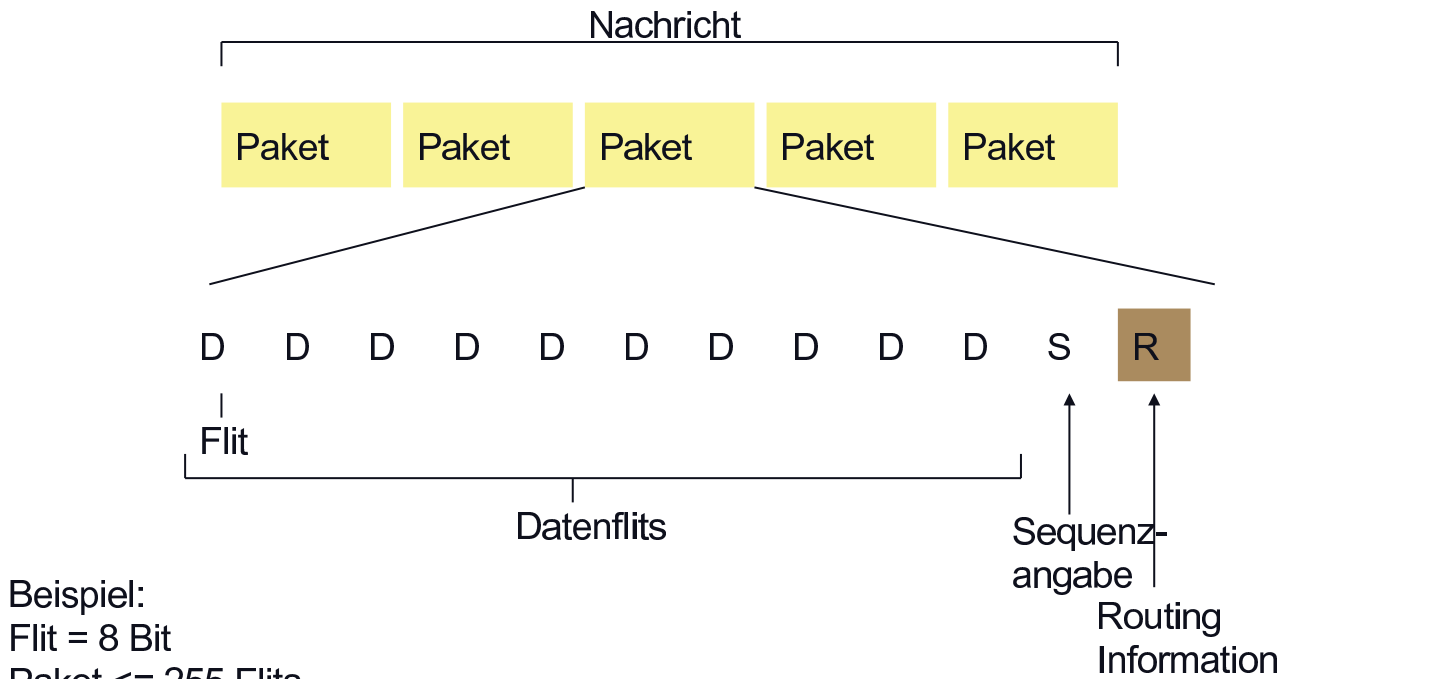


- Zwei grundlegende Mechanismen der Abstraktion:
  - Kapselung: Protokollinformation einer höheren Ebene wird uninterpretiert auf niedrigerer Ebene mit übertragen
  - Fragmentierung: Aufteilung der Nachrichtenstruktur (einer höheren Ebene) in Folge von Übertragunseinheiten

Beachte:
- Abstraktionshierarchie in Parallelrechnern i.d.R. flacher als im LAN/WAN-Bereich
- Greifen enger/effizienter ineinander

