

High Performance Computing – Programming Paradigms and Scalability

Part 2: High-Performance Networks

PD Dr. rer. nat. habil. Ralf-Peter Mundani
Computation in Engineering (CiE)
Scientific Computing (SCCS)

Summer Term 2015

Overview

- **some definitions**
- static network topologies
- dynamic network topologies
- examples

*640k is enough for anyone,
and by the way, what's a network?*
—William Gates III,
chairman Microsoft Corp., 1984

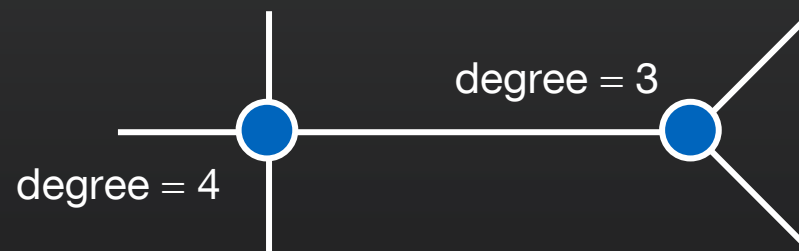
Some definitions

- reminder: protocols

3-component model		ISO/OSI model		internet protocols (examples)
application	7	application layer		data transfer, email
communication system	6	presentation layer		
	5	session layer		
	4	transport layer		TCP, UDP
	3	network layer		IP, ICMP, IGMP
network	2	data link layer	logical link control medium access control	network adaptation
	1	physical layer		

Some Definitions

- **degree (node degree)**
 - number of connections (incoming and outgoing) between this node and other nodes
 - **degree of a network = max. degree of all nodes in the network**
 - higher degrees lead to
 - more parallelism and bandwidth for the communication
 - more costs (due to a higher amount of connections)
- objective: keep degree and, thus, costs small



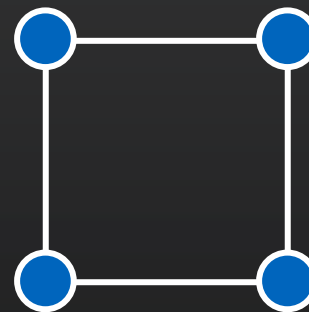
Some Definitions

- **diameter**
 - distance of a pair of nodes (length of the shortest path between a pair of nodes), i.e. the amount of nodes a message has to pass on its way from the sender to the receiver
 - **diameter of a network = max. distance of all pairs of nodes in the network**
 - higher diameters (between two nodes) lead to
 - longer communications
 - less fault tolerance (due to the higher amount of nodes that have to work properly)
 - objective: small diameter



Some Definitions

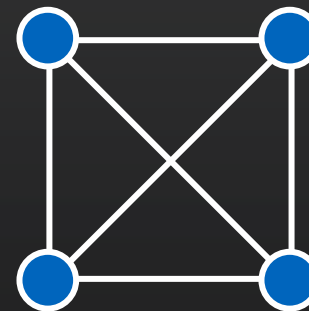
- **connectivity**
 - min. amount of edges (cables) that have to be removed to disconnect the network, i.e. the network falls apart into two loose sub-networks
 - higher connectivity leads to
 - more independent paths between two nodes
 - better fault tolerance (due to more routing possibilities)
 - faster communication (due to the avoidance of congestions in the network)
 - objective: high connectivity



connectivity = 2

Some Definitions

- **bisection width**
 - min. amount of edges (cables) that have to be removed to separate the network into two equal parts (bisection width \neq connectivity, see below)
 - important for determining the amount of messages that can be transmitted in parallel between one half of the nodes to the other half without the repeated usage of any connection
 - extreme case: Ethernet with bisection width = 1
 - objective: high bisection width (ideal: amount of nodes/2)



bisection width = 4
(connectivity = 3)

Some Definitions

- **blocking**
 - a desired connection between two nodes cannot be established due to already existing connections between other pairs of nodes
 - objective: non-blocking networks
- **fault tolerance (redundancy)**
 - connections between (arbitrary) nodes can still be established even under the breakdown of single components
 - a fault-tolerant network has to provide at least one redundant path between all arbitrary pairs of nodes
 - *graceful degradation*: the ability of a system to stay functional (maybe with less performance) even under the breakdown of single components

Some Definitions

- **bandwidth**
 - max. transmission performance of a network for a certain amount of time
 - bandwidth B in general measured as megabits or megabytes per second (Mbps or MBps, resp.), nowadays more often as gigabits or gigabytes per second (Gbps or GBps, resp.)
- **bisection bandwidth**
 - max. transmission performance of a network over the bisection line, i.e. sum of single bandwidths from all edges (cables) that are “cut” when bisecting the network
 - thus bisection bandwidth is a *measure of bottleneck bandwidth*
 - units are same as for bandwidth

Overview

- some definitions
- **static network topologies**
- dynamic network topologies
- examples

Static Network Topologies

- **to be distinguished**
 - *static networks*
 - fixed connections between pairs of nodes
 - control functions are done by the nodes or by special connection hardware
 - *dynamic networks*
 - no fixed connections between pairs of nodes
 - all nodes are connected via inputs and outputs to a so called switching component
 - control functions are concentrated in the switching component
 - various routes can be switched

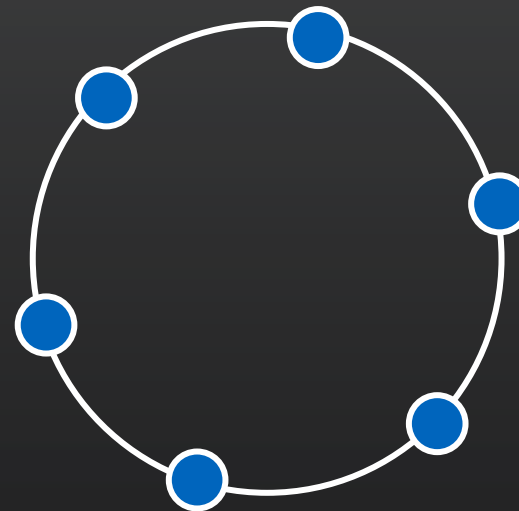
Static Network Topologies

- chain (linear array)
 - one-dimensional network
 - N nodes and $N-1$ edges
 - degree = 2
 - diameter = $N-1$
 - bisection width = 1
 - drawback: too slow for large N



Static Network Topologies

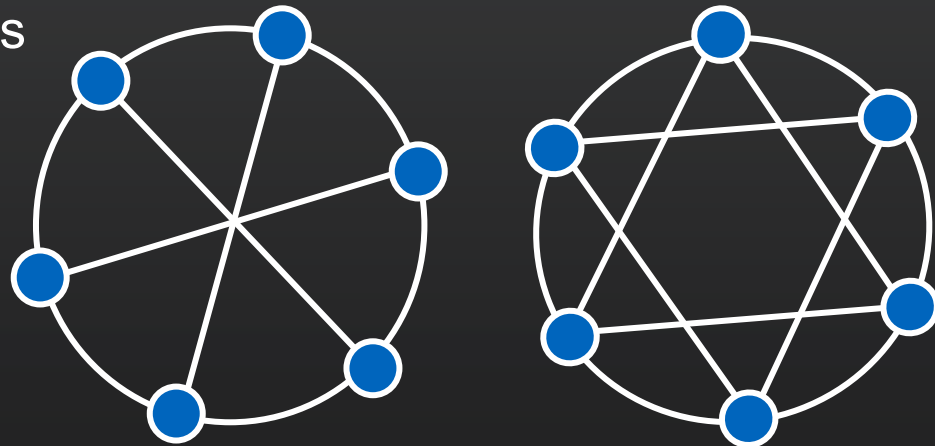
- ring
 - two-dimensional network
 - N nodes and N edges
 - degree = 2
 - diameter = $\lfloor N/2 \rfloor$
 - bisection width = 2
 - drawback: too slow for large N
 - how about fault tolerance?



Static Network Topologies

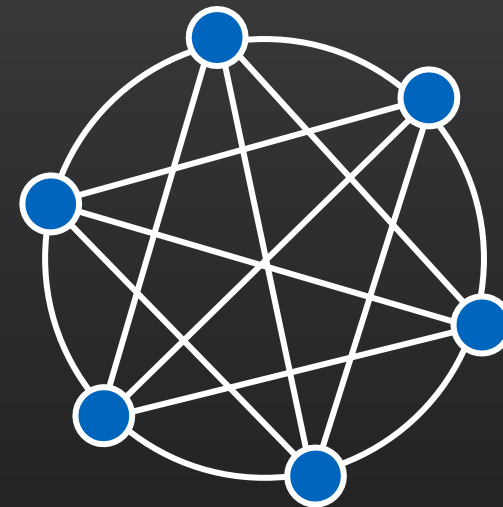
- chordal ring
 - two-dimensional network
 - N nodes and $3N/2, 4N/2, 5N/2, \dots$ edges
 - degree = 3, 4, 5, ...
 - higher degrees lead to
 - smaller diameters
 - higher fault tolerance (due to redundant connections)
 - drawback: higher costs

ring with degree = 3 (left)
and degree = 4 (right)



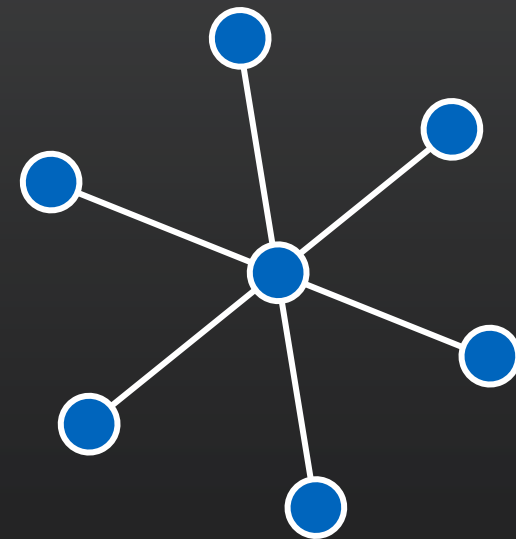
Static Network Topologies

- **completely connected**
 - two-dimensional network
 - N nodes and $N \cdot (N-1) / 2$ edges
 - degree = $N-1$
 - diameter = 1
 - bisection width = $\lfloor N/2 \rfloor \cdot \lceil N/2 \rceil$
 - very high fault tolerance
 - drawback: too expensive for large N