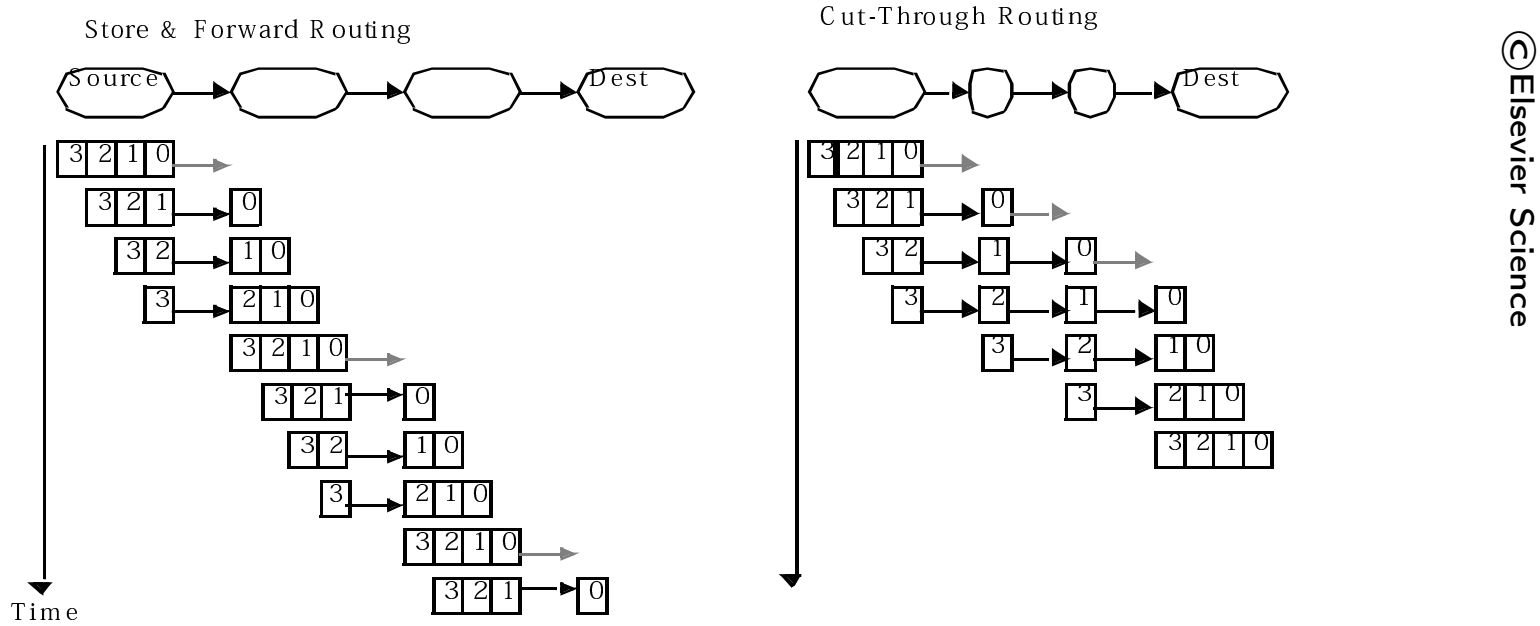
Kommunikation in Verbindungsnetzen II

- "Mikrostruktur" von Datenpaketen: Aufteilung in sog. Flits (*flow control bits*)



Quelle: R. Yahyapour, ehem. TU Dortmund

Routing-Mechanismen



- *Store-and-Forward*: Pakete werden erst weitergeleitet, wenn komplett empfangen
- *Cut-Through Routing*: Flits eines Pakets werden einzeln weitergeleitet; Entscheidung für Routing zum nächsten Knoten fällt bereits, wenn Paketheader bekannt
  $\Rightarrow$ Übertragungslatenz ähnlich Pipelining reduziert!

Routing-Mechanismen II

Grundlegendes Problem von Routing-Verfahren:
Auftreten von Kollisionen

- Mehr als ein Paket/Flit versucht dieselbe Verbindung zwischen zwei Knoten zu überqueren
- Kollisionen können in jedem Vermittlungsknoten auftreten

  [im Gegensatz zu *circuit switching*: komplette Route wird zu Beginn der Kommunikation aufgebaut]
- Ein Paket/Flit kann übertragen werden
- Wie andere(s) Paket(e) behandeln?