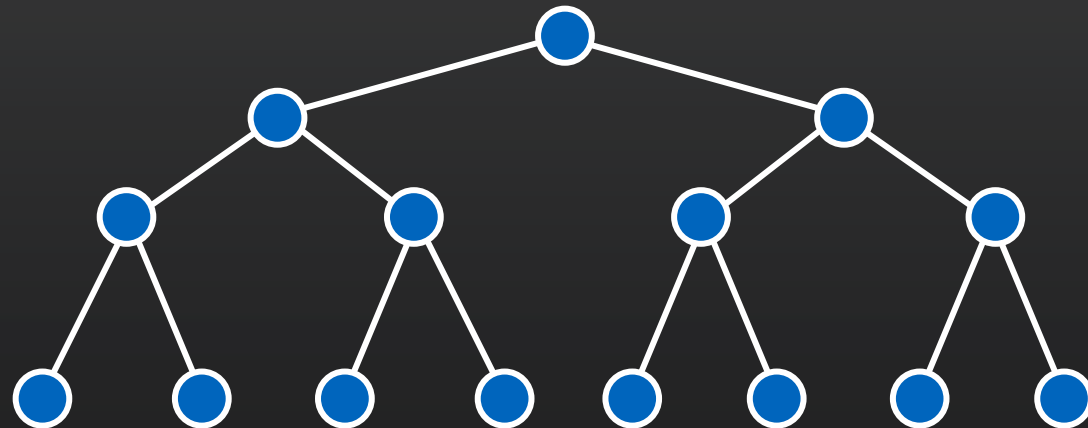
Static Network Topologies

- **star**
 - two-dimensional network
 - N nodes and $N-1$ edges
 - degree = $N-1$
 - diameter = 2
 - bisection width = $\lfloor N/2 \rfloor$
 - drawback: bottleneck in central node



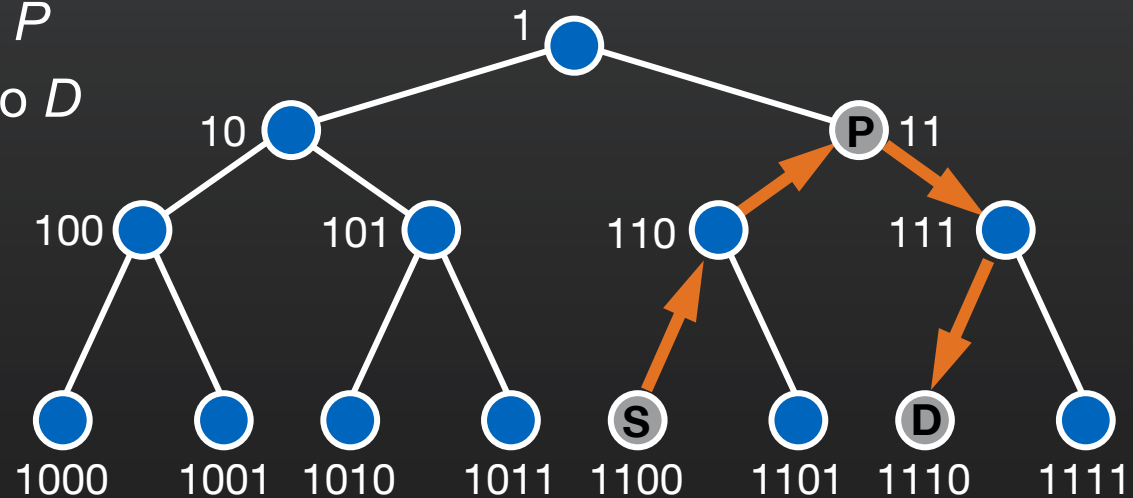
Static Network Topologies

- **binary tree**
 - two-dimensional network
 - N nodes and $N-1$ edges (tree height $h = \lfloor \lg N \rfloor$)
 - degree = 3
 - diameter = $2h$
 - bisection width = 1
 - drawback: bottleneck in direction of root (\rightarrow blocking)



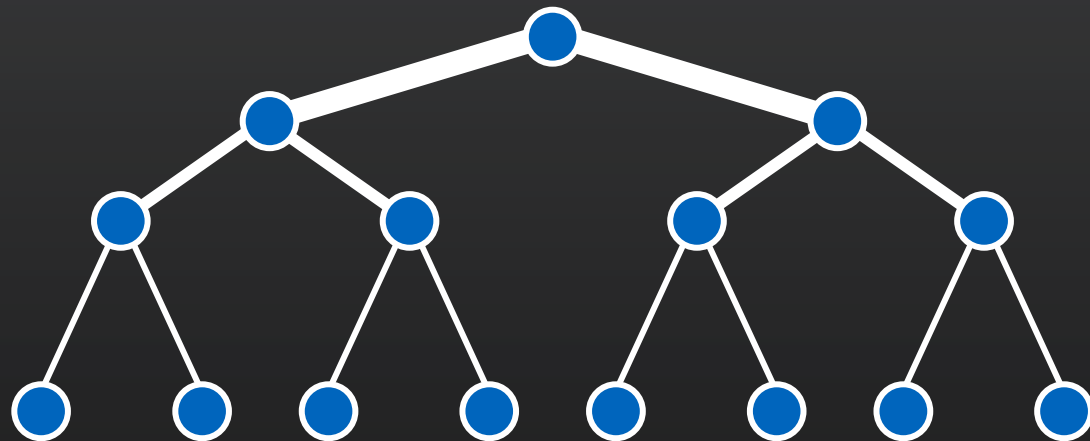
Static Network Topologies

- **binary tree (cont'd)**
 - addressing
 - label on level m consists of m bits; root has label "1"
 - suffix "0" is added to left son, suffix "1" is added to right son
 - routing
 - find common parent node P of nodes S and D
 - ascend from S to P
 - descend from P to D



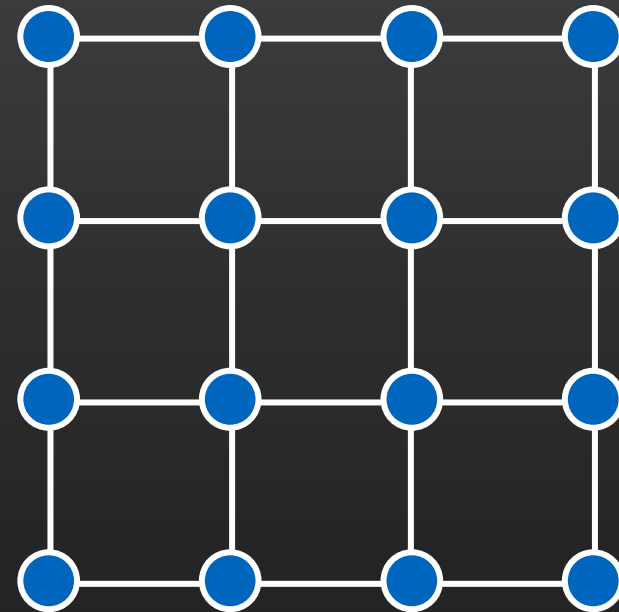
Static Network Topologies

- **binary tree (cont'd)**
 - solution to overcome the bottleneck → fat tree
 - edges on level m get higher priority than edges on level $m+1$
 - capacity is doubled on each higher level
 - now, bisection width = 2^{h-1}
 - frequently used: HLRB II, e.g.



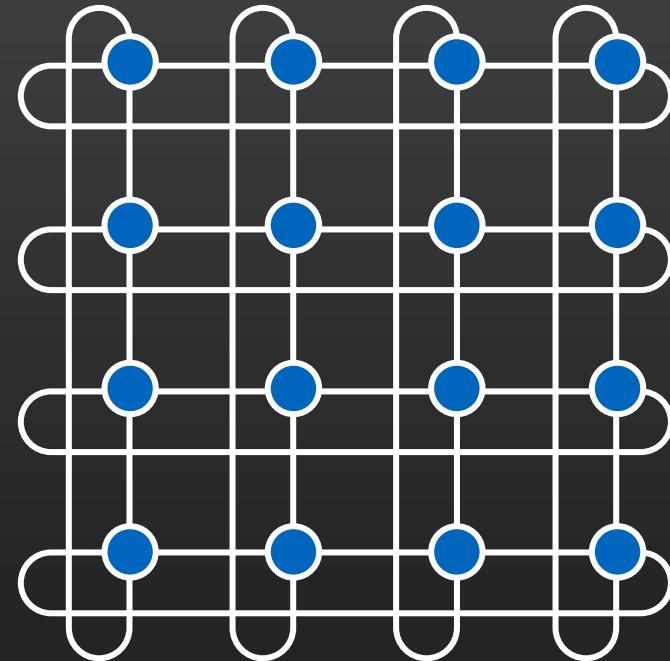
Static Network Topologies

- **mesh / torus**
 - k -dimensional network
 - N nodes and $k \cdot (N - r^{k-1})$ edges ($r = \sqrt[k]{N}$)
 - degree = $2k$
 - diameter = $k \cdot (r - 1)$
 - bisection width = r^{k-1}
 - high fault tolerance
 - drawback
 - large diameter
 - too expensive for $k > 3$



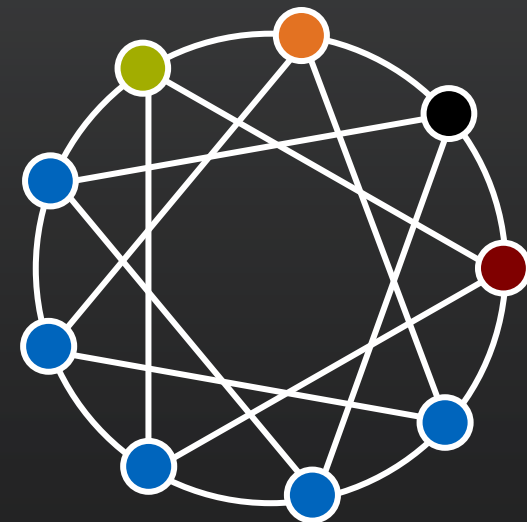
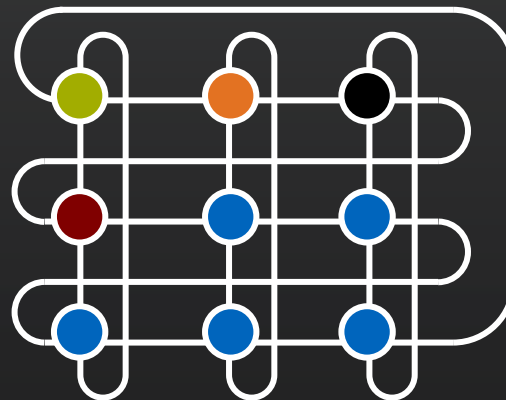
Static Network Topologies

- **mesh / torus (cont'd)**
 - k -dimensional mesh with cyclic connections in each dimension
 - N nodes and $k \cdot N$ edges ($r = \sqrt[k]{N}$)
 - diameter = $k \cdot \lfloor r/2 \rfloor$
 - bisection width = $2r^{k-1}$
 - frequently used: BlueGene/L, e.g.
 - drawback: too expensive for $k > 3$



Static Network Topologies

- **ILLIAC mesh**
 - two-dimensional network
 - N nodes and $2N$ edges ($r \times r$ mesh, $r = \sqrt{N}$)
 - degree = 4
 - diameter = $r-1$
 - bisection width = $2r$
 - conforms to a chordal ring of degree = 4



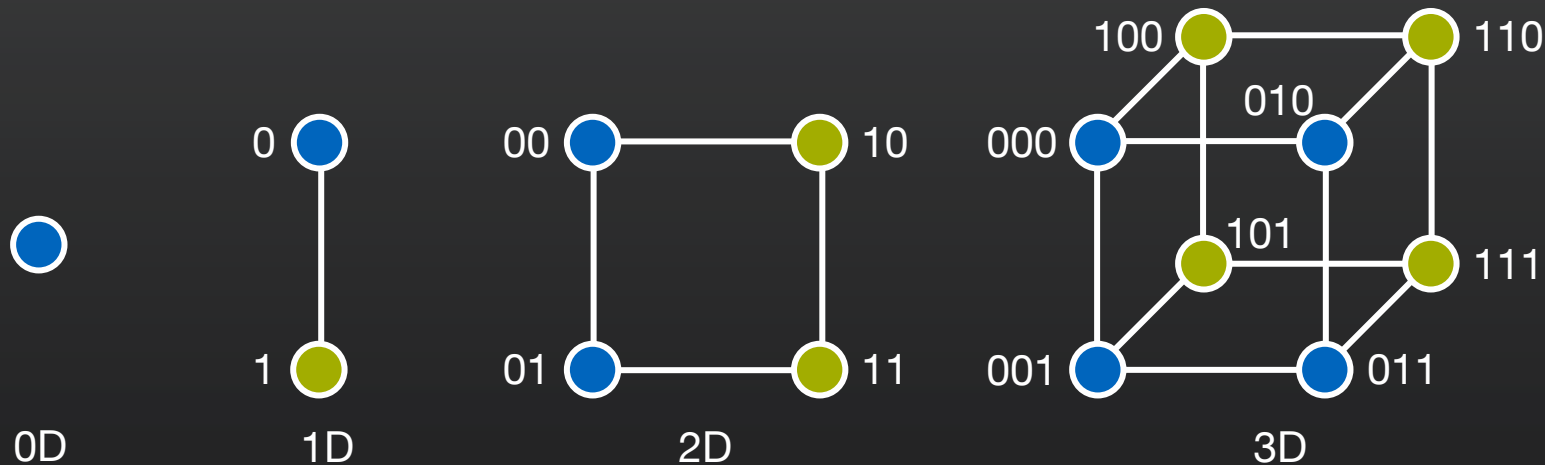
Static Network Topologies

- **hypercube**
 - k -dimensional network
 - 2^k nodes and $k \cdot 2^{k-1}$ edges
 - degree = k
 - diameter = k
 - bisection width = 2^{k-1}
 - drawback: scalability (only doubling of nodes allowed)



Static Network Topologies

- hypercube (cont'd)
 - principle design
 - construction of a k -dimensional hypercube via connection of the corresponding nodes of two $k-1$ -dimensional hypercubes
 - inherent labelling via adding prefix "0" to one sub-cube and prefix "1" to the other sub-cube

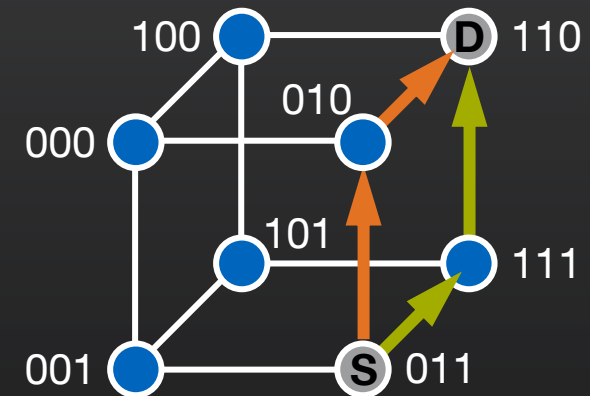


Static Network Topologies

- hypercube (cont'd)
 - nodes are directly connected for a HAMMING distance of 1 only
 - routing
 - compute $S \otimes D$ (xor) for possible ways between nodes S and D
 - route frames in increasingly / decreasingly order until final destination is reached

- example

- $S = "011", D = "110"$
- $S \otimes D = "101"$
- decreasing: $"011" \Rightarrow "010" \Rightarrow "110"$
- increasing: $"011" \Rightarrow "111" \Rightarrow "110"$



Overview

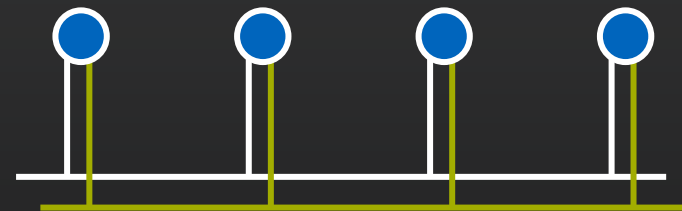
- some definitions
- static network topologies
- **dynamic network topologies**
- examples

Dynamic Network Topologies

- bus
 - simple and cheap single stage network
 - shared usage from all connected nodes, thus, just one frame transfer at any point in time
 - frame transfer in one step (i.e. diameter = 1)
 - good extensibility, but bad scalability
 - fault tolerance only for multiple bus systems
 - example: Ethernet



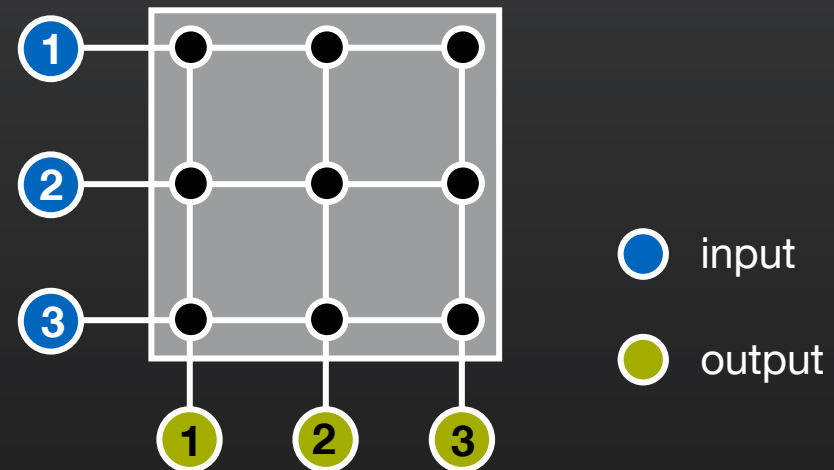
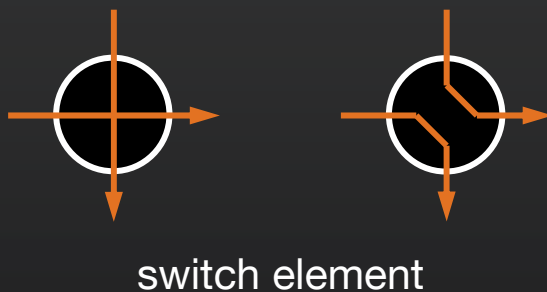
single bus



multiple bus (here dual)

Dynamic Network Topologies

- **crossbar**
 - completely connected network with all possible permutations of N inputs and N outputs (in general $N \times M$ inputs / outputs)
 - switch elements allow simultaneous communication between all possible disjoint pairs of inputs and outputs without blocking
 - very fast (diameter = 1), but expensive due to N^2 switch elements
 - used for processor—processor and processor—memory coupling
 - example: The Earth Simulator



Dynamic Network Topologies

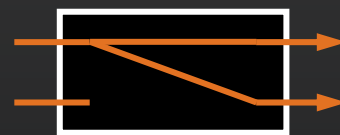
- **permutation networks**
 - tradeoff between low performance of buses and high hardware costs of crossbars
 - often 2×2 crossbar as basic element
 - N inputs can simultaneously be switched to N outputs \rightarrow permutation of inputs (to outputs)
 - single stage: consists of one column of 2×2 switch elements
 - multistage: consists of several of those columns



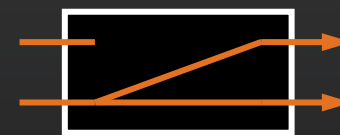
straight



crossed



upper
broadcast



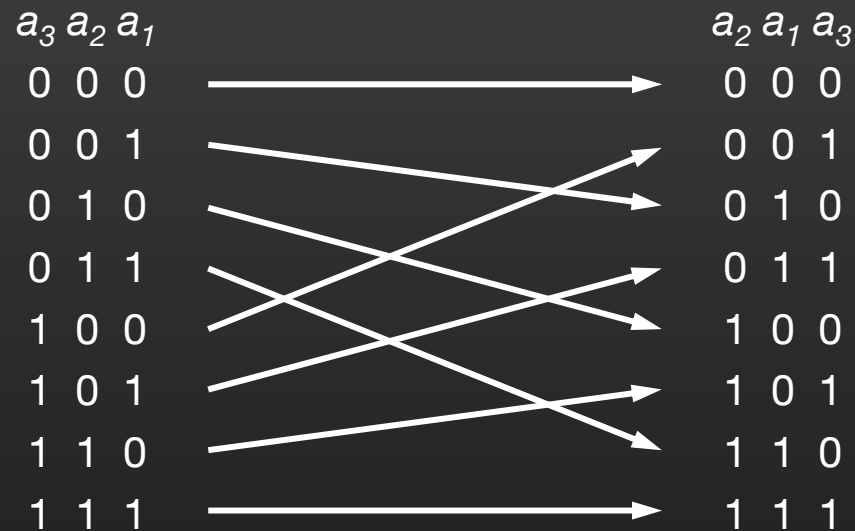
lower
broadcast

Dynamic Network Topologies

- permutation networks (cont'd)
 - permutations: unique (bijective) mapping of inputs to outputs
 - addressing
 - label inputs from 0 to $2N-1$ (in case of N switch elements)
 - write labels in binary representation ($a_K, a_{K-1}, \dots, a_2, a_1$)
 - permutations can now be expressed as simple bit manipulation
 - typical permutations
 - perfect shuffle
 - butterfly
 - exchange

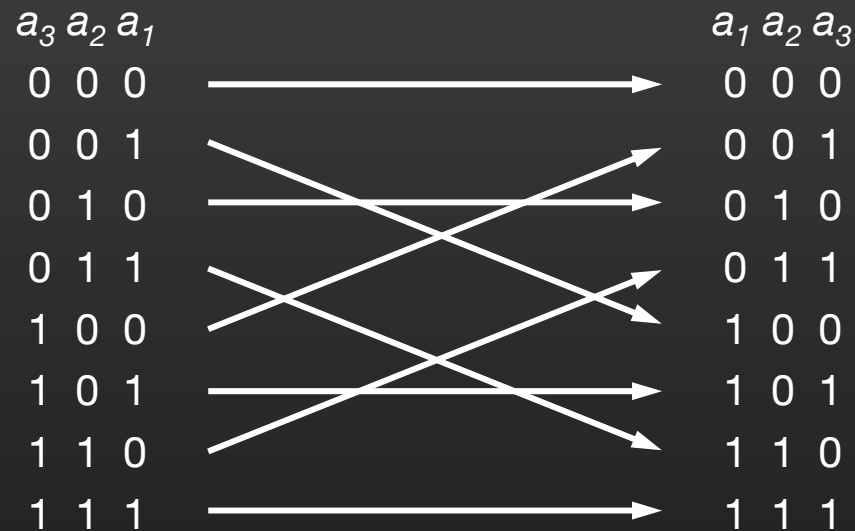
Dynamic Network Topologies

- permutation networks (cont'd)
 - perfect shuffle permutation
 - cyclic left shift
 - $P(a_K, a_{K-1}, \dots, a_2, a_1) \rightarrow (a_{K-1}, \dots, a_2, a_1, a_K)$