$\Rightarrow$ Vier Möglichkeiten zur Behandlung von Kollisionen

Routing-Mechanismen III

Möglichkeiten zur Kollisionsbehandlung:

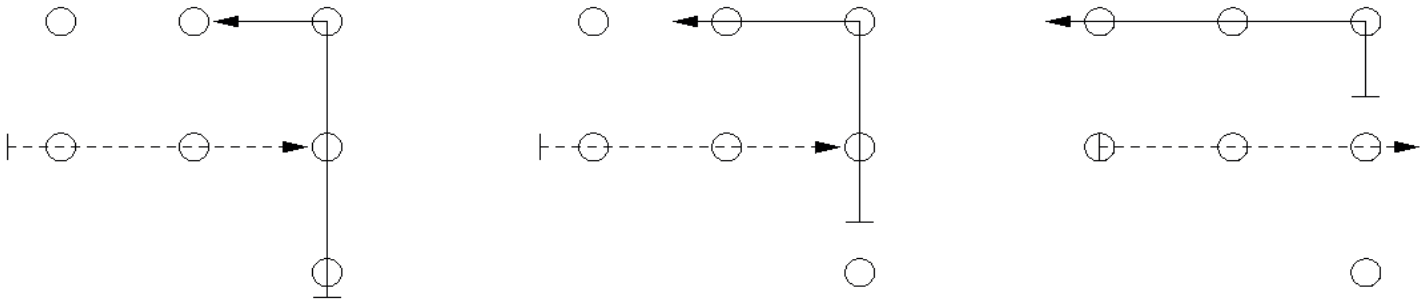
- *Verwerfen* des Pakets (im LAN/WAN-Bereich, insbes. Internet)
  - Netzwerkprotokoll muss Verlust detektieren und ...
  - Neuübertragung initiieren.
- *Puffern* der blockierten Übertragung
  - in Verbindung mit Cut-Through Routing: *Virtual Cut-Through*

    [nachfolgende Flits werden am Blockadepunkt in Puffer "aufgesammelt"]
  - Puffergröße ggf. problematisch!
- *Blockieren* der weiteren Übertragung
  - in Verbindung mit Cut-Through Routing: *Wormhole Routing*

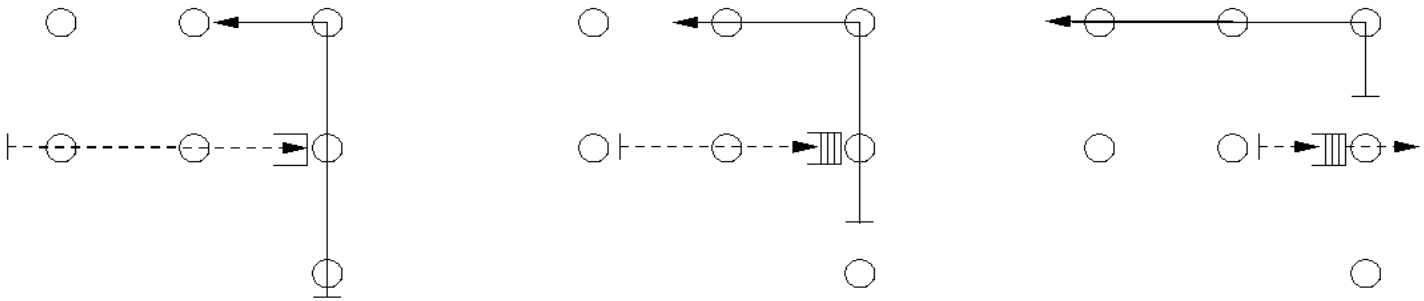
    [nur wenige (im Extremfall: eins) nachfolgende Flits werden am Blockadepunkt gepuffert]
  - Rückstau problematisch

    [blockiert bei Wormhole Routing ggf. n Knoten im Netzwerk, n = Anz. der Flits]
- *Umleitung* von Paketen
  - Erfordert entsprechendes Routing-Verfahren