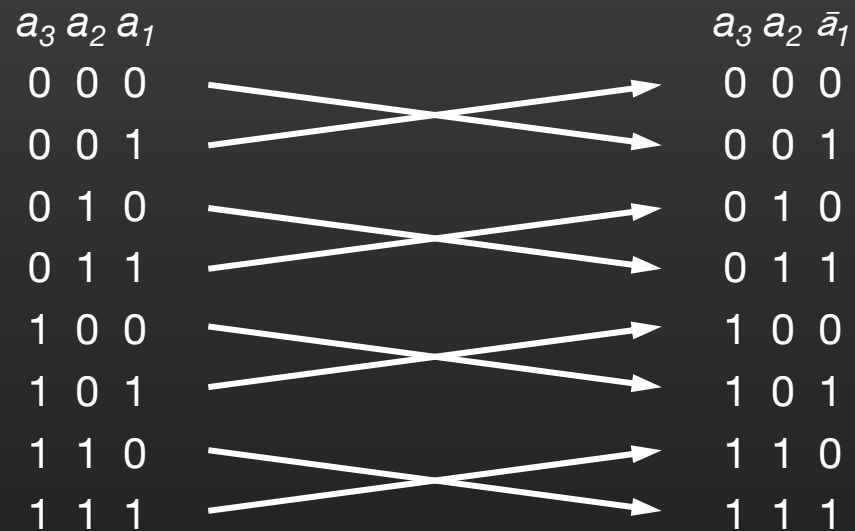
Dynamic Network Topologies

- permutation networks (cont'd)
 - butterfly permutation
 - exchange of first / highest and last / lowest bit
 - $B(a_K, a_{K-1}, \dots, a_2, a_1) \rightarrow (a_1, a_{K-1}, \dots, a_2, a_K)$



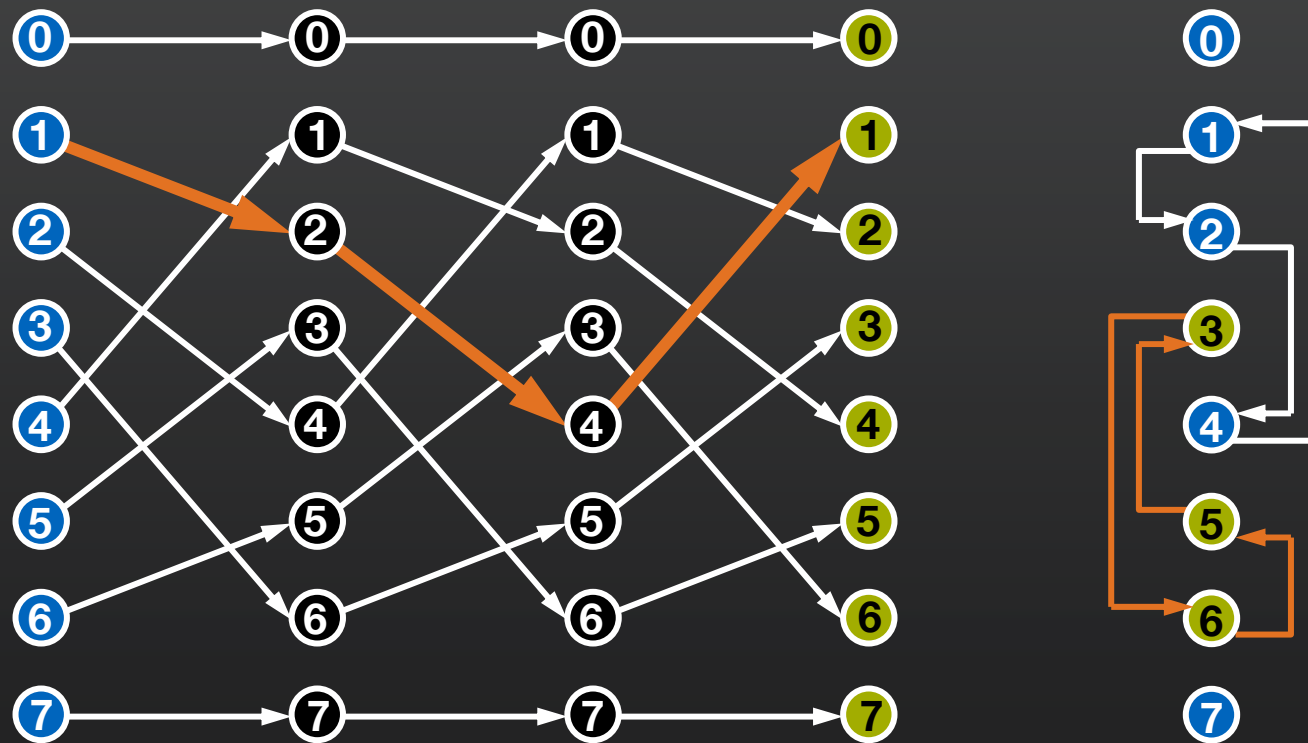
Dynamic Network Topologies

- permutation networks (cont'd)
 - exchange permutation
 - negation of last / lowest bit
 - $E(a_K, a_{K-1}, \dots, a_2, a_1) \rightarrow (a_K, a_{K-1}, \dots, a_2, \bar{a}_1)$



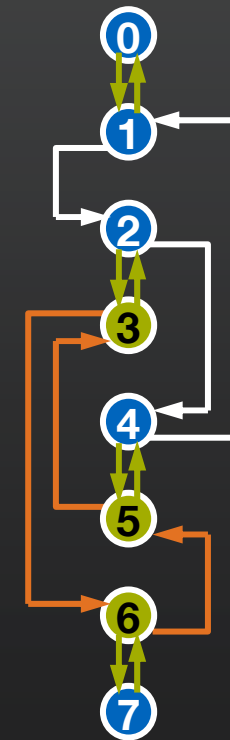
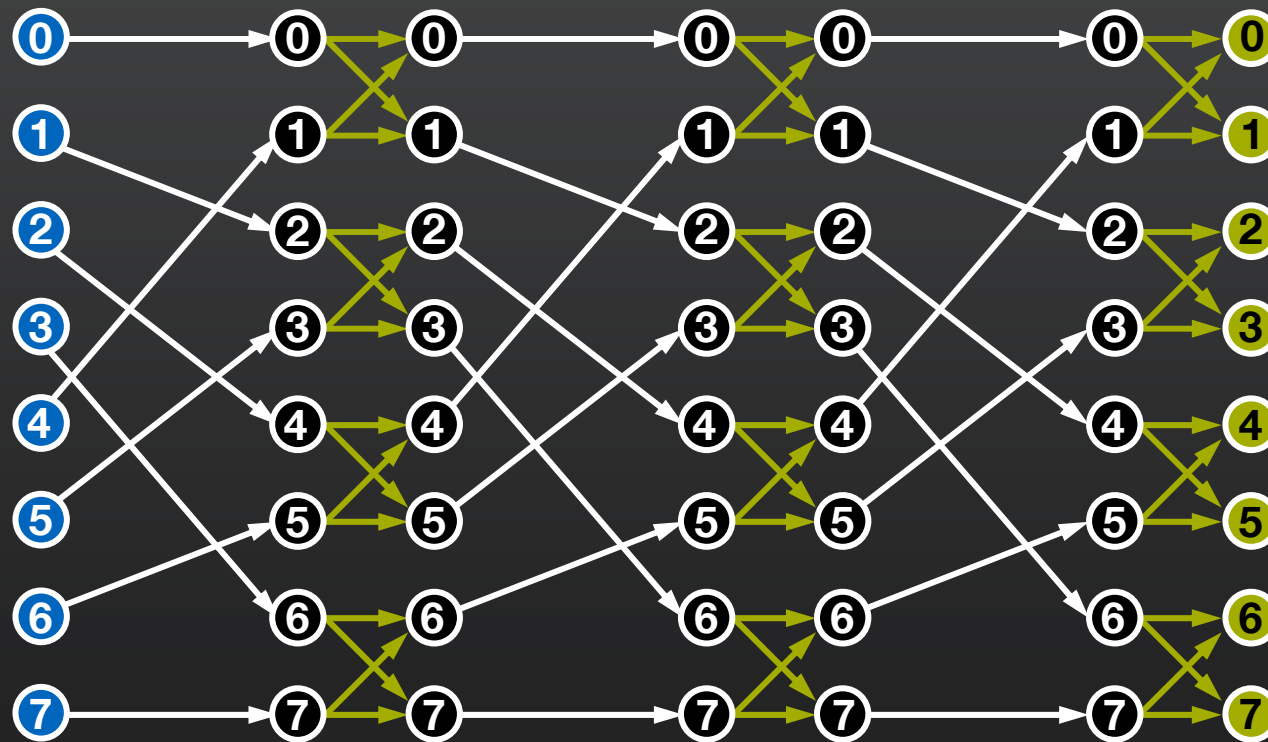
Dynamic Network Topologies

- permutation networks (cont'd)
 - example: perfect shuffle connection pattern
 - problem: not all destinations are accessible from a source



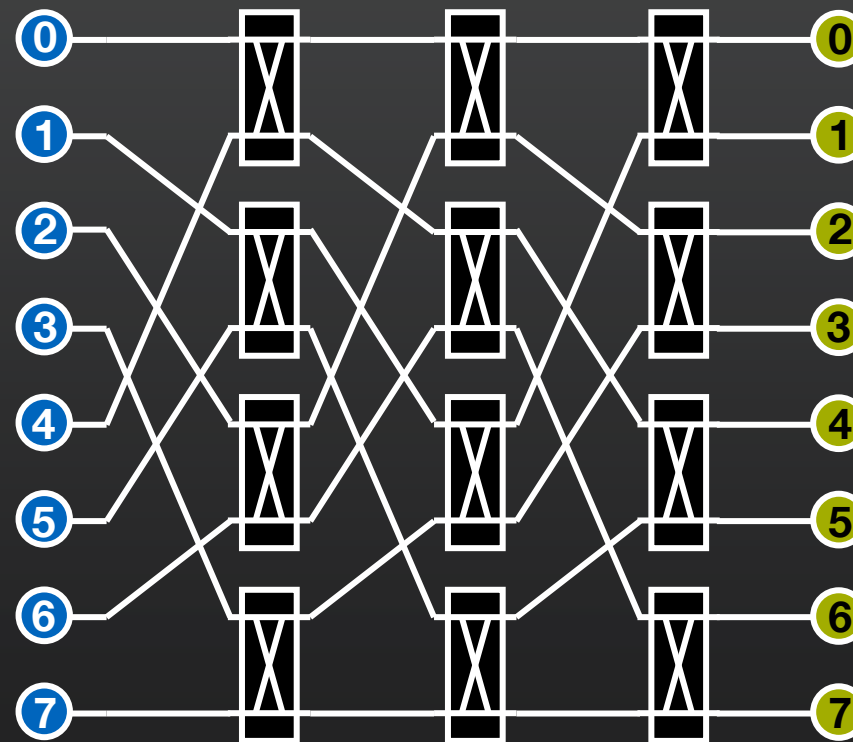
Dynamic Network Topologies

- permutation networks (cont'd)
 - adding additional exchange permutations (→ shuffle-exchange)
 - all destinations are now accessible from any source



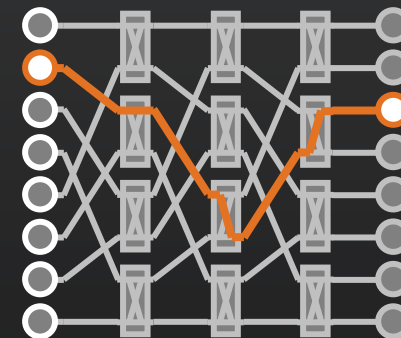
Dynamic Network Topologies

- **omega**
 - based on the shuffle-exchange connection pattern
 - exchange permutations replaced by 2×2 switch elements



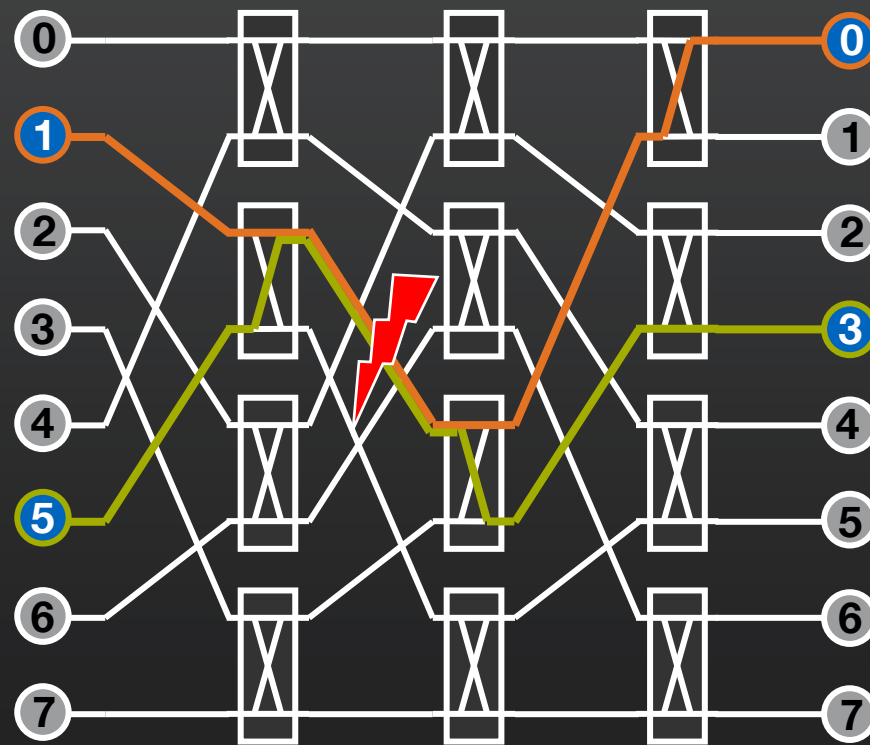
Dynamic Network Topologies

- **omega (cont'd)**
 - multistage network with N nodes and $E = N/2 \cdot \lg N$ switch elements
 - diameter = $\lg N$ (all stages have to be passed)
 - $N!$ permutations possible, but only 2^E different switch states
 - (self configuring) routing
 - compare addresses from S and D bitwise from left to right, i.e. stage i evaluates address bits s_i and d_i
 - if equal switch straight (–), otherwise switch crossed (×)
- example
 - $S = "001"$, $D = "010"$
 - switch states: – × ×



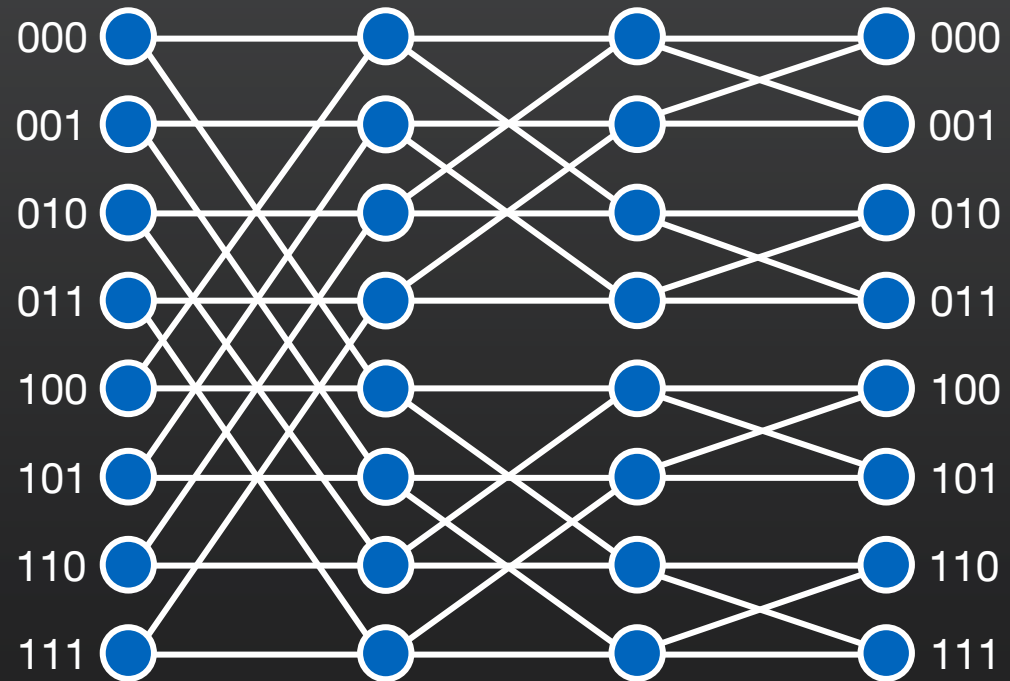
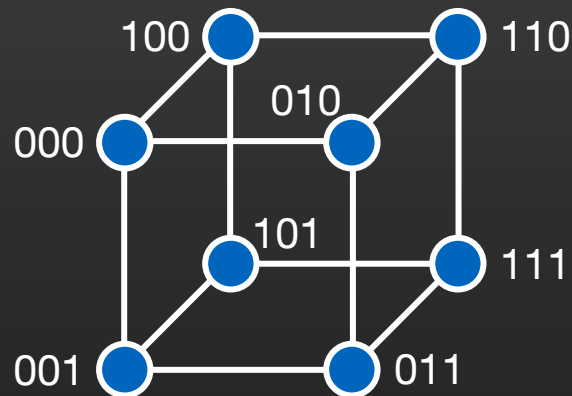
Dynamic Network Topologies

- **omega (cont'd)**
 - omega is a *bidelta network* → operates also backwards
 - drawback: blocking possible



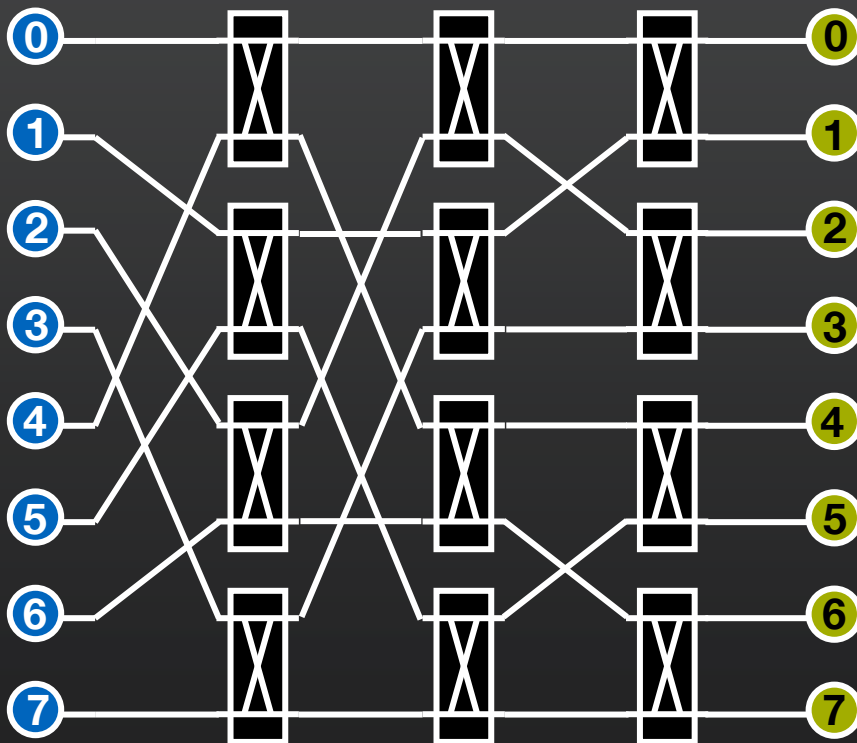
Dynamic Network Topologies

- **banyan / butterfly**
 - idea: unrolling of a static hypercube
 - bitwise processing of address bits a_i from left to right \rightarrow dynamic hypercube a.k.a. butterfly (known from FFT flow diagram)