Routing: Wormhole vs. Virtual Cut-Through



a) Wormhole



b) Virtual-Cut-Through

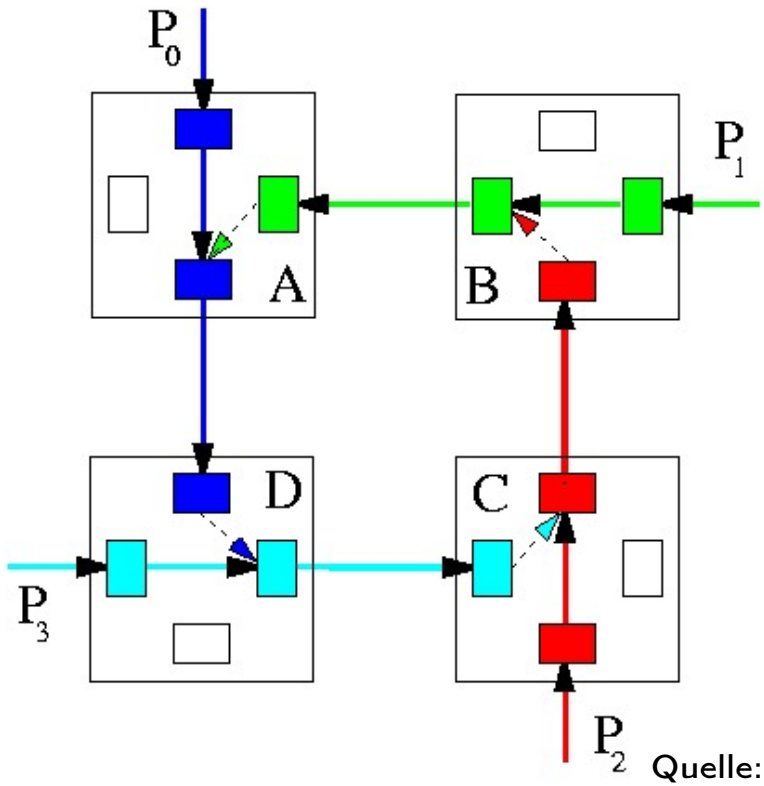
Quelle: Tichy/Pankratius/Jannesari, KIT

Beachte: Nachrichten-Trailer zur Freigabe der Kommunikationsknoten erforderlich

Routing-Mechanismen V

Weitergehendes Problem von Routing-Verfahren: Deadlocks

- Problem vergleichbar der Situation beim gegenseitigen Ausschluss: Für Übertragung von Nachrichten müssen physikalische Kanäle *exklusiv* belegt werden



Quelle: Univ. Wisconsin-Madison

Routing-Mechanismen V

Weitergehendes Problem von Routing-Verfahren: Deadlocks

- Problem vergleichbar der Situation beim gegenseitigen Ausschluss: Für Übertragung von Nachrichten müssen physikalische Kanäle *exklusiv* belegt werden
  $\Rightarrow$ Zyklus im Kanalbelegungsgraphen signalisiert Deadlock

Mögliche Lösungen:

- *Virtuelle Kanäle* (und strikte Ordnung bei der Kanalbelegung)
- Einschränkung der gültigen Routen:
  Routingverfahren selbst verhindert Entstehung von Zyklen
  $\Rightarrow$ Up*-Down*-Routing, Turn-Model Routing

Up*-Down* Routing

Verfahren zum *deadlock-freien* (Wormhole-)Routing

Strategie ähnlich zum Routing in einem baumartigen Netzwerk:

- Kanäle zwischen Verbindungsknoten sind bidirektional
- Konstruiere minimalen Spannbaum über die Verbindungsknoten
  $\Rightarrow$ Knoten können nach Position im Baum sortiert werden
- Kanten (d.h. richtungsgebundene Kanäle) werden als *up* bzw. *down* markiert (bzgl. Baum)

$\Rightarrow$ Jeder Zielknoten kann von jedem Quellknoten über Folge von *up* und *down*-Kanten erreicht werden

$\Rightarrow$ Routen werden eingeschränkt auf:

- Zunächst beliebige Folge von *up* Kanten
- Dann "Umkehr" und beliebige Folge von *down* Kanten

Turn-Model Routing

Verfahren zum *deadlock-freien* Routing in gitterartigen Netzwerkstrukturen

Beobachtung: 8 Richtungswechsel ($\hat{=}$ *turns*) in Routen mit bi-direktionalen Kanälen möglich (im 2D-Fall)