Dynamic Network Topologies

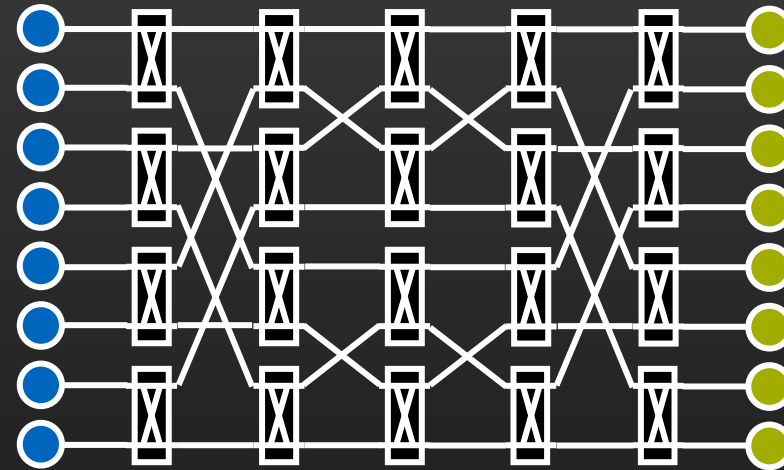
- **banyan / butterfly (cont'd)**
 - replace crossed connections by 2×2 switch elements
 - introduced by GOKE and LIPOVSKI in 1973; blocking still possible



banyan tree

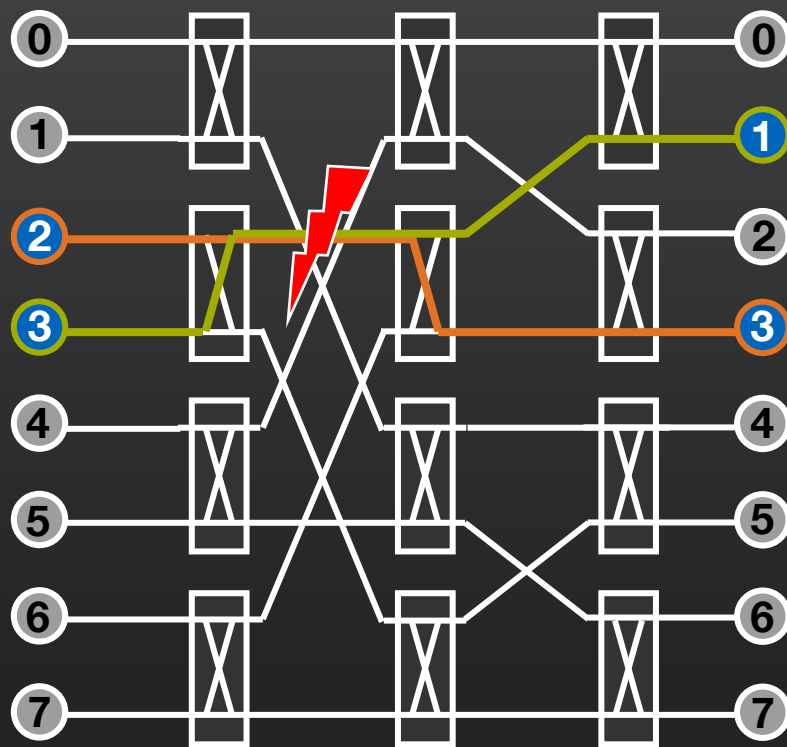
Dynamic Network Topologies

- **BENEŠ**
 - multistage network with N nodes and $N \cdot (\lg N) - N/2$ switch elements
 - butterfly merged at the last column with its copied mirror
 - diameter = $2(\lg N) - 1$
 - $N!$ permutations possible, all can be switched
 - key property: for any permutation of inputs to outputs there is a contention-free routing



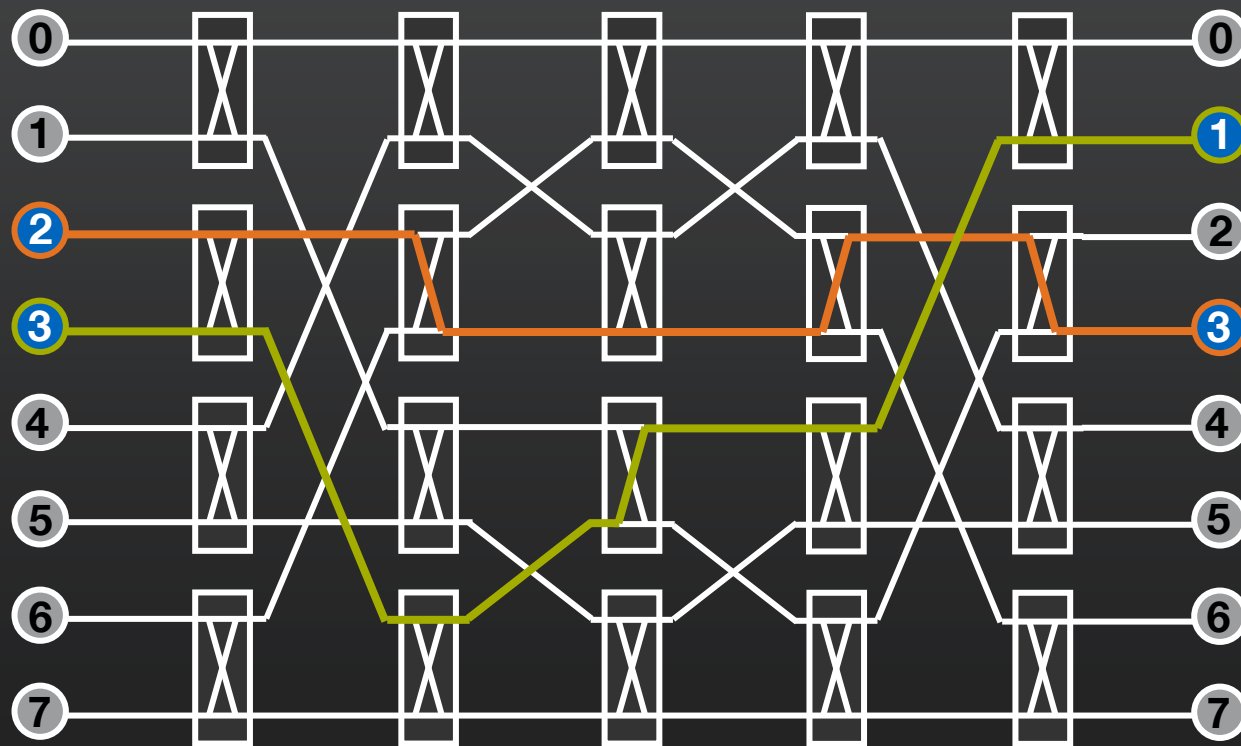
Dynamic Network Topologies

- **BENEŠ (cont'd)**
 - example
 - $S_1 = 2, D_1 = 3$ and $S_2 = 3, D_2 = 1 \rightarrow$ blocking for butterfly



Dynamic Network Topologies

- BENEŠ (cont'd)
 - example
 - $S_1 = 2, D_1 = 3$ and $S_2 = 3, D_2 = 1 \rightarrow$ no blocking for BENEŠ

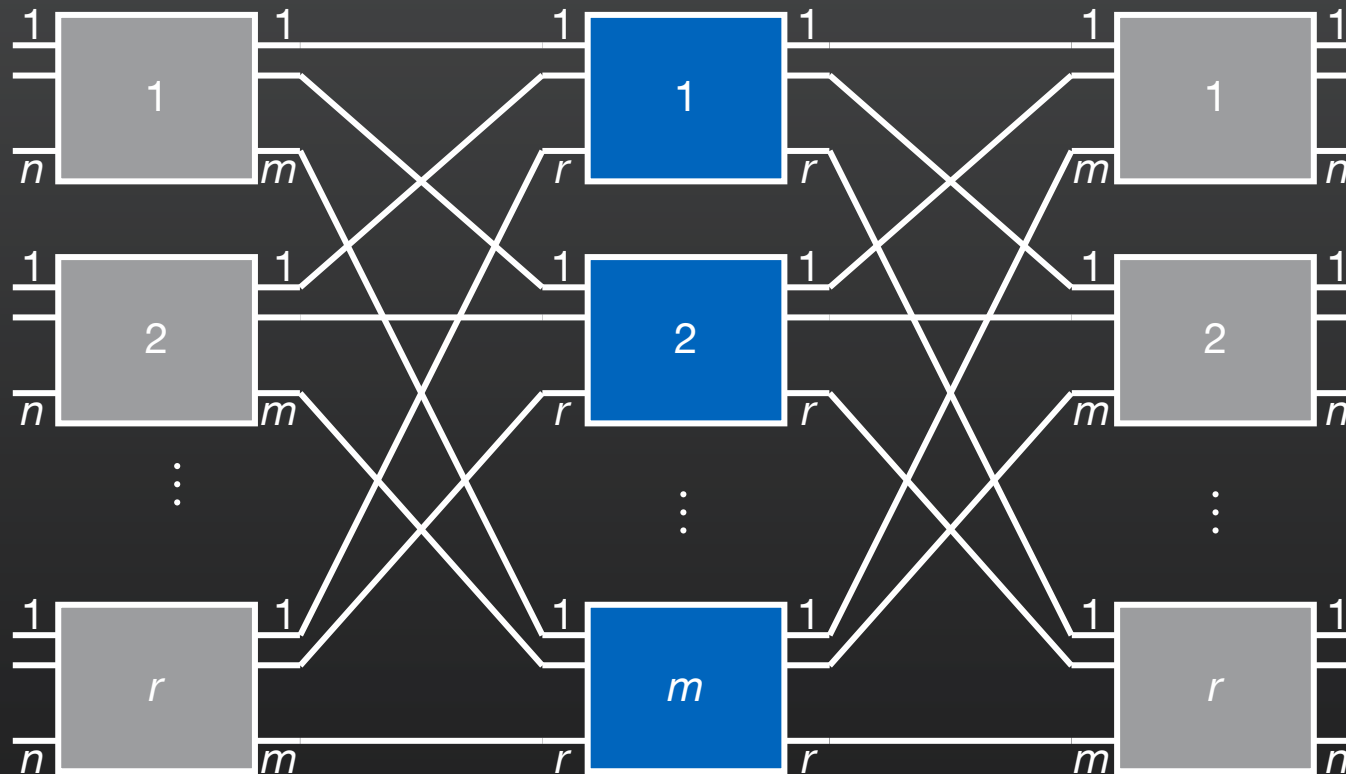


Dynamic Network Topologies

- **CLOS**
 - proposed by CLOS in 1953 for telephone switching systems
 - objective: overcome the costs of crossbars (N^2 switch elements)
 - *idea*: replace the entire crossbar with three stages of smaller ones
 - *ingress stage*: R crossbars with $N \times M$ inputs / outputs
 - *middle stage*: M crossbars with $R \times R$ inputs / outputs
 - *egress stage*: R crossbars with $M \times N$ inputs / outputs
 - thus much fewer switch elements than for the entire system
 - any incoming frame is routed from the input via one of the middle stage crossbars to the respective output
 - a middle stage crossbar is available if both links to the ingress and egress stage are free

Dynamic Network Topologies

- CLOS (cont'd)
 - $R \cdot N$ inputs can be assigned to $R \cdot N$ outputs

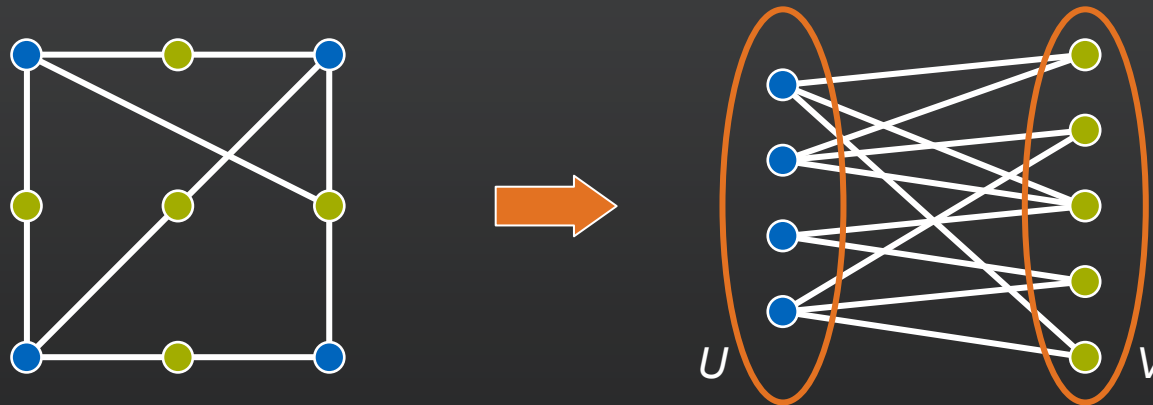


Dynamic Network Topologies

- CLOS (cont'd)
 - relative values of M and N define the blocking characteristics
 - $M \geq N$: *rearrangeable non-blocking*
 - a free input can always be connected to a free output
 - existing connections might be assigned to different middle stage crossbars (rearrangement)
 - $M \geq 2N-1$: *strict-sense non-blocking*
 - a free input can always be connected to a free output
 - no re-assignment necessary

Dynamic Network Topologies

- **reminder: bipartite graph**
 - *definition:* a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V ; that is, U and V are each independent sets

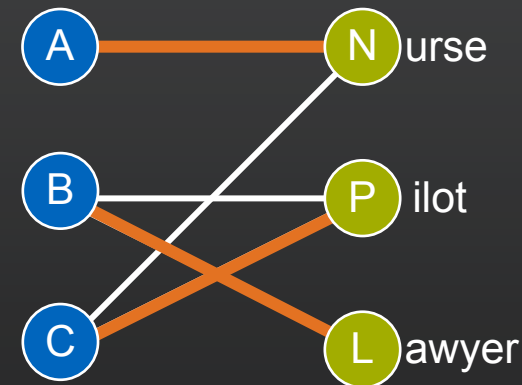


division of vertices in U and V , i.e. there are **no edges within U and V** , only between U and V

Dynamic Network Topologies

- **reminder: perfect matching**
 - *definition:* perfect matching (a.k.a. 1-factor) is a matching that matches all vertices of a graph, i.e. every vertex is incident to exactly one edge of the matching

	nurse	pilot	lawyer
Alice	✓		
Bob		✓	✓
Carol	✓	✓	



problem: perfect matching for bipartite graph to be found

Dynamic Network Topologies

- CLOS (cont'd)
 - proof for $M \geq N$ via HALL's "Marriage Theorem" (1)

Let $G = (V_{IN}, V_{OUT}, E)$ be a bipartite graph. A *perfect matching* for G is an injective function $f: V_{IN} \rightarrow V_{OUT}$ so that for every $x \in V_{IN}$, there is an edge in E whose endpoints are x and $f(x)$. One would expect a perfect matching to exist if G contains "enough" edges, i.e. if for every subset $A \subset V_{IN}$ the image set $\delta A \subset V_{OUT}$ is sufficient large.

Theorem: G has a perfect matching if and only if for every subset $A \subset V_{IN}$ the inequality $|A| \leq |\delta A|$ holds.

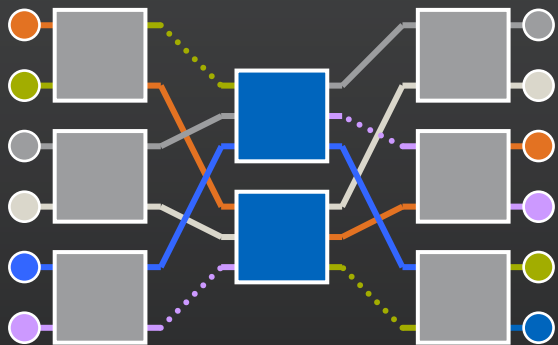
Often explained as follows: Imagine two groups of N men and N women. If any subset of S boys (where $0 \leq S \leq N$) knows S or more girls, each boy can be married with a girl he knows.

Dynamic Network Topologies

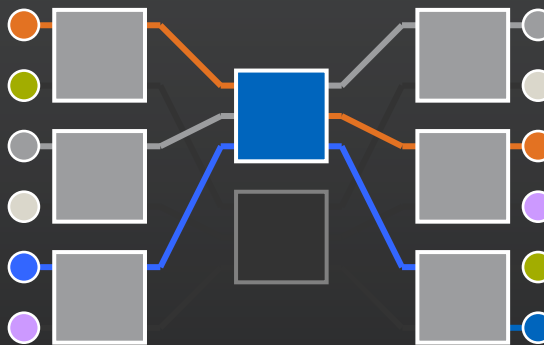
- CLOS (cont'd)
 - proof for $M \geq N$ via HALL's "Marriage Theorem" (2)
 - ▶ boy := ingress stage crossbar
 - ▶ girl := egress stage crossbar
 - ▶ a boy knows a girl if there exists a (direct) connection between them
 - ▶ assume there's one free input and one free output left
- 1) for $0 \leq S \leq R$ boys there are $S \cdot N$ connections \rightarrow at least S girls
 - 2) thus, HALL's theorem states there exists a perfect matching
 - 3) R connections can be handled by one middle stage crossbar
 - 4) bundle these connections and delete the middle stage crossbar
 - 5) repeat from step 1) until $M = 1$
 - 6) new connection can be handled, maybe rearrangement necessary \square

Dynamic Network Topologies

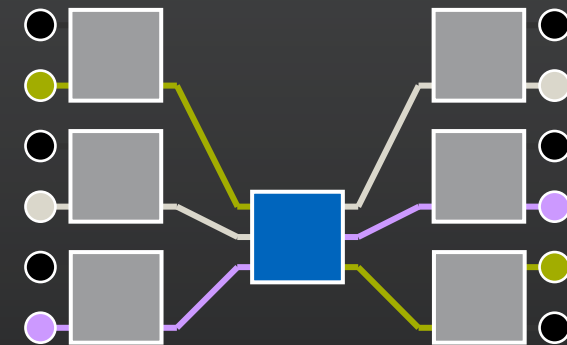
- CLOS (cont'd)
 - proof for $M \geq N$ via HALL's "Marriage Theorem" (3)
- example: $M = N = 2$



initial situation: two connections cannot be established



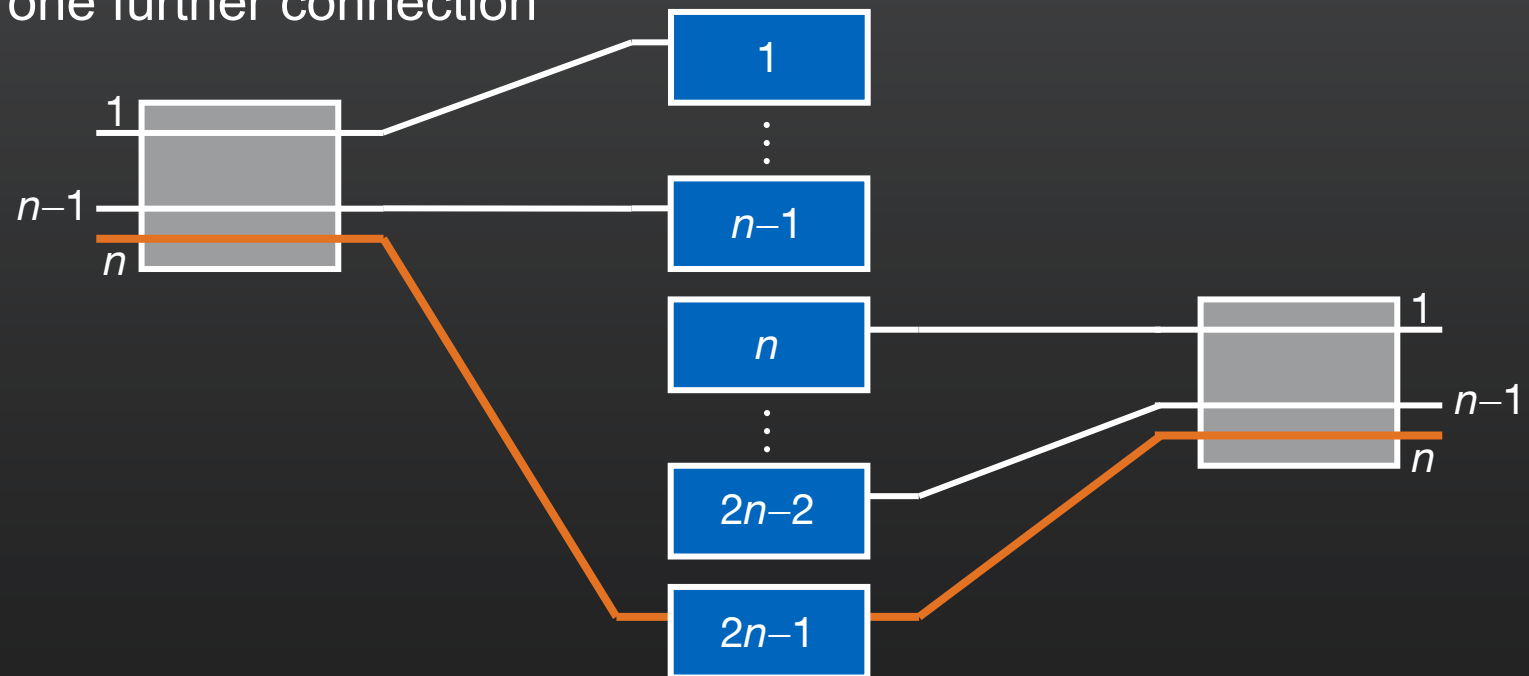
bundle connections on one middle stage crossbar and delete it afterwards → maybe rearrangements are necessary



repeat steps until $M = 1$, then all connections should be possible

Dynamic Network Topologies

- CLOS (cont'd)
 - proof for $M \geq 2N-1$ via worst case scenario
 - crossbar with $N-1$ inputs and crossbar with $N-1$ outputs, all connected to different middle stage crossbars
 - one further connection

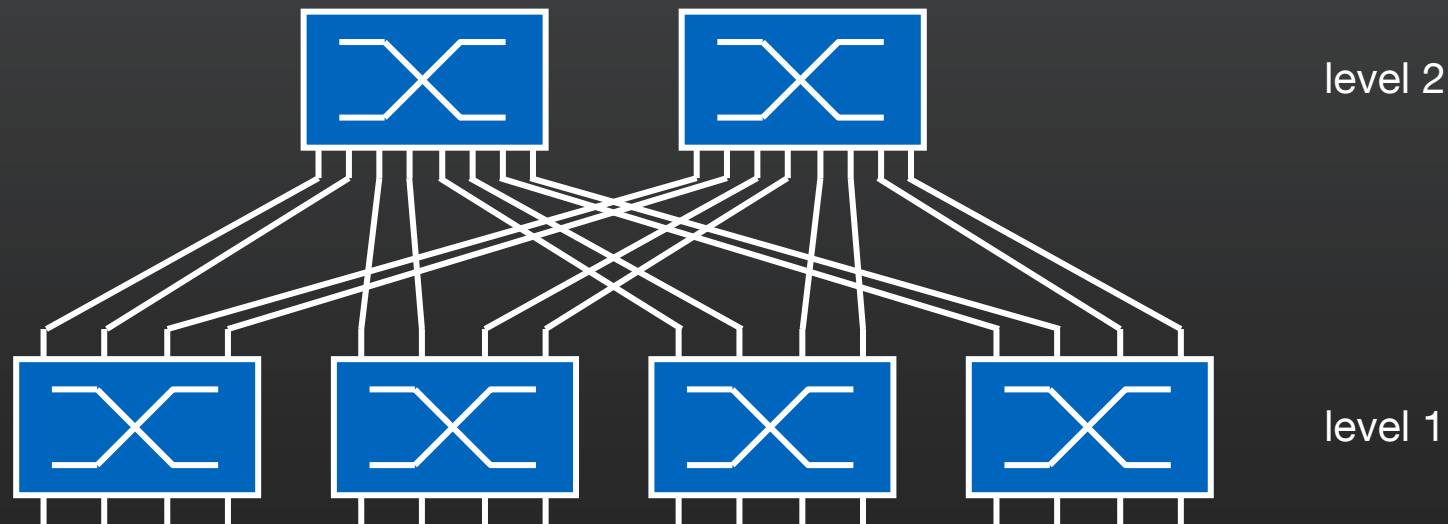


Dynamic Network Topologies

- **constant bisection bandwidth (CBB)**
 - more general concept of CLOS and fat tree networks
 - construction of a non-blocking network connecting M nodes
 - using multiple levels of basic $N \times N$ switch elements ($M > N$)
 - for any given level, downstream BW (in direction to nodes) is identical to upstream BW (in direction to interconnection)
 - *key for non-blocking: always preserve identical bandwidth (upstream and downstream) between any two levels*
 - observation: for two-stage constant bisection bandwidth networks connecting M nodes always $3M$ ports (i.e. sum of inputs and outputs) are necessary
 - CBB frequently used for high-speed interconnects (InfiniBand, e.g.)