Grundidee: Richtungswechsel so einschränken, dass keine Zyklen entstehen können

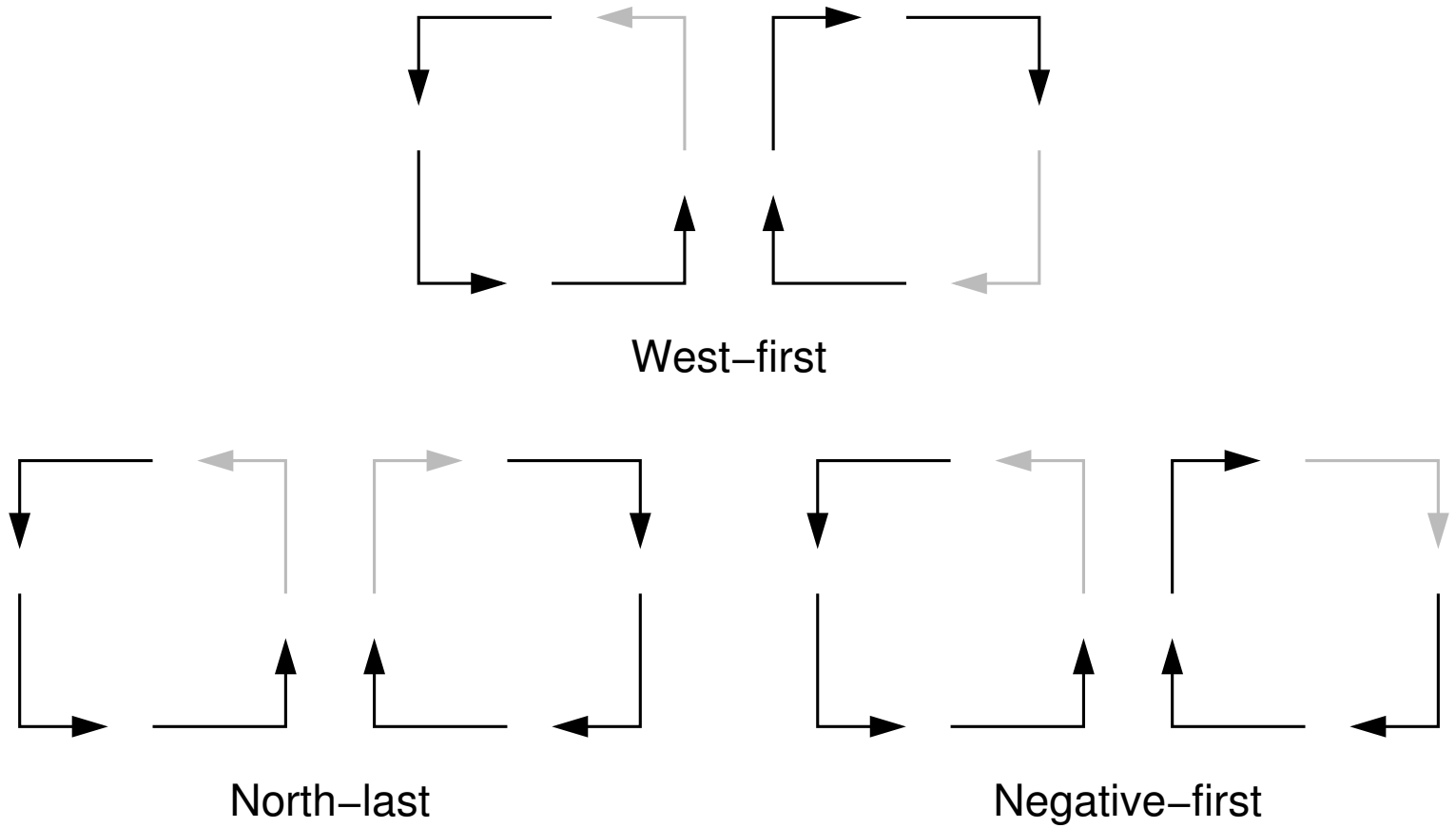
Einfaches Verfahren: *dimension order routing*

- Dimensionen im Gitter werden sortiert (z.B. x vor y vor ...)
- Dann Routing gemäß der Dimensionsordnung vornehmen (d.h. zuerst in $\pm\Delta x$ dann $\pm\Delta y$, ...; dabei Δx etc. $\hat{=}$ rel. Abstand zwischen Quelle und Ziel im Gitter)

Allgemeinere Lösungen: Verhindern von je zwei Richtungswechseln ausreichend für deadlock-freies Routing

Turn-Model Routing II

Beispiele für gültige Routingalgorithmen/-einschränkungen in 2D



nach Culler/Gupta/Singh

(jeweils inkl. entsprechend rotierter Schemata)