Dynamic Network Topologies

- constant bisection bandwidth (cont'd)
 - example: CBB connecting 16 nodes with 4×4 switch elements → in total 48 ports (i.e. 6 switch elements) are necessary



Overview

- some definitions
- static network topologies
- dynamic network topologies
- **examples**

Examples

- in the past years, different (proprietary) high-performance networks have established on the market
- typically, these consist of
 - a static and / or dynamic network topology
 - sophisticated network interface cards (NIC)
- popular networks
 - Myrinet
 - InfiniBand
 - Scalable Coherent Interface (SCI)

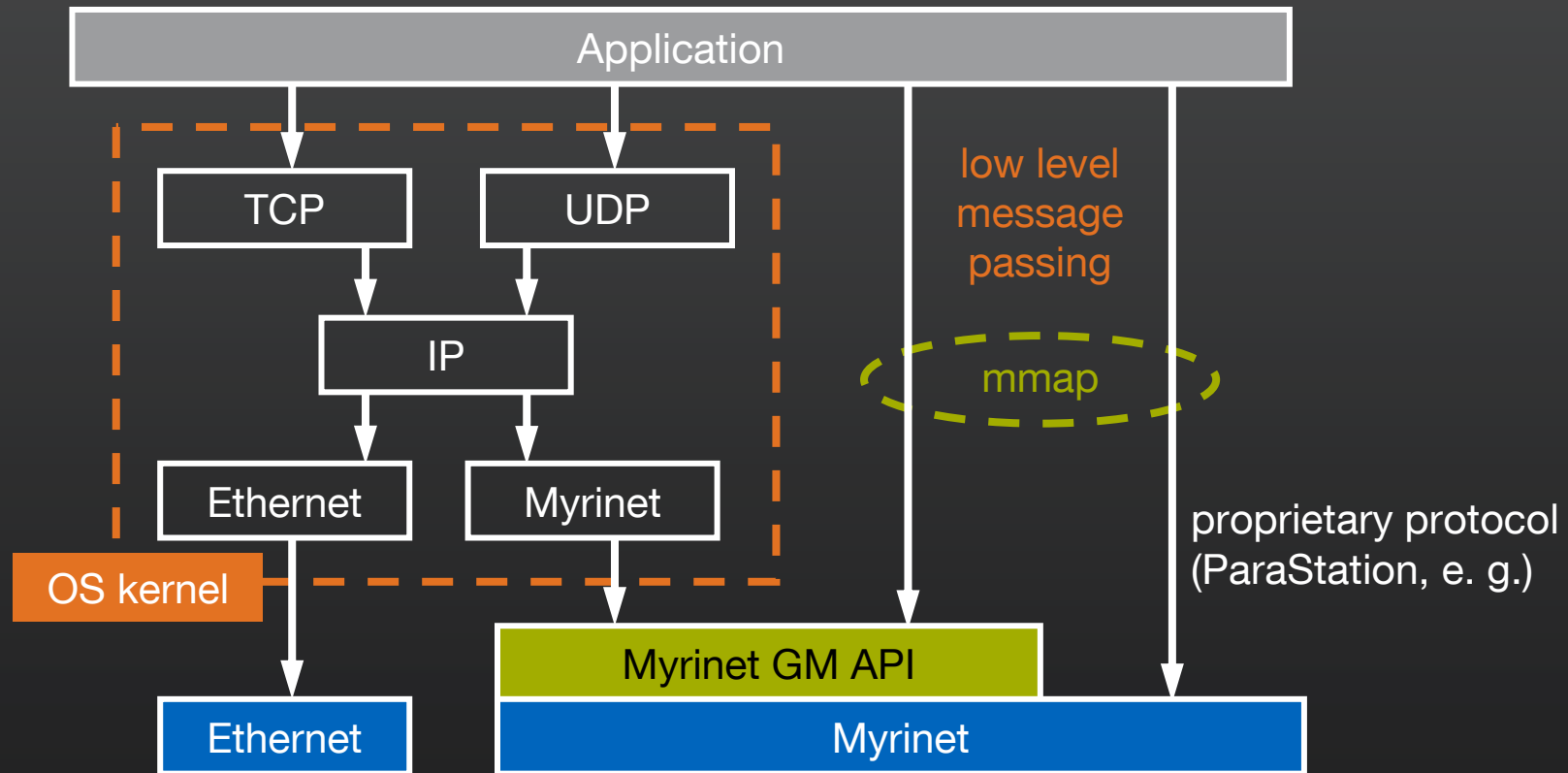
Examples

- **Myrinet**
 - developed by Myricom (1994) for clusters
 - particularly efficient due to
 - usage of onboard (NIC) processors for protocol offload and low-latency, kernel-bypass operations (ParaStation, e.g.)
 - highly scalable, cut-through switching
 - switching
 - rearrangeable non-blocking CLOS (128 nodes)
 - “spine” of CLOS network consists of eight 16×16 crossbars
 - nodes are connected via line-cards with 8×8 crossbar each



Examples

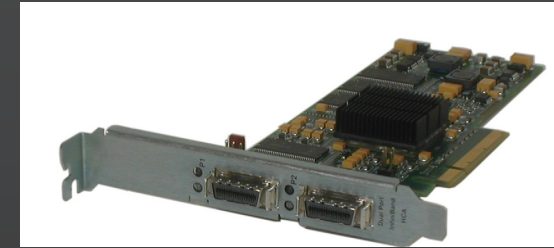
- Myrinet (cont'd)
 - programming model



Examples

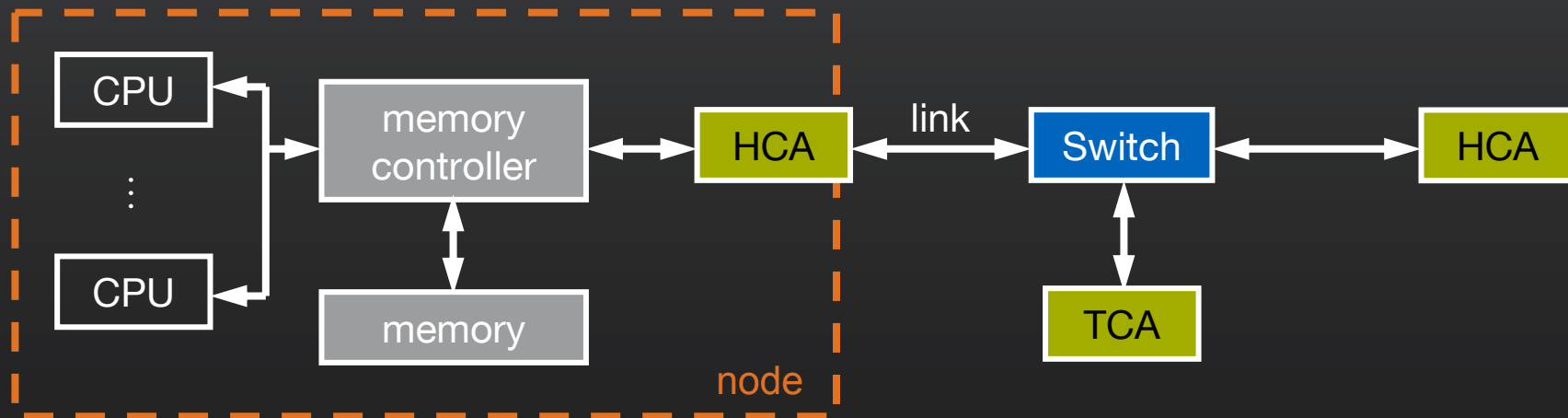
- **InfiniBand**

- unification of two competing efforts in 1999
 - Future I/O initiative (Compaq, IBM, HP)
 - Next-Generation I/O initiative (Dell, Intel, SUN et al.)
- idea: introduction of a future I/O standard as successor for PCI
 - overcome the bottleneck of limited I/O bandwidth
 - connection of hosts (via host channel adapters (HCA)) and devices (via target channel adapters (TCA)) to the I/O “fabric”
- switched point-to-point bidirectional links
- bonding of links for bandwidth improvements: 1× (up to 5Gbps), 4× (up to 20Gbps), 8× (up to 40Gbps), and 12× (up to 60Gbps)
- nowadays only used for cluster connection



Examples

- **InfiniBand (cont'd)**
 - particularly efficient (among others) due to
 - protocol offload and reduced CPU utilisation
 - Remote Direct Memory Access (RDMA), i.e. direct R/W access via HCA to local/remote memory without CPU usage/interrupts
 - switching: constant bisection bandwidth (up to 288 nodes)



Examples

- **Scalable Coherent Interface (SCI)**
 - originated as an offshoot from IEEE Futurebus+ project in 1988
 - became IEEE standard in 1992
 - SCI is a high performance interconnect technology that
 - connects up to 64,000 nodes (both hosts and devices)
 - supports remote memory access for read/write (NUMA)
 - uses packet switching point-to-point communication
 - SCI controller monitors I/O transactions (memory) to assure cache coherence of all attached nodes, i.e. all write accesses that invalidate cache entries of other SCI modules are detected
 - performance: up to 1GBps with latencies smaller than $2\mu\text{s}$
 - different topologies such as ring or torus possible

Examples

- **Scalable Coherent Interface (cont'd)**
 - shared memory: SCI uses a 64-bit fixed addressing scheme
 - upper 16 bits: node on which physical storage is located
 - lower 48 bits: local physical address within memory
 - hence, any physical memory location of the entire memory space can be mapped into a node's local memory